SIEMENS

SIMATIC

S7
Программируемый контроллер S7-1200

Системное руководство

Предисловие

Обзор продукта 1

Монтаж 2

Основы ПЛК 3

Конфигурирование устройств 4

Основы программирования 5

Руководство по программированию 6

PROFINET 7

Двухточечная связь (Point-to-Point, PtP) 8

Инструментальные средства онлайнового режима и диагностики 9

Технические данные A

Расчет баланса мощностей B

Номера для заказа C

11/2009
A5E02669003-02

Правовая информация

Система предупреждений

Это руководство содержит указания, которые вы должны учитывать для обеспечения вашей личной безопасности и предотвращения материального ущерба. Указания, относящиеся к вашей личной безопасности, выделены в руководстве предупреждающим знаком, указания, касающиеся только материального ущерба, не имеют предупреждающего знака. Эти указания представлены ниже в порядке убывания степени опасности.

ОПАСНОСТЬ

означает, что если надлежащие меры предосторожности не будут приняты, то это **приведет** к гибели людей или тяжким телесным повреждениям.

ПРЕДУПРЕЖДЕНИЕ

означает, что если надлежащие меры предосторожности не будут приняты, то это **может привести** к гибели людей или тяжким телесным повреждениям.

ОСТОРОЖНО

с предупреждающим знаком означает, что если надлежащие меры предосторожности не будут приняты, то это может привести к легким телесным повреждениям.

ОСТОРОЖНО

без предупреждающего знака означает, что если надлежащие меры предосторожности не будут приняты, то это может привести к материальному ущербу.

ВНИМАНИЕ

означает, что если соответствующее указание не будет принято во внимание, то это может привести к нежелательному результату или состоянию.

При возникновении более одной степени опасности используется предупреждающе указание, соответствующее наивысшей степени опасности. Предупреждающее указание о возможности нанесения вреда людям с предупреждающим знаком может содержать также предупреждение о возможном материальном ущербе.

Квалифицированный персонал

Продукт/система, описанный в этой документации, может обслуживаться только **персоналом**, имеющим надлежащую **квалификацию** для решения поставленной задачи в соответствии с документацией, относящейся к этой задаче, в частности с указаниями по технике безопасности. Квалифицированный персонал – это люди, которые на основе своего образования и опыта способны распознавать риски и избегать потенциальных опасностей при работе с этими продуктами/системами.

Надлежащее использование продуктов фирмы Siemens

Примите во внимание следующее:

ПРЕДУПРЕЖДЕНИЕ

Продукты фирмы Siemens могут использоваться только для приложений, описанных в каталоге и в соответствующей технической документации. Если используются продукты и компоненты других производителей, то они должны быть рекомендованы или допущены фирмой Siemens. Обеспечение безопасной и безотказной работы предполагает надлежащую транспортировку, хранение, установку, монтаж, ввод в действие, управление и обслуживание. Должны соблюдаться допустимые условия окружающей среды. Должны быть приняты во внимание указания, содержащиеся в соответствующей документации.

Торговые марки

Все имена, помеченные знаком ®, являются зарегистрированными торговыми марками фирмы Siemens AG. Остальные обозначения в этой документации могут быть торговыми марками, использование которых третьими лицами для своих собственных целей могут нарушать права собственника.

Отказ от ответственности

Мы проверили содержание этой публикации на соответствие описанному программному и аппаратному обеспечению. Но так как отклонения не могут быть полностью исключены, мы не можем гарантировать полной согласованности. Однако информация, содержащаяся в этой публикации, регулярно пересматривается, и необходимые исправления вносятся в последующие издания.

Siemens AG
Сектор промышленности
п/я 48 48
90026 НЮРНБЕРГ
ГЕРМАНИЯ

Номер для заказа: 6ES7298-8FA30-8BH0
℗ 11/2009

Предисловие

Цель руководства

Семейство S7-1200 представляет собой серию программируемых логических контроллеров (ПЛК), с помощью которых можно решать широкий спектр задач автоматизации. Компактная конструкция, низкая стоимость и мощный набор команд делают S7-1200 в высшей степени пригодным для множества приложений в области управления. Различные модели S7-1200 и инструментальные средства программирования на основе Windows обеспечивают гибкость, необходимую вам для решения ваших задач автоматизации.

Это руководство содержит информацию об установке и программировании ПЛК S7-1200, и оно ориентировано на инженеров, программистов и обслуживающий персонал, имеющий общие знания о программируемых логических контроллерах.

Необходимые основные знания

Для понимания этого руководства необходимы общие знания об автоматизации и программируемых логических контроллерах.

Область применения руководства

Это руководство действительно для STEP 7 Basic V10.5 и семейства продуктов S7-1200. Полный список продуктов S7-1200, описанных в этом руководстве, вы найдете в технических данных (стр. 329).

Сертификация, метка CE, C-Tick и другие стандарты

Подробную информацию вы найдете в технических данных (стр. 329).

Обслуживание и поддержка

В дополнение к нашей документации мы предлагаем наши технические знания в Интернете по адресу: http://www.siemens.com/automation/support-request

Если у вас есть технические вопросы, вам нужно обучение, или вы хотите заказать продукты S7, обратитесь в свое представительство фирмы Siemens. Так как торговые представители фирмы Siemens технически хорошо подготовлены и имеют специальные знания о возможностях использования и процессах, а также о различных продуктах фирмы Siemens, то они могут быстрее всего дать наиболее эффективные ответы на любые проблемы, с которыми вы можете встретиться.

Содержание

Предисловие **3**

1 Обзор продукта **11**

1.1 Введение в ПЛК S7-1200 11

1.2 Сигнальные платы 13

1.3 Сигнальные модули 14

1.4 Коммуникационные модули 14

1.5 STEP 7 Basic 15
1.5.1 Различные представления для облегчения работы 16
1.5.2 Доступ к помощи в любом месте программы 17

1.6 Индикаторные панели 20

2 Монтаж **21**

2.2 Процедуры монтажа и демонтажа 24
2.2.1 Установка и удаление CPU 26
2.2.2 Установка и удаление сигнального модуля 28
2.2.3 Установка и удаление коммуникационного модуля 30
2.2.4 Установка и удаление сигнальной платы 32
2.2.5 Удаление и повторная установка клеммного блока S7-1200 33

2.3 Указания по подключению 34

3 Основы ПЛК **39**

3.1 Исполнение программы пользователя 39
3.1.1 Режимы работы CPU 42
3.1.2 Приоритеты и очереди для исполнения событий 46
3.1.3 Память CPU 52
3.1.4 Защита паролем для CPU S7-1200 57
3.1.5 Восстановление утерянного пароля 58

3.2 Память данных, области памяти и адресация 58

3.3 Типы данных 64

3.4 Использование карты памяти 68
3.4.1 Вставка карты памяти в CPU 69
3.4.2 Настройка параметров запуска CPU перед копированием проекта в карту памяти 70
3.4.3 Передаточная карта 70
3.4.4 Программная карта 73

4 Конфигурация устройств ... 77
4.1 Вставка CPU ... 78
4.2 Выявление конфигурации для заранее не заданного CPU ... 79
4.3 Конфигурирование работы CPU ... 80
4.4 Добавление модулей к конфигурации ... 81
4.5 Конфигурирование параметров модулей ... 82
4.6 Создание сетевого соединения ... 83
4.7 Конфигурирование IP-адреса в вашем проекте ... 84
5 Основы программирования ... 87
5.1 Указания по проектированию системы с ПЛК ... 87
5.2 Структурирование программы пользователя ... 88
5.3 Использование блоков для структурирования вашей программы ... 90
5.3.1 Организационный блок (OB) ... 92
5.3.2 Функция (FC) ... 93
5.3.3 Функциональный блок (FB) ... 94
5.3.4 Блок данных (DB) ... 95
5.4 Согласованность данных ... 96
5.5 Выбор языка программирования ... 97
5.6 Защита от копирования ... 99
5.7 Загрузка элементов вашей программы в CPU ... 100
5.8 Загрузка элементов вашей программы из CPU ... 101
5.9 Отладка и тестирование программы ... 102
6 Руководство по программированию ... 103
6.1 Основные команды ... 103
6.1.1 Двоичная логика ... 103
6.1.1.1 Команды установки и сброса ... 106
6.1.1.2 Команды нарастающий и падающий фронт ... 109
6.1.2 Таймеры ... 112
6.1.3 Счетчики ... 116
6.1.3.1 Счетчики ... 116
6.1.3.2 Команда CTRL_HSC ... 119
6.1.3.3 Принцип действия скоростных счетчиков ... 121
6.1.3.4 Конфигурирование скоростного счетчика ... 124
6.1.4 Сравнение ... 125
6.1.5 Арифметические команды ... 127
6.1.5.1 Команда MOD (получение остатка от деления) ... 128
6.1.6 Команда Move ... 136
6.1.6.1 Команда Swap (обмен байтов) ... 140
6.1.7 Преобразование ... 141
6.1.7.1 Команды масштабирования и нормализации ... 143
6.1.8 Управление программой ... 144
6.1.9 Логические операции ... 146
6.1.10 Операции сдвига и циклического сдвига ... 150

6.2 Расширенные команды ... 152
6.2.1 Общие параметры ошибок для расширенных команд ... 152
6.2.2 Команды для часов и календаря ... 153
6.2.3 Операции над строками и символами ... 156
6.2.3.1 Обзор данных строки ... 156
6.2.3.2 Команды преобразования строки ... 157
6.2.3.3 Операции со строками ... 168
6.2.4 Команды управления программой ... 176
6.2.4.1 Сброс контроля времени цикла ... 176
6.2.4.2 Команда остановки цикла ... 177
6.2.4.3 Команды Get Error ... 178
6.2.5 Коммуникационные операции ... 181
6.2.5.1 Обмен данными через открытый Ethernet ... 181
6.2.5.2 Команды для двухточечного соединения ... 196
6.2.6 Команды прерывания ... 197
6.2.6.1 Команды Attach и Detach ... 197
6.2.6.2 Команды запуска и отмены прерываний с задержкой ... 200
6.2.6.3 Команды активизации и деактивизации прерываний ... 203
6.2.7 PID-регулирование ... 203
6.2.8 Команды управления перемещением ... 204
6.2.9 Команда формирования импульсов ... 206
6.2.9.1 Команда CTRL_PWM ... 206
6.3 Глобальные библиотечные команды ... 210
6.3.1 USS ... 210
6.3.1.1 Предпосылки для использования протокола USS ... 210
6.3.1.2 Команда USS_DRV ... 213
6.3.1.3 Команда USS_PORT ... 216
6.3.1.4 Команда USS_RPM ... 217
6.3.1.5 Команда USS_WPM ... 219
6.3.1.6 Коды состояния USS ... 221
6.3.2 MODBUS ... 222
6.3.2.1 MB_COMM_LOAD ... 222
6.3.2.2 MB_MASTER ... 225
6.3.2.3 MB_SLAVE ... 237

7 PROFINET ... 249

7.1 Обмен данными с устройством программирования ... 251
7.1.1 Создание аппаратного коммуникационного соединения ... 251
7.1.2 Конфигурирование устройств ... 252
7.1.3 Назначение IP-адресов ... 252
7.1.3.1 Назначение IP-адресов устройству программирования и сетевым устройствам ... 252
7.1.3.2 Назначение IP-адресов в режиме online ... 255
7.1.3.3 Конфигурирование IP-адреса в вашем проекте ... 257
7.1.4 Тестирование сети PROFINET ... 259
7.2 Обмен данными между устройствами человеко-машинного интерфейса и ПЛК ... 262
7.2.1 Конфигурирование логических сетевых соединений между устройством человеко-машинного интерфейса и CPU ... 264
7.3 Обмен данными между ПЛК ... 265
7.3.1 Конфигурирование логических сетевых соединений между двумя CPU ... 266
7.3.2 Конфигурирование параметров передачи и приема ... 267
7.3.2.1 Конфигурирование параметров передачи для TSEND_C ... 267
7.3.2.2 Конфигурирование параметров приема для TRCV_C ... 271
7.4 Справочные данные ... 275
7.4.1 Получение адреса Ethernet (MAC-адреса) для CPU ... 275

7.4.2 Конфигурирование синхронизирующего сетевого протокола (NTP) ... 277

8 Двухточечная связь (PtP) ... 279

8.1 Использование коммуникационных модулей RS232 и RS485 ... 280

8.2 Конфигурирование коммуникационных портов ... 281

8.3 Управление потоками ... 282

8.4 Конфигурирование параметров приема и передачи ... 284

8.5 Программирование обмена данными через PtP ... 290
8.5.1 Архитектура опроса ... 291

8.6 Команды для двухточечного соединения ... 292
8.6.1 Общие параметры команд для двухточечного соединения ... 292
8.6.2 Команда PORT_CFG ... 294
8.6.3 Команда SEND_CFG ... 296
8.6.4 Команда RCV_CFG ... 298
8.6.5 Команда SEND_PTP ... 305
8.6.6 Команда RCV_PTP ... 308
8.6.7 Команда RCV_RST ... 309
8.6.8 Команда SGN_GET ... 310
8.6.9 Команда SGN_SET ... 311

8.7 Ошибки ... 312

9 Инструментальные средства онлайнового режима и диагностики ... 317

9.1 Светодиоды состояния ... 317

9.2 Создание онлайнового соединения с CPU ... 319

9.3 Установка IP-адреса и времени суток ... 320

9.4 Панель оператора для онлайнового CPU ... 320

9.5 Контроль времени цикла и использования памяти ... 321

9.6 Отображение диагностических событий в CPU ... 322

9.7 Таблицы наблюдения для контроля программы пользователя ... 323

A Технические данные ... 329

A.1 Общие технические данные ... 329

A.2 CPU ... 335
A.2.1 Технические данные CPU 1211C ... 335
A.2.2 Технические данные CPU 1212C ... 340
A.2.3 Технические данные CPU 1214C ... 345

A.3 Цифровые сигнальные модули (SM) ... 351
A.3.1 Технические данные цифрового модуля ввода SM 1221 ... 351
A.3.2 Технические данные цифрового модуля вывода SM 1222 ... 353
A.3.3 Технические данные цифрового модуля ввода/вывода SM 1223 ... 355

A.4 Аналоговые сигнальные модули (SM) ... 358
A.4.1 Технические данные аналоговых сигнальных модулей SM 1231, SM 1232, SM 1234 ... 358

A.5 Сигнальные платы (SB) ... 364
A.5.1 Технические данные SB 1223 2 X 24 VDC Input / 2 X 24 VDC Output ... 364
A.5.2 Технические данные SB 1232 с 1 аналоговым выходом ... 367

A.6 Коммуникационные модули (CM) 369
A.6.1 Технические данные CM 1241 RS485 369
A.6.2 Технические данные CM 1241 RS232 370
A.7 Карты памяти SIMATIC 370
A.8 Имитаторы входов 371
A.9 Кабель для расширения ввода/вывода 372
B Расчет баланса мощностей 373
B.2 Пример расчета потребности в мощности 375
B.3 Расчет вашей потребности в мощности 376
C Номера для заказа 377
Предметный указатель 381

1 Обзор продукта

1.1 Введение в ПЛК S7-1200

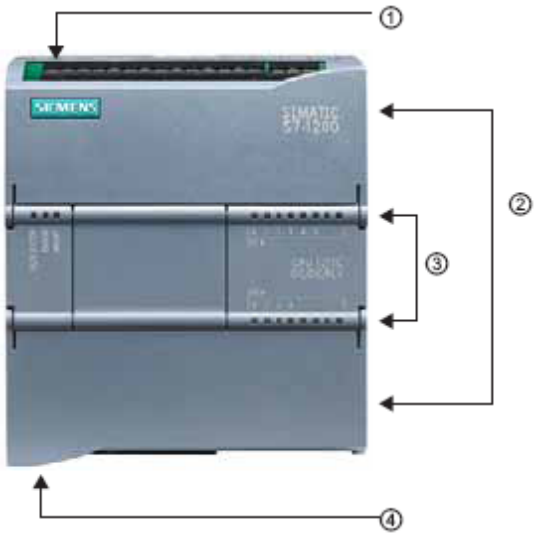
Программируемый логический контроллер (ПЛК) S7-1200 обеспечивает гибкость и поставляет достаточную мощность для управления широким кругом устройств, поддерживающих ваши потребности в автоматизации. Компактная конструкция, гибкая конфигурация и мощный набор команд, делают S7-1200 прекрасным решением для управления широким спектром приложений.

CPU объединяет в компактном корпусе микропроцессор, встроенный блок питания, входные и выходные цепи, образуя мощный ПЛК. После загрузки вашей программы CPU содержит логику, необходимую для контроля и управления устройствами в вашем приложении. CPU контролирует входы и изменяет выходы в соответствии с логикой вашей пользовательской программы, которая может включать булевы логические операции, счет, отсчет времени, сложные математические операции и связь с другими интеллектуальными устройствами.

Ряд функций обеспечения безопасности помогают защитить доступ как к CPU, так и к управляющей программе:

- Каждый CPU обеспечивает защиту паролем, которая позволяет вам формировать доступ к CPU в соответствии с вашими потребностями.
- Вы можете использовать "защиту ноу-хау", чтобы скрыть код внутри конкретного блока. Подробную информацию вы найдете в главе "Основы программирования" (стр. 99).

CPU снабжен портом PROFINET для обмена данными через сеть PROFINET. Для обмена данными через сети RS485 или RS232 в вашем распоряжении имеются коммуникационные модули.



① Разъем питания
② Съемный клеммный блок для подключения пользователя (за дверцами)
② Гнездо для карты памяти под верхней дверцей
③ Светодиоды состояния для встроенных входов/выходов
④ Разъем PROFINET (на нижней стороне CPU)

Различные модели CPU предлагают многообразные характеристики и возможности, которые помогают вам создавать эффективные решения для самых разных приложений. Подробные данные для конкретных CPU вы найдете в технических данных (стр. 329).

Характеристика	**CPU 1211C**	**CPU 1212C**	**CPU 1214C**
Физический размер (мм)	90 x 100 x 75		110 x 100 x 75
Пользовательская память • Рабочая память • Загрузочная память • Сохраняемая память	 • 25 Кбайт • 1 Мбайт • 2 Кбайта		 • 50 Кбайт • 2 Мбайта • 2 Кбайта
Локальные встроенные входы/выходы • цифровые • аналоговые	 • 6 входов/4 выхода • 2 входа	 • 8 входов/6 выходов • 2 входа	 • 14 входов/10 выходов • 2 входа
Величина образа процесса	1024 байта входов (I) и 1024 байта выходов (Q)		
Битовая память (M)	4096 байт		8192 байта
Дополнительные сигнальные модули	Нет	2	8
Сигнальная плата	1		
Коммуникационные модули	3 (левостороннее расширение)		
Скоростные счетчики • однофазные • со сдвигом фаз на 90°	3 • 3 на 100 кГц • 3 на 80 кГц	4 • 3 на 100 кГц 1 на 30 кГц • 3 на 80 кГц 1 на 20 кГц	6 • 3 на 100 кГц 3 на 30 кГц • 3 на 80 кГц 3 на 20 кГц
Импульсные выходы	2		
Карта памяти	Карта памяти SIMATIC (факультативно)		
Длительность сохранения времени для часов реального времени	Тип. 10 дней / 6 дней минимум при 40 градусах С.		
PROFINET	1 коммуникационный порт для связи с Ethernet		
Скорость выполнения арифметических операций	18 мкс/команду		
Скорость выполнения булевых операций	0,1 мкс/ команду		

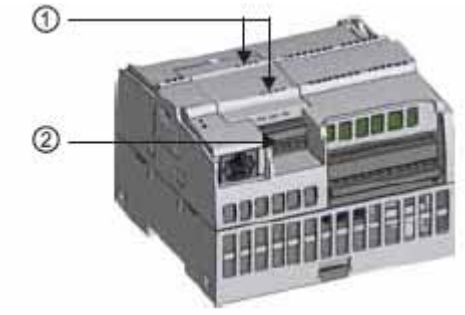
Семейство S7-1200 предлагает ряд сигнальных модулей и сигнальных плат для расширения возможностей CPU. Вы можете также устанавливать дополнительные коммуникационные модули для поддержки других протоколов связи. Подробную информацию о конкретных модулях вы найдете в технических данных (стр. 329).

Модуль		**Только ввод**	**Только вывод**	**Комбинация ввода и вывода**
Сигнальный модуль (SM)	Цифровой	8 входов пост. тока	8 выходов пост. тока 8 релейных выходов	8 входов пост. тока / 8 выходов пост. тока 8 входов пост. тока / 8 релейных выходов
		16 входов пост. тока	16 выходов пост. тока 16 релейных выходов	16 входов пост. тока / 16 выходов пост. тока 16 входов пост. тока / 16 релейных выходов
	Аналоговый	4 аналоговых входа 8 аналоговых входов	2 аналоговых выхода 4 аналоговых выхода	4 аналоговых входа/ 2 аналоговых выхода
Сигнальная плата (SB)	Цифровая	-	-	2 входа пост. тока/ 2 выхода пост. тока
	Аналоговая	-	1 аналоговый выход	-
Коммуникационный модуль (CM) • RS485 • RS232				

1.2 Сигнальные платы

Сигнальная плата (SB) предоставляет возможность добавлять входы/выходы к вашему CPU. Вы можете установить одну SB с цифровыми или аналоговыми входами/выходами. SB подключается спереди CPU.

- SB с 4 цифровыми входами/выходами (2 входа пост. тока и 2 выхода пост. тока)
- SB с 1 аналоговым выходом

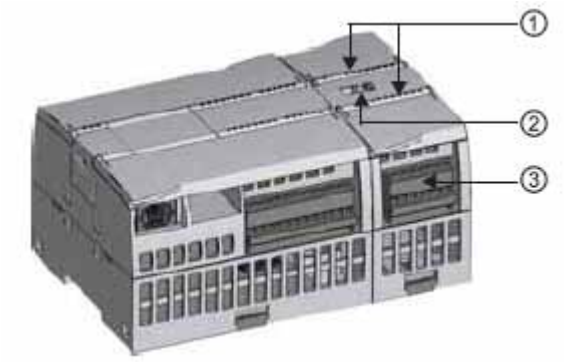


① Светодиоды состояния на SB

② Съемный клеммный блок для подключения пользователя

1.3 Сигнальные модули

Для расширения функциональных возможностей CPU вы можете использовать сигнальные модули. Сигнальные модули подключаются с правой стороны CPU.



① Светодиоды состояния входов/выходов сигнального модуля
② Шинный соединитель
③ Съемный клеммный блок для подключения пользователя

1.4 Коммуникационные модули

Семейство S7-1200 предоставляет в распоряжение коммуникационные модули (CM) для расширения функциональных возможностей системы. Имеются два коммуникационных модуля: RS232 и RS485.

- CPU поддерживает до 3 коммуникационных модулей
- Каждый CM подключается к левой стороне CPU (или к левой стороне другого CM)

① Светодиоды состояния для коммуникационного модуля
② Коммуникационный разъем

1.5 STEP 7 Basic

Программное обеспечение STEP 7 Basic предоставляет в распоряжение пользователя удобную среду для разработки, редактирования и контроля логики, необходимой для управления вашим приложением, включая инструментальные средства для управления и конфигурирования всех устройств в вашем проекте, таких как ПЛК и устройства человеко-машинного интерфейса. Для удобства и эффективности в разработке управляющей программы для вашего приложения STEP 7 Basic предоставляет в распоряжение два языка программирования (LAD и FBD), а также обеспечивает инструментальными средствами для создания и конфигурирования устройств человеко-машинного интерфейса в вашем проекте.

Чтобы помочь вам в поиске необходимой вам информации, STEP 7 Basic предоставляет обширную онлайновую систему помощи.

Для установки STEP 7 Basic вставьте компакт-диск в дисковод CD-ROM вашего компьютера. Мастер установки запускается автоматически и выдает вам подсказки в процесс установки. Для получения дальнейшей информации обратитесь к файлу Readme.

Указание

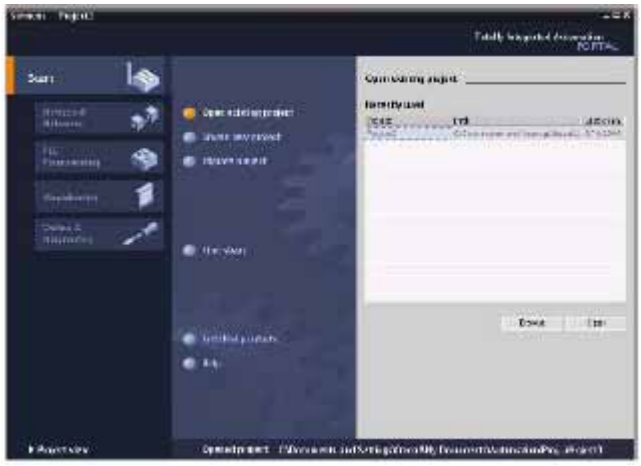
Для установки программного обеспечения STEP 7 Basic на ПК с операционной системой Windows 2000, Windows XP или Windows Vista вы должны войти в систему с привилегиями администратора.

1.5.1 Различные представления для облегчения работы

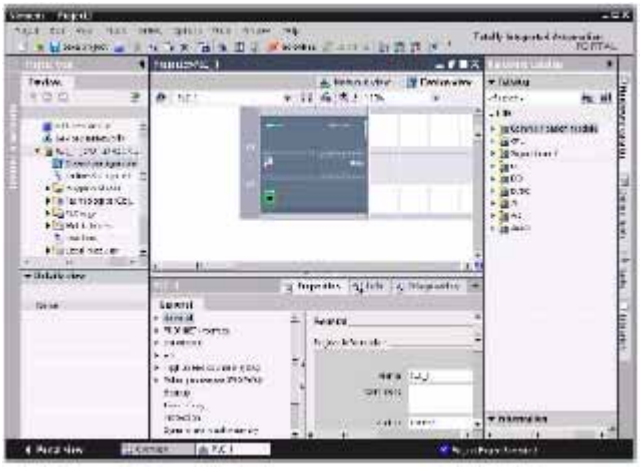
Для повышения вашей производительности портал комплексной автоматизации (Totally Integrated Automation Portal) предоставляет в ваше распоряжение два различных набора инструментальных средств: проблемно-ориентированный набор порталов для отдельных функций (портальное представление) и проектно-ориентированное представление элементов в проекте (проектное представление). Вы принимаете решение, в каком представлении вы сможете работать наиболее эффективно. Одним щелчком мыши вы можете переходить от одного представления к другому.

Портальное представление дает функциональный взгляд на задачи проекта и организует функции инструментальных средств в соответствии с задачами, которые должны быть выполнены, например, конфигурирование аппаратуры и сетей.

Вы можете легко определить, как вы хотели бы действовать и какую задачу выбрать.



Портальное представление дает доступ ко всем компонентам внутри проекта. Имея все эти компоненты в одном месте, вы получаете легкий доступ к любому аспекту вашего проекта. Проект содержит все создаваемые и готовые элементы.



1.5.2 Доступ к помощи в любом месте программы

Быстрые ответы на ваши вопросы

Чтобы вы могли быстро и эффективно решать ваши задачи, STEP 7 Basic предоставляет в ваше распоряжение интеллектуальную помощь там, где она нужна:

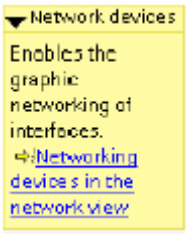
- Поле ввода предоставляет вам необходимую помощь для правильного ввода данных (допустимые диапазоны и тип данных) в этом поле. Если, например, вводится недопустимое значение, то появляется окно с текстом сообщения, содержащего допустимый диапазон.
- Некоторые из всплывающих подсказок в интерфейсе (например, для команд) представлены в виде "каскада", чтобы предоставить дополнительную информацию. Некоторые из этих каскадных подсказок дают также ссылки на родственные темы в онлайновой информационной системе (онлайновая помощь).

Кроме того, STEP 7 Basic содержит обширную информационную систему, которая полностью описывает функциональные возможности инструментальных средств SIMATIC.

Помощь при вводе и каскадные всплывающие подсказки

Некоторые поля ввода для диалоговых окон и карточек задач обеспечивают обратную связь в виде отображаемого на экране окна сообщения, которое информирует вас о необходимом диапазоне значений и типе данных.

Элементы программного интерфейса снабжены всплывающими подсказками, объясняющими функциональные возможности этого элемента. Некоторые из этих элементов, например, пиктограммы "Open [Открыть]" или "Save [Сохранить]", не требуют дополнительной информации. Однако некоторые из элементов снабжены механизмом для отображения дополнительного описания элемента. Эта дополнительная информация отображается в каскадной всплывающей подсказке. (Черный треугольник рядом со всплывающей подсказкой указывает, что имеется дополнительная информация.)



Всплывающая подсказка появляется, когда курсор находится над элементом программного интерфейса. Для получения дополнительной информации просто наведите курсор на всплывающую подсказку. Некоторые из каскадных всплывающих подсказок предоставляют также ссылки на соответствующие темы в информационной системе. Щелчок на этой ссылке отображает соответствующую тему.

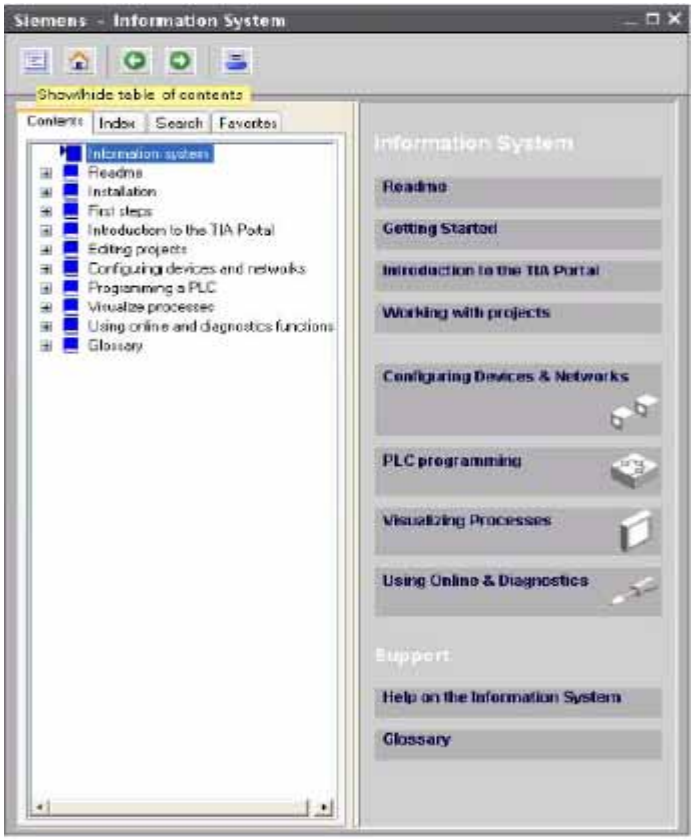
Информационная система

STEP 7 Basic предоставляет в распоряжение обширную онлайновую информационную систему, описывающую все продукты SIMATIC, которые вы установили. Эта информационная система включает в себя также справочную информацию и примеры. Для вызова информационной системы выберите одну из следующих точек доступа:

- В портальном представлении откройте стартовый портал и щелкните на команде "Help [Помощь]".
- В проектном представлении выберите в меню "Help [Помощь]" команду "Show help [Показать помощь]".
- В каскадной всплывающей подсказке щелкните на ссылке, чтобы отобразить дальнейшую информацию по этой теме.

Информационная система открывается в окне, которое не закрывает рабочие области.

Щелкните в информационной системе на кнопке "Show/hide contents [Показать/скрыть содержание]", чтобы отобразить содержание и освободить окно помощи. Тогда вы сможете изменять размер этого окна. Для выполнения поиска в информационной системе по теме или ключевому слову используйте вкладку "Contents [Содержание]" или "Index [Предметный указатель]".

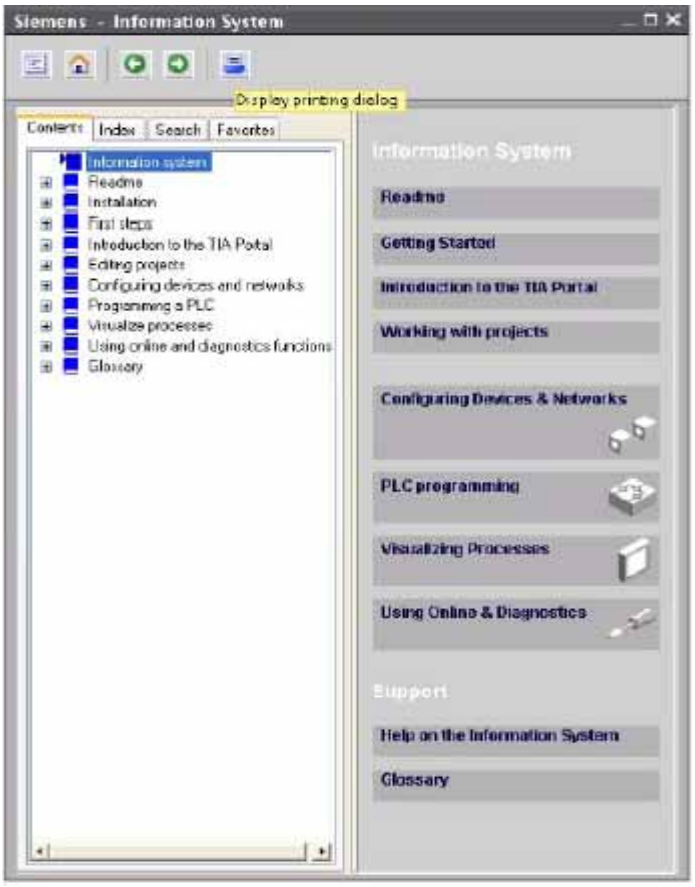
Окно помощи (по умолчанию)	**Окно помощи с открытым содержанием**
	

Указание

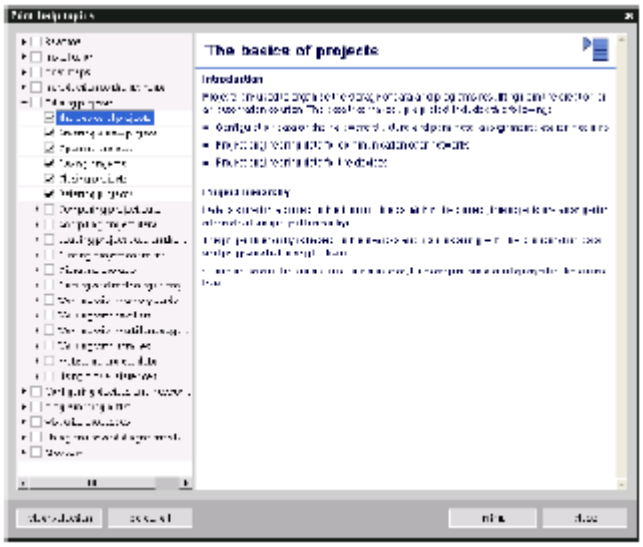
Если STEP 7 Basic максимизирован, то щелчок на кнопке "Show/hide contents" не освобождает окно помощи. Чтобы освободить окно помощи, щелкните на кнопке "Restore down [Восстановить свернутое состояние]". Тогда вы сможете перемещать и изменять размер окна помощи.

Распечатка тем из информационной системы

Для распечатки тем из информационной системы щелкните на кнопке "Print [Печатать]" в окне помощи.



Для распечатки тем из информационной системы щелкните на кнопке "Print [Печатать]" в окне помощи.



Диалоговое окно "Print [Печать]" позволяет выбирать темы для печати. Обратите внимание на то, чтобы окно содержало какую-нибудь тему. Затем вы сможете выбрать для печати любую другую тему. Щелкните на кнопке "Print [Печатать]", чтобы отправить выбранные темы на свой принтер.

1.6 Индикаторные панели

Так как визуализация становится стандартным компонентом конструкции большинства машин, то базовые панели человеко-машинного интерфейса SIMATIC предоставляют в распоряжение устройства с сенсорным экраном для реализации основных задач оператора по контролю и управлению. Все панели имеют степень защиты IP65 и имеют сертификаты CE, UL, cULus и NEMA 4x.

KTP 400 Basic PN

- Монохромный (STN, шкала уровня серого)
- Сенсорный экран 4" с 4 тактильными клавишами
- Ориентация книжная или альбомная
- Размер: 3.8"
- Разрешение: 320 x 240
- 128 переменных
- 50 изображений процесса
- 200 прерываний
- 25 кривых
- 32 KB памяти для рецептов
- 5 рецептов, 20 записей данных, 20 компонентов

KTP 600 Basic PN

- Цветной (TFT, 256 цветов) или монохромный (STN, шкалы уровня серого)
- Сенсорный экран 6" с 6 тактильными клавишами
- Ориентация книжная или альбомная
- Размер: 5.7"
- Разрешение: 320 x 240
- 128 переменных
- 50 изображений процесса
- 200 прерываний
- 25 кривых
- 32 KB памяти для рецептов
- 5 рецептов, 20 записей данных, 20 компонентов

KTP1000 Basic PN

- Цветной (TFT, 256 цветов)
- Сенсорный экран 10" с 8 тактильными клавишами
- Размер: 10.4"
- Разрешение: 640 x 480
- 256 переменных
- 50 изображений процесса
- 200 прерываний
- 25 кривых
- 32 KB памяти для рецептов
- 5 рецептов, 20 записей данных, 20 компонентов

TP1500 Basic PN

- Цветной (TFT, 256 цветов)
- Сенсорный экран 15"
- Размер: 15.1"
- Разрешение: 1024 x 768
- 256 переменных
- 50 изображений процесса
- 200 прерываний
- 25 кривых
- 32 KB памяти для рецептов (встроенная флэш-память)
- 5 рецептов, 20 записей данных, 20 компонентов

Монтаж

2

Оборудование S7-1200 спроектировано так, чтобы его можно было легко монтировать. S7-1200 можно монтировать в стандартном пульте управления или на стандартной профильной шине, и вы можете располагать S7-1200 горизонтально или вертикально. Компактные размеры S7-1200 позволяют эффективно использовать пространство.

⚠ ПРЕДУПРЕЖДЕНИЕ

ПЛК SIMATIC S7-1200 являются контроллерами открытого типа. Их необходимо монтировать в корпусе, шкафу или центральном диспетчерском пункте. Доступ к корпусу, шкафу или диспетчерскому пункту должен быть ограничен лицами, имеющими на это право.

Несоблюдение этих требований к монтажу может повлечь смерть, тяжкие телесные повреждения и/или материальный ущерб.

Всегда выполняйте эти требования при монтаже ПЛК S7-1200.

Держите устройства S7-1200 вдали от тепла, высокого напряжения и электрических помех

Общее правило для размещения устройств вашей системы состоит в том, чтобы всегда держать устройства, генерирующие высокое напряжения и большие электрические помехи, вдали от низковольтных электронных приборов, таких, как S7-1200.

При планировании размещения S7-1200 внутри пульта управления обратите внимание на устройства, выделяющие тепло, и размещайте электронные устройства в более прохладных местах своего шкафа. Чем меньше электронное устройство находится в высокотемпературной среде, тем больше срок его службы.

Обратите также внимание на прокладку проводки для ваших устройств в распределительном шкафу. Избегайте размещения низковольтных сигнальных проводов и коммуникационных кабелей в одном кабельном канале с питающими проводами переменного тока и проводами постоянного тока, через которые производятся быстрые переключения.

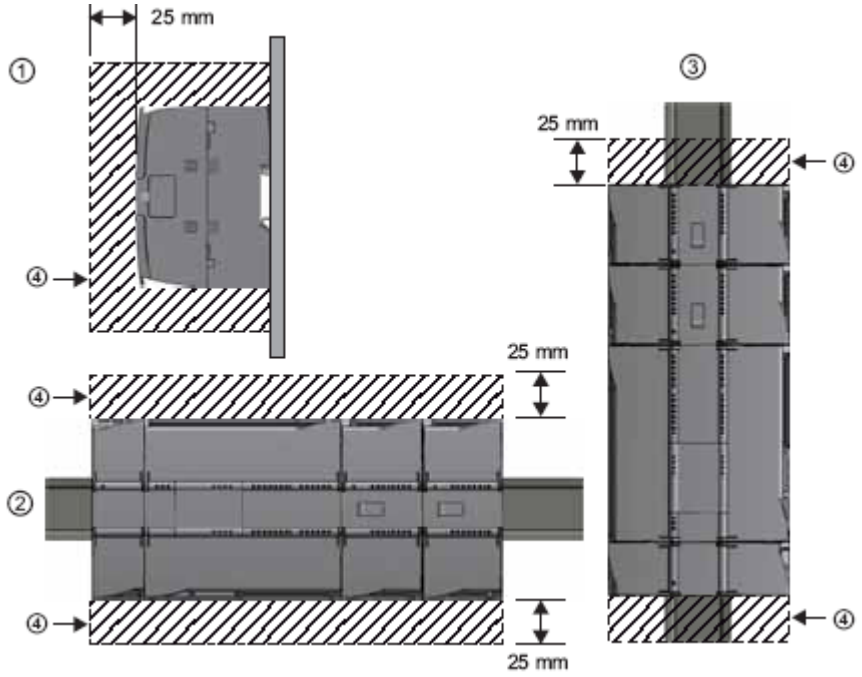
Оставляйте достаточные зазоры для охлаждения и проводки

Устройства S7-1200 разработаны для естественного отвода тепла благодаря конвекции. Для надлежащего охлаждения вы должны оставлять зазор не менее 25 мм сверху и снизу устройств. Обращайте также внимание на то, чтобы между модулями и внутренней стороной корпуса оставался зазор не менее 25 мм.

⚠ ОСТОРОЖНО

Для вертикального монтажа максимально допустимая температура окружающей среды снижается на 10 градусов C. Располагайте вертикально смонтированную систему S7-1200 так, чтобы CPU находился на нижней стороне модуля.

При планировании размещения для системы S7-1200 оставляете достаточный зазор для проводов и подключения коммуникационного кабеля.



① Вид сбоку
② Горизонтальная установка
③ Вертикальная установка
④ Свободное пространство

Баланс мощностей

В CPU имеется внутренний источник питания, который обеспечивает энергией CPU, сигнальные модули, сигнальные платы, коммуникационные модули и другие потребители напряжения 24 В пост. тока.

В технических данных (стр. 329) вы найдете информацию о балансе мощностей для напряжения 5 В пост. тока вашего CPU и потребностях в мощности напряжением 5 В пост. тока сигнальных модулей, сигнальных плат и коммуникационных модулей. С помощью информации в разделе "Расчет баланса мощностей" (стр. 373) вы можете рассчитать, какую мощность (или ток) CPU может поставить для вашей конфигурации.

CPU имеет также блок питания датчиков 24 В пост. тока, который поставляет питание 24 В пост. тока для входов, для катушек реле сигнальных модулей и других потребителей. Если ваши потребности в питании 24 В пост. тока превосходят мощность источника питания датчиков, то вы должны подключить к вашей системе внешний источник питания 24 В пост. тока. В технических данных (стр. 329) вы найдете баланс мощностей для источника питания датчиков 24 В пост. тока для вашего конкретного CPU S7-1200.

Если вам нужен внешний источник питания 24 В пост. тока, то вы должны обратить внимание на то, чтобы этот источник питания не был подключен параллельно с источником питания датчиков вашего CPU. Для достижения наилучшей помехоустойчивости рекомендуется соединить между собой клеммы массы (M) соответствующих источников питания.

⚠ ПРЕДУПРЕЖДЕНИЕ

Подключение внешнего источника питания 24 В пост. тока параллельно источнику питания датчиков 24 В пост. тока может привести к конфликту между двумя источниками, так как каждый из них стремится установить свой собственный уровень выходного напряжения.

Результатом этого конфликта может быть сокращение срока службы или немедленный выход из строя одного или обоих источников питания с последующим непредсказуемым поведением системы ПЛК. Непредсказуемое поведение системы может привести к гибели людей, тяжким телесным повреждениям и/или материальному ущербу.

Блок питания датчиков постоянного тока и внешний источник питания должны поставлять напряжение в различные пункты.

Некоторые входные порты 24 В пост. тока системы S7-1200 соединены друг с другом, причем общий логический провод соединяет между собой несколько клемм M. Например, следующие цепи соединены друг с другом, если в технических данных они обозначены как "не имеющие потенциальной развязки (not isolated)": блок питания 24 В пост. тока в CPU, питающий вход катушки реле сигнального модуля, или блок питания не имеющего потенциальной развязки аналогового входа. Все не имеющие потенциальной развязки клеммы M должны быть подсоединены к одному и тому же внешнему опорному потенциалу.

⚠ ПРЕДУПРЕЖДЕНИЕ

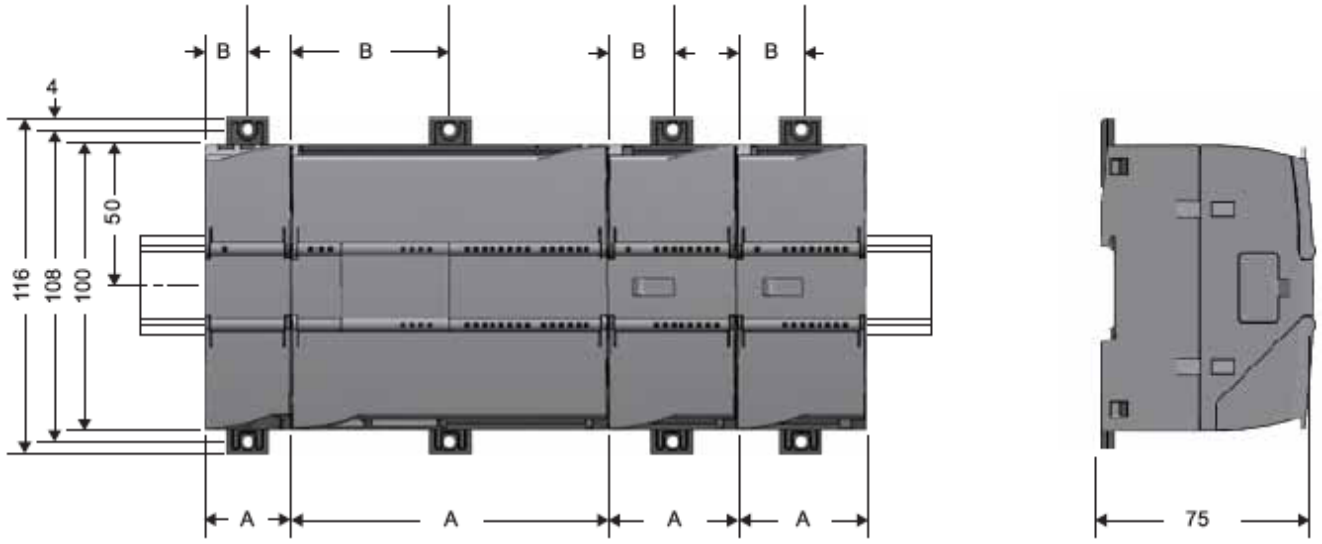
Подключение не имеющих потенциальной развязки клемм M к различным опорным потенциалам вызовет появление непредусмотренных токов, которые могут привести к повреждению или к непредсказуемому поведению ПЛК и подключенного оборудования.

Несоблюдение этих указаний может причинить вред или вызвать непредсказуемое поведение, что может привести к гибели или тяжким телесным повреждениям обслуживающего персонала и/или материальному ущербу.

Всегда подключайте все не имеющие потенциальной развязки клеммы M в системе S7-1200 к одному и тому же опорному потенциалу.

2.1 Процедуры монтажа и демонтажа

Монтажные размеры (мм)



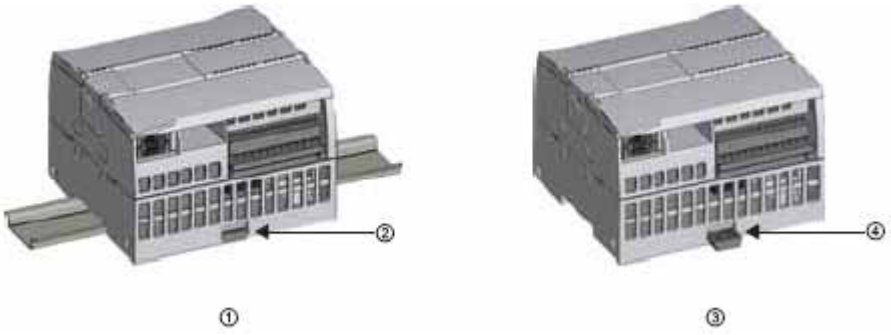
Устройства S7-1200		Ширина A	Ширина B
CPU:	CPU 1211C и CPU 1212C	90 мм	45 мм
	CPU 1214C	110 мм	55 мм
Сигнальные модули:	8 и 16 входов/выходов, пост. тока и релейных (8I, 16I, 8Q, 16Q, 8I/8Q) аналоговые (4AI, 8AI, 4AI/4AQ, 2AQ, 4AQ)	45 мм	22,5 мм
	16I/16Q релейные (16I/16Q)	70 мм	35 мм
Коммуникационные модули:	CM 1241 RS232 и CM 1241 RS485	30 мм	15 мм

CPU, сигнальные и коммуникационные модули пригодны для монтажа на стандартной профильной шине и для встраивания в пульт управления. Для крепления устройства на профильной шине используйте шинные зажимы. Эти зажимы защелкиваются также в извлеченном положении, чтобы сделать возможным монтаж устройства в пульте управления. Внутренние размеры отверстия для шинных зажимов на устройстве составляет 4,3 мм.

Сверху и снизу от устройства в качестве защиты от перегрева должен выдерживаться зазор в 25 мм для свободной циркуляции воздуха.

Установка и удаление устройств S7-1200

CPU легко устанавливается на стандартной профильной шине или в пульте управления. Для крепления устройства на шине с ним поставляются шинные зажимы. Эти зажимы могут быть защелкнуты также и в извлеченном положении, делая возможным крепление устройства винтами в пульте управления.



① Монтаж на профильной шине
② Шинный зажим в запертом положении
③ Монтаж в пульте управления
④ Зажим в извлеченном положении для монтажа в пульте управления

Перед установкой или снятием любого электрического устройства вы должны обеспечить отключение источника питания устройств. Кроме того, обратите внимание, чтобы были выключены и все подключенные устройства.

⚠ ПРЕДУПРЕЖДЕНИЕ

Установка или снятие S7-1200 или относящегося к нему оборудования с включенным питанием может вызвать поражение электрическим током или непредусмотренное поведение оборудования.

Если питающее напряжение S7-1200 и всех подключенных к нему устройств при установке или снятии устройств не выключено, то это может привести из-за поражения электрическим током или непредусмотренного поведения оборудования к гибели людей, тяжким телесным повреждениям и/или материальному ущербу.

Всегда принимайте необходимые меры предосторожности и перед установкой или снятием устройства убедитесь в том, питание CPU S7-1200 выключено.

При установке или замене устройства S7-1200 всегда обращайте внимание на то, чтобы был использован правильный модули или устройство.

⚠ ПРЕДУПРЕЖДЕНИЕ

Установка неправильного модуля S7-1200 может привести к непредсказуемому функционированию программы S7-1200.

Если устройство S7-1200 заменено другой моделью, неправильно ориентировано или смонтировано не в том порядке, то это может привести из-за непредсказуемого поведения устройства к смерти или тяжким телесным повреждениям персонала и/или к материальному ущербу.

Всегда заменяйте устройство S7-1200 то же самой моделью, правильно ориентируйте его и располагайте в правильном месте.

2.1.1 Установка и удаление CPU

Монтаж

Вы можете установить CPU в пульте управления или на профильной шине.

Указание

Прикрепите коммуникационные модули к CPU и монтируйте всю сборку как один узел. Устанавливайте сигнальные модули отдельно, после того, как CPU будет установлен.

Для монтажа CPU в пульте управления действуйте следующим образом:

1. Просверлите монтажные отверстия (M4) в соответствии с указанными монтажными размерами.
2. Вытащите монтажные зажимы из модуля. Убедитесь, что шинные зажимы в верхней и нижней части CPU находятся в извлеченном положении.
3. Закрепите модуль винтами, помещенными в зажимы.

Указание

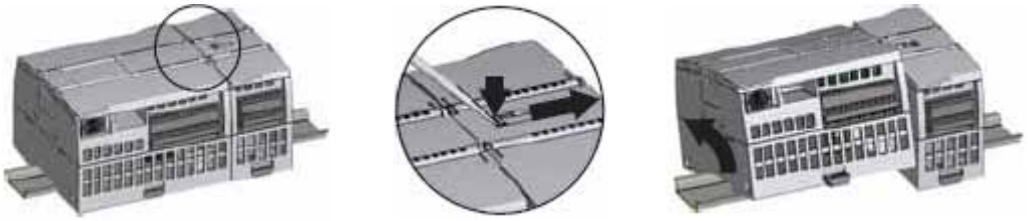
Если ваша система подвергается сильным вибрациям, или она монтируется вертикально, то монтаж S7-1200 в пульте управления обеспечивает лучший уровень защиты.

Для монтажа CPU на стандартной профильной шине действуйте следующим образом:

1. Смонтируйте профильную шину. Прикрепите ее к монтажной панели через каждые 75 мм.
2. Навесьте CPU сверху на профильную шину.
3. Вытащите шинный зажим в нижней части CPU, чтобы CPU мог плотно прилечь к шине.
4. Поверните CPU вниз в монтажное положение на шине.
5. Вдавите зажимы, чтобы закрепить CPU на шине.

Удаление

Для подготовки CPU к удалению выключите питание CPU и отсоедините зажимы ввода/вывода, провода и кабели от CPU. Снимите CPU и прикрепленные к нему коммуникационные модули как одно целое. Все сигнальные модули должны оставаться смонтированными.



Если к CPU подключен сигнальный модуль, извлеките шинный соединитель:

1. Подденьте отверткой планку на верхней стороне сигнального модуля.
2. Нажмите вниз, чтобы освободить из CPU клеммный блок.
3. Сдвиньте планку полностью вправо.

Снимите CPU:

1. Вытащите шинный зажим, чтобы можно было отсоединить CPU от шины.
2. Поверните CPU вверх и в сторону от шины и удалите CPU из системы.

2.1.2 Установка и удаление сигнального модуля

Монтаж

SM монтируется после установки CPU.

Удалите крышку для соединителя с правой стороны CPU.

- Вставьте отвертку в щель над крышкой.
- Осторожно приподнимите крышку вверх, чтобы освободить ее от держателя и снимите ее. Сохраните крышку для последующего использования.

Поместите SM рядом с CPU.

1. Навесьте SM сверху на профильную шину.
2. Вытащите нижний шинный зажим, чтобы SM мог плотно прилечь к шине.
3. Поверните SM вниз, чтобы он оказался рядом с CPU и нажмите нижний зажим, чтобы SM прочно закрепился на шине.

Выдвиньте шинный соединитель.

1. Поместите отвертку рядом с планкой на верхней стороне SM.
2. Сдвиньте планку полностью влево, чтобы сдвинуть шинный соединитель в CPU.

Шинный соединитель создает как механическое, так и электрическое соединение для SM.

Действуйте таким же образом, чтобы присоединить сигнальный модуль к сигнальному модулю.

Удаление

Вы можете удалить SM, не удаляя CPU или другие SM. Для подготовки SM к удалению выключите питание CPU и удалите зажимы ввода/вывода и провода из SM.

Оттяните назад шинный соединитель.

1. Поместите отвертку рядом с планкой на верхней стороне SM.
2. Нажмите вниз, чтобы освободить клеммный блок из CPU.
3. Сдвиньте планку полностью вправо.

Если справа есть еще один SM, повторите эту процедуру для этого SM.

Снимите SM:

1. Вытащите нижний шинный зажим, чтобы отсоединить SM от шины.
2. Поверните SM вверх и в сторону от шины. Удалите SM из системы.
3. Если необходимо, закройте шинный соединитель на CPU крышкой, чтобы предотвратить загрязнение.

Для отделения сигнального модуля от сигнального модуля выполните такую же процедуру.

2.1.3 Установка и удаление коммуникационного модуля

Монтаж

Присоедините CM к CPU и монтируйте эти модули вместе как одно целое на профильной шине или в пульте управления.

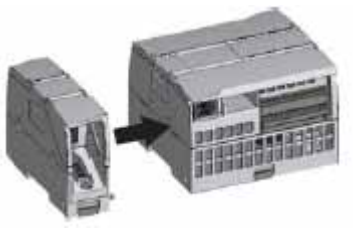
Снимите крышку шины с левой стороны CPU:

1. Вставьте отвертку в щель над крышкой шины.
2. Используя отвертку в качестве рычага, осторожно удалите крышку из держателя.

Снимите крышку шины. Сохраните ее для дальнейшего использования.

Соедините блоки:

1. Совместите шинный соединитель и штифты CM с отверстиями в CPU
2. Крепко прижмите блоки друг к другу, пока штифты не защелкнутся.



Монтаж устройств на профильной шине или в пульте управления.

1. При монтаже на профильной шине обратите внимание на то, чтобы верхний шинный зажим CPU и присоединенных CM находился в положении фиксации (вдвинут), а нижний шинный зажим был выдвинут.
2. Установите CPU и подключенные CM, как это показано в разделе Установка и удаление CPU (стр. 26).
3. После установки устройств на профильной шине переведите нижние шинные зажимы в положение фиксации, чтобы закрепить устройства на профильной шине.

При монтаже в пульте управления убедитесь в том, что шинные зажимы выдвинуты.

Удаление

Удаляйте CPU и CM из профильной шины или пульта управления вместе, как одно целое.

Подготовка к удалению CM.

1. Отключите питание CPU.
2. Удалите зажимы ввода/вывода, все провода и кабели из CPU и CM.
3. При монтаже на профильной шине выдвиньте нижние шинные зажимы на CPU и CM.
4. Удалите CPU и CM с профильной шины или из пульта управления.

Снимите CM.

1. Крепко держите CPU и CM.
2. Отделите их друг от друга.

Не используйте инструменты для разделения модулей, так как это может их повредить.

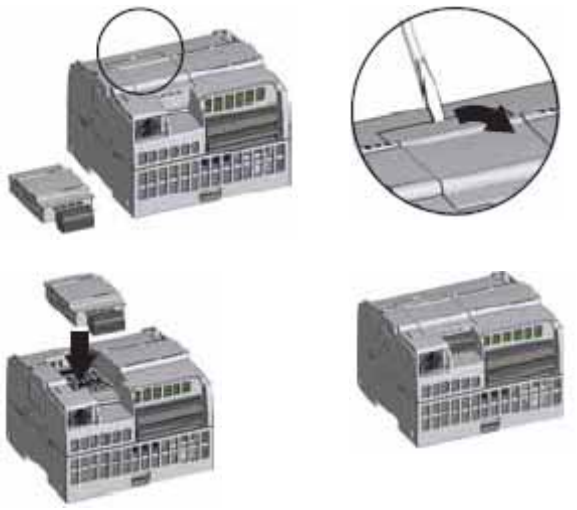
2.1.4 Установка и удаление сигнальной платы

Монтаж

Подготовьте CPU к установке SB, отключив питание CPU и сняв верхнюю и нижнюю крышку клеммных блоков с CPU.

Для установки SB действуйте следующим образом:

1. Введите отвертку в паз сверху на CPU с задней стороны крышки.
2. Используя отвертку как рычаг, осторожно приподнимите крышку и снимите ее с CPU.
3. Вставьте сигнальную плату прямо сверху в ее монтажное положение в верхней части CPU.
4. Крепко нажмите SB, чтобы она защелкнулась.
5. Поставьте на место крышки клеммных блоков.

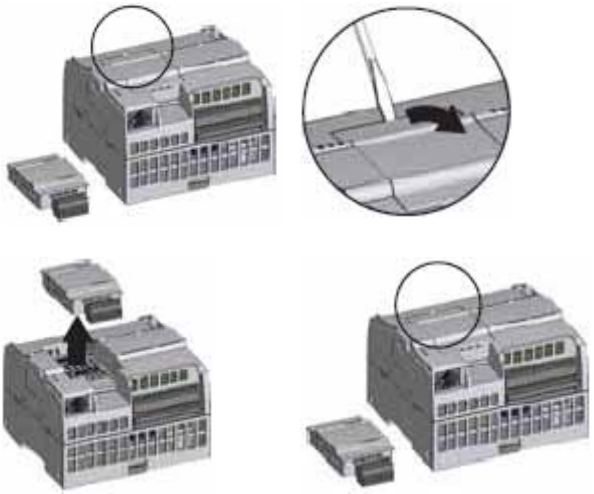


Удаление

Подготовьте CPU к удалению SB, отключив питание CPU и сняв верхнюю и нижнюю крышку клеммных блоков с CPU.

Для удаления SB действуйте следующим образом:

1. Введите отвертку в паз на верхней стороне SB.
2. Действуя отверткой как рычагом, осторожно отсоедините SB от CPU.
3. Вытащите сигнальную плату прямо кверху из ее монтажного положения в CPU.
4. Верните на место крышку SB.
5. Вставьте обратно крышки клеммных блоков.



2.1.5 Удаление и повторная установка клеммного блока S7-1200

Модули CPU, SB и SM снабжены съемными разъемами, облегчающими подключение. Подготовьте систему к удалению клеммного блока:

- Отключите питание CPU.
- Откройте крышку клеммного блока.

Для удаления клеммного блока действуйте следующим образом:

1. Найдите на верхней стороне клеммного блока паз для острия отвертки.
2. Вставьте отвертку в паз.
3. Используя отвертку в качестве рычага, осторожно освободите верхнюю часть клеммного блока из фиксатора в CPU. При освобождении клеммного блока будет слышен щелчок.
4. Захватите клеммный блок рукой и вытащите его из CPU.

Для установки клеммного блока действуйте следующим образом:

1. Подготовьте компоненты к установке клеммного блока, отключив питание CPU и открыв крышку клеммного блока.
2. Выровняйте его в соответствии с положением контактов в устройстве.
3. Выровняйте сторону клеммного блока, к которой подключаются провода, по краю основания клеммного блока.
4. Сильно нажмите клеммный блок, одновременно поворачивая его вниз, пока он не защелкнется.

Тщательно проверьте, правильно ли расположен клеммный блок и надежно ли он закреплен.

2.2 Указания по подключению

Надлежащее заземление и подключение всего электрического оборудования играет важную роль для обеспечения оптимального функционирования вашей системы и повышения помехоустойчивости для вашего приложения и S7-1200. Подробную информацию вы найдете в технических данных (стр. 329) для схем соединений S7-1200.

Предпосылки

Перед заземлением или подключением любого электрического устройства вы должны убедиться, что питание этих устройств отключено. Кроме того, обратите внимание на то, чтобы все подключенные устройства также были выключены.

При подключении S7-1200 и всего относящегося к нему оборудования обеспечьте выполнение всех действующих и обязательных для исполнения стандартов. Монтируйте и эксплуатируйте все оборудование в соответствии с действующими национальными и региональными предписаниями. Обратитесь к местным властям, чтобы выяснить, каким стандартам и предписаниям необходимо следовать в вашем конкретном случае.

⚠ ПРЕДУПРЕЖДЕНИЕ

Монтаж или подключение S7-1200 или соответствующего оборудования во включенном состоянии может привести к получению удара электрическим током или к неожиданному поведению оборудования. Если питание S7-1200 и всего подключенного к нему оборудования во время установки или удаления не выключено, то это может привести к гибели людей, тяжким телесным повреждениям и/или к материальному ущербу из-за удара электрическим током или неожиданного поведения оборудования.

Всегда принимайте необходимые меры предосторожности и удостоверьтесь перед установкой или удалением S7-1200 или подключенного к нему оборудования, что питание S7-1200 выключено.

Всегда принимайте во внимание вопросы безопасности при проектировании заземления и подключение своей системы S7-1200. Электронные устройства управления, например, S7-1200, могут выходить из строя и вызвать непредсказуемое поведение контролируемого и управляемого оборудования. Поэтому вам следует реализовать предохранительные устройства, не зависимые от S7-1200, чтобы защитить персонал от возможных травм, а оборудование от повреждения.

⚠ ПРЕДУПРЕЖДЕНИЕ

Устройства управления могут выходить из строя в небезопасных рабочих состояниях и вызвать из-за этого неконтролируемое поведение управляемого оборудования. Такое непредсказуемое поведение системы автоматизации может привести к гибели людей, тяжким телесным повреждениям и/или материальному ущербу.

Поэтому позаботьтесь о функции аварийного отключения, электромеханических или других устройствах обеспечения безопасности, не зависящих от S7-1200.

Указания для потенциальной развязки

Граничные значения напряжения для источника питания переменного тока S7-1200 и для входов и выходов для цепей переменного тока рассчитаны и допущены к эксплуатации так, чтобы обеспечить надежную электрическую развязку между напряжениями линий переменного тока и низковольтными цепями. В зависимости от стандарта, эти границы требуют для себя двойной или усиленной изоляции или основной плюс дополнительной изоляции. Компоненты, пересекающие эти границы, например, оптические устройства сопряжения, конденсаторы, трансформаторы и реле, допущены к эксплуатации как устройства, обеспечивающие надежную электрическую развязку. Граничные значения для потенциальной развязки, которые удовлетворяют этим требованиям, в спецификациях на продукты S7-1200 указаны как имеющие электрическую развязку для напряжения не менее 1500 В переменного тока. Это значение основано на стандартном заводском испытании (2Ue + 1000 В перем. тока) или эквивалентном в соответствии с допущенными методами. Граничные значения для безопасной электрической развязки S7-1200 прошли типовые испытания при 4242 В пост. тока.

Выход питания датчиков, коммуникационные цепи и электрические цепи внутренней логики S7-1200 со встроенным источником питания переменного тока в соответствии с EN 61131-2 получают питание как цепи безопасного сверхнизкого напряжения (SELV, safety extra-low voltage).

Для поддержания характеристик безопасности цепей низкого напряжения S7-1200 внешние соединения с коммуникационными портами, аналоговыми цепями и всеми источниками питания с номинальным напряжением 24 В, а также с цепями ввода/вывода должны получать питание от сертифицированных в соответствии с различными стандартами источников, удовлетворяющих требованиям SELV, PELV (Protective Extra Low Voltage – защитное сверхнизкое напряжение), класс 2, с ограничением напряжения или мощности.

⚠ ПРЕДУПРЕЖДЕНИЕ

Использование для питания цепей низкого напряжения источников, не имеющих развязки с линией переменного тока или имеющих одинарную изоляцию, может привести к появлению опасных напряжений в цепях, которые считаются безопасными для прикосновения, например, в цепях связи и в проводке датчиков низкого напряжения.

Неожиданно высокие напряжения могут вызвать удар электрическим током, что может привести к смерти, тяжким телесным повреждениям и/или материальному ущербу.

Используйте только такие преобразователи высокого напряжения в низкое, которые сертифицированы как безопасные для прикосновения цепи с ограниченным напряжением.

Указания по заземлению S7-1200

Заземлять ваше приложение лучше всего, подключив все общие клеммы и клеммы заземления вашего S7-1200 и всех подключенных к нему устройств к одной точке. Эта точка должна быть непосредственно соединена с системной землей.

Провода заземления должны быть по возможности короткими и иметь жилы с большим поперечным сечением, например, 2 мм^2.

При выборе точек заземления примите во внимание предписания по технике безопасности и обеспечьте надлежащее функционирование защитных отключающих устройств.

Указания по подключению S7-1200

При проектировании подключения для вашего S7-1200 предусмотрите единое устройство отключения, которое одновременно снимает напряжение с блока питания CPU S7-1200, со всех входных и всех выходных цепей. Предусмотрите максимальную токовую защиту, например, предохранитель или автоматический выключатель, чтобы ограничить аварийный ток в питающей проводке. Подумайте о дополнительной защите с помощью предохранителей или других ограничителей тока в отдельных выходных цепях.

Снабдите линии, которые могут быть подвергнуты ударам молнии, подходящей защитой от перенапряжений.

Избегайте располагать линии сигналов низкого напряжения и кабели связи в одном кабельном канале с проводами питания переменного тока, с проводами, по которым протекает быстро переключающийся постоянный ток. Всегда прокладывайте провода парами: нейтраль или нулевой провод вместе с фазой или проводом, несущим сигнал.

Используйте возможно более короткие провода и обращайте внимание на то, чтобы поперечное сечение провода соответствовало требуемому току. К клеммному блоку можно подключать провода с поперечным сечением от 2 мм^2 до 0,3 мм^2. Для оптимальной защиты от электрических помех используйте экранированные провода. Наилучшие результаты обычно получаются путем заземления экрана на S7-1200.

При подключении входных цепей, которые получают питание от внешнего источника, включайте в эту цепь устройство максимальной токовой защиты. Внешняя защита не требуется для цепей, которые получают питание от источника питания датчиков 24 В пост. тока в S7-1200, так как это источник уже имеет ограничитель тока.

Все модули S7-1200 имеют съемные клеммные блоки для подключения пользователя. Для предотвращения плохо закрепленных соединений обратите внимание НАТО, чтобы клеммный блок был установлен надежно и чтобы провода были надежно вставлены в клеммный блок. Во избежание повреждения клеммного блока не затягивайте винты слишком сильно. Максимальный крутящий момент для винтов клеммного блока составляет 0,56 Нм.

S7-1200 работает в границах, определяемых потенциальной развязкой, что препятствует возникновению нежелательных токов в вашей установке. При планировании подключения для вашей системы примите во внимание эти границы. В разделе Технические данные вы найдете значения для предоставляемой в ваше распоряжение потенциальной развязки и о расположении ее границ. Не полагайтесь на границы потенциальной развязки с номинальными значениями ниже 1500 В перем. тока как на безопасные границы.

Указания для индуктивных нагрузок

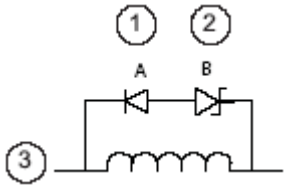
Индуктивные нагрузки следует снабжать защитными цепями для ограничения роста напряжения при выключении выхода контроллера. Защитные цепи защищают ваши выходы от преждевременного выхода из строя из-за высоких напряжений при выключении индуктивных нагрузок. Кроме того, защитные цепи ограничивают электрические помехи, возникающие при выключении индуктивных нагрузок. Наиболее эффективное уменьшение электрических помех достигается подключением внешней защитной цепи параллельно с нагрузкой и физическим размещением ее рядом с нагрузкой.

Указание

Эффективность конкретной защитной цепи зависит от приложения, и вы должны проверить ее для конкретного случая. Все компоненты защитной цепи всегда должны быть рассчитаны для использования в конкретном приложении.

Управление индуктивными нагрузками постоянного тока

Выходы постоянного тока S7-1200 DC включают в себя защитные цепи, подходящие для индуктивных нагрузок в большинстве приложений. Так как релейные выходы могут быть использованы для нагрузок как постоянного, так и переменного тока, то для них внутренняя защита не предусмотрена. На рисунке справа показан пример защитной цепи для нагрузки постоянного тока.
В большинстве приложений достаточно использования одного диода (A) параллельно индуктивной нагрузке, но если ваше приложение быстрого отключения, то рекомендуется использование стабилитрона (B).

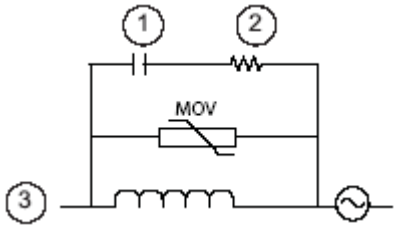


① Диод I1N4001 или эквивалентный
② Стабилитрон 8,2 В (выходы постоянного тока), стабилитрон 36 В (релейные выходы)
③ Выход

Выбирайте стабилитрон в соответствии с током в выходной цепи.

Релейные выходы для управления нагрузками переменного тока

Если вы используете релейные выходы для коммутации нагрузок 115 В/230 В перем. тока, то размещайте резисторные или конденсаторные цепи параллельно с нагрузками переменного тока, как показано на рисунке. Для ограничения пиковых напряжений вы можете использовать также металлооксидный варистор (MOV). Обратите внимание на то, чтобы рабочее напряжение MOV было, по крайней мере, на 20% больше номинального напряжения защищаемой цепи.



① 0,1 мкФ
② от 100 до 120 Ом
③ Выход

Указания для ламповых нагрузок

Ламповые нагрузки повреждают контакты реле из-за больших бросков тока при включении. Этот бросок тока обычно в 10 – 15 выше стационарного тока вольфрамовой лампы. Для часто включаемых в течение срока службы приложения ламповых нагрузок рекомендуется сменное промежуточное реле или ограничитель бросков тока.

3 Основы ПЛК

3.1 Исполнение программы пользователя

CPU поддерживает следующие виды блоков, позволяющие создать эффективную структуру вашей пользовательской программы:

- Организационные блоки (OB) определяют структуру программы. Некоторые OB имеют предопределенное поведение и стартовые события, но вы можете также создавать OB со своими собственными стартовыми событиями.
- Функции (FC) и функциональные блоки (FB) содержат программный код, соответствующий конкретным задачам или комбинациям параметров. Каждая функция и каждый функциональный блок предоставляет в распоряжение набор входных и выходных параметров для совместного использования данных с вызываемым блоком. FB использует также связанный с ним блок данных (называемый экземплярным DB) для сохранения данных о состоянии во время исполнения, которые могут быть использованы другими блоками в программе.
- Блоки данных (DB) хранят данные, которые могут быть использованы программными блоками.

Исполнение программы пользователя начинается одним или несколькими необязательными организационными блоками (OB), которые после перехода в режим RUN обрабатываются один раз, затем следует один или более OB программного цикла, которые обрабатываются циклически. OB может быть также поставлен в соответствие прерывающему событию, которое может быть стандартным событием или событием-ошибкой; затем он исполняется, когда происходит соответствующее событие.

Функция (FC) или функциональный блок (FB) – это блок с кодом программы, который может быть вызван из OB или из другой функции или другого функционального блока. При этом возможны следующие уровни вложения:

- 16 из циклического OB или OB запуска
- 4 из OB прерываний с задержкой, OB циклических прерываний, OB аппаратных прерываний, OB ошибок по времени или OB диагностируемых ошибок

FC не ставятся в соответствие никакому конкретному блоку данных (DB), тогда как FB непосредственно связаны с DB и используют этот DB для передачи параметров и сохранения промежуточных значений и результатов.

Размер пользовательской программы, данных и конфигурации ограничен имеющейся в распоряжении загрузочной памятью и рабочей памятью в CPU. В рамках свободной рабочей памяти число поддерживаемых блоков не ограничено.

Каждый цикл включает в себя запись выходов, чтение входов, исполнение команд программы пользователя и выполнение обслуживания системы или фоновая обработка. Этот цикл называется также циклом сканирования или просто сканированием.

Сигнальная плата, сигнальные и коммуникационные модули обнаруживаются и регистрируются только при запуске.

Указание

Вставка и извлечение сигнальной платы, сигнальных и коммуникационных модулей при включенном устройстве невозможны. Единственным исключением является карта памяти SIMATIC, которая может быть вставлена и извлечена при включенном CPU.

При стандартной конфигурации все цифровые и аналоговые входы и выходы обновляются синхронно с циклом с помощью внутренней области памяти, называемой образом процесса. Образ процесса содержит моментальное отображение физических входов и выходов (физических входов/выходов CPU, сигнальной платы и сигнальных модулей).

CPU выполняет следующие задачи:

- CPU записывает выходы из области выходов образа процесса в физические выходы.
- CPU считывает физические входы непосредственно перед исполнением программы пользователя и сохраняет значения входов в области входов образа процесса. Это гарантирует, что эти значения во время исполнения команд пользователя останутся согласованными.
- CPU выполняет логику команд пользователя и обновляет значения выходов в области выходов образа процесса, вместо того чтобы записывать их в фактические физические выходы.

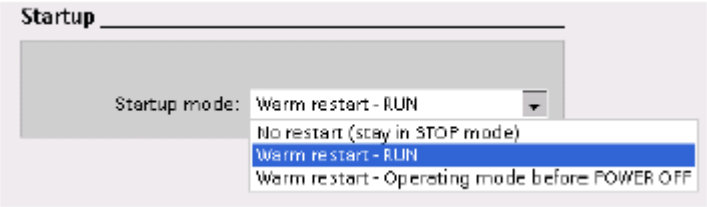
Этот процесс обеспечивает согласованность логики в соответствующем цикле во время исполнения команд пользователя и предотвращает дрожание физических выходов, которое могло бы возникнуть из-за многократного изменения состояния выходов в образе процесса.

Вы можете указать, должны ли сохраняться цифровые и аналоговые входы/выходы в образе процесса. Если вы вставляете модуль в отображение набора устройств, то его данные находятся в образе процесса CPU S7-1200 (по умолчанию). CPU автоматически выполняет обмен данными между модулем и образом процесса во время обновления образа процесса. Чтобы удалить цифровые или аналоговые входы или выходы из автоматического обновления образа процесса, выберите соответствующее устройство конфигурации устройств, обратитесь к вкладке Properties [Свойства], расширьте ее, если необходимо, чтобы желаемые входы и выходы, а затем выберите "IO addresses/HW identifier [Адреса входов/выходов/Идентификатор аппаратуры]". Затем измените запись для образа процесса "Process image:" вместо "Cyclic PI [Циклический образ процесса]" вставьте "---". Чтобы снова ввести входы и выходы в автоматическое обновление образа процесса, снова установите для этого параметра "Cyclic PI".

При исполнении операции вы можете вы можете непосредственно считывать значения физических входов, а также непосредственно записывать значения в физические выходы. При непосредственном считывании происходит обращение к текущему состоянию физического входа. Область входов образа процесса при этом не обновляется, независимо от того, сконфигурирован ли этот вход для сохранения в образе процесса. При непосредственной записи в физический выход обновляется как область выходов образа процесса (если этот выход сконфигурирован для сохранения в образе процесса), так и физический выход. Добавьте окончание ":P" к адресу входа или выхода, если вы хотите, чтобы программа обращалась к данным ввода/вывода прямо через физический вход или выход, а не через образ процесса.

Конфигурирование параметров запуска

С помощью свойств CPU вы можете настроить поведение CPU при запуске после выключения и последующего включения питания.



Выберите состояние (STOP, RUN или последнее перед выключением питания), в которое будет переходить CPU после восстановления питания.

Пояснения к рисунку: Startup – Запуск; Startup mode – Режим запуска; Warm restart – Теплый пуск; No restart (stay in STOP mode) – Нет запуска (оставаться в состоянии STOP); Warm restart –Operating mode before POWER OFF – Теплый пуск – Режим работы перед выключением питания.

CPU выполняет теплый пуск перед переходом в режим RUN. При теплом пуске вся несохраняемая память сбрасывается на начальные значения по умолчанию, но текущие значения в сохраняемой памяти сохраняются.

Указание

CPU после загрузки всегда выполняет новый пуск

Если вы загружаете в CPU какой-нибудь элемент вашего проекта (напр., программный блок, блок данных или аппаратную конфигурацию) CPU перед следующим переходом в режим RUN выполняет новый пуск. Наряду со стиранием входов, инициализацией выходов и инициализацией несохраняемой памяти при новом пуске инициализируются также и области сохраняемой памяти.

После нового пуска, который следует за процессом загрузки, при всех последующих переходах из STOP в RUN выполняется теплый пуск (при этом сохраняемая память не инициализируется).

3.1.1 Режимы работы CPU

CPU имеет три режима работы: режим STOP, режим STARTUP и режим RUN. Светодиоды состояния на передней стороне CPU показывают текущий режим работы.

- В режиме STOP CPU не выполняет программу, и вы можете загружать проект.
- В режиме STARTUP один раз выполняются OB запуска (если имеются). События, связанные с прерываниями, на этапе запуска режима RUN не обрабатываются.
- В режиме RUN многократно выполняется цикл сканирования. События, связанные с прерываниями, могут возникнуть и быть обработаны в любых точках внутри программного цикла.

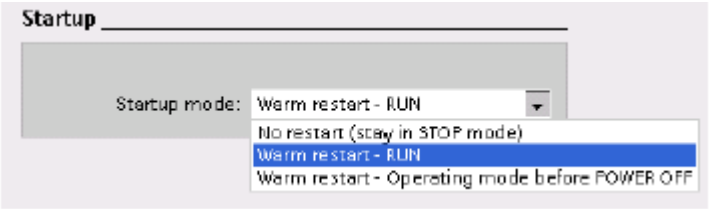
  В режиме RUN нет возможности загрузить проект.

CPU поддерживает теплый пуск для перехода в режим RUN. При теплом пуске не производится полное стирание памяти. При теплом пуске все несохраняемые системные и пользовательские данные инициализируются. Сохраняемые данные сохраняются.

При полном стирании рабочая память, а также все сохраняемые и несохраняемые области памяти стираются, а загрузочная память копируется в рабочую. Полное стирание не очищает диагностический буфер или постоянно хранимые значения IP-адресов.

Вы можете определить поведение CPU при запуске и вид запуска с помощью программного обеспечения. Эти настройки вы найдете в аппаратной конфигурации CPU под названием Startup [Запуск]. При запуске CPU проводит ряд диагностических проверок, а затем инициализацию системы. После этого CPU переключается в соответствующий режим запуска. Определенные ошибки препятствуют тому, чтобы CPU перешел в режим RUN. CPU поддерживает следующие режимы запуска:

- Режим STOP
- Переход в режим RUN после теплого пуска
- Переход в предыдущий режим после теплого пуска

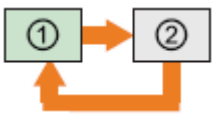


Пояснения к рисунку: Startup – Запуск; Startup mode – Режим запуска; Warm restart – Теплый пуск; No restart (stay in STOP mode) – Нет запуска (оставаться в состоянии STOP); Warm restart –Operating mode before POWER OFF – Теплый пуск – Режим работы перед выключением питания.

Вы можете изменить текущий режим работы с помощью команд "STOP" и "RUN" в онлайновых инструментальных средствах программного обеспечения. Вы можете также включить в свою программу команду STP для перевода CPU в режим STOP. Это позволяет вам прервать выполнение вашей программы в зависимости от ее логики.

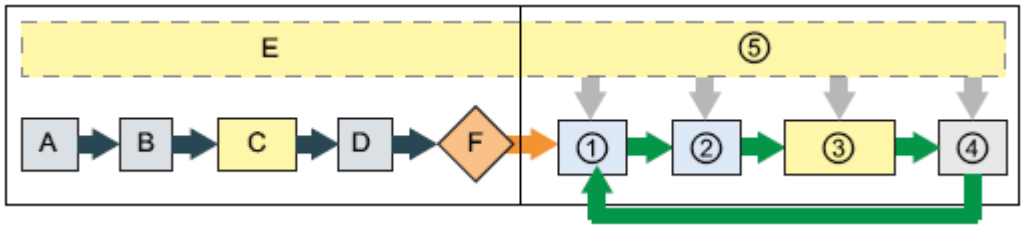
В режиме STOP CPU ① обрабатывает коммуникационные запросы (по обстановке) и ② выполняет самодиагностику.

В режиме STOP CPU не выполняет программу пользователя, и отсутствует автоматическое обновление образа процесса.



Вы можете загрузить свой проект только в том случае, если CPU находится в состоянии STOP.

В режиме RUN CPU выполняет задачи, показанные на следующем рисунке.



STARTUP [Запуск]

- A Очищается область памяти входов (I)
- B Выходы инициализируются последним или заменяющим значением
- C Исполняются OB запуска
- D Состояние физических входов копируется в область памяти входов (I)
- E Все события, связанные с прерываниями, сохраняются в очереди ожидания для обработки в режиме RUN
- F Активизируется запись области памяти выходов (Q) в физические выходы

RUN

- ① Область памяти Q записывается в физические выходы
- ② Состояние физических входов копируется в область памяти I
- ③ Исполняются OB программного цикла
- ④ Выполняется самодиагностика
- ⑤ Во всех частях цикла обрабатываются прерывания и коммуникации

Обработка запуска (STARTUP)

Всякий раз, когда режим работы изменяется со STOP на RUN, CPU стирает область входов образа процесса, инициализирует область выходов образа процесса и обрабатывает OB запуска. Все обращения команд на чтение к области входов образа процесса в OB запуска дают значение ноль, а не текущее значение физического входа. Поэтому, чтобы прочитать текущее состояние физического входа в режиме запуска, вы должны выполнить непосредственное чтение. Затем выполняются OB запуска и относящиеся к ним FC и FB. Если имеется несколько OB запуска, то они выполняются последовательно в соответствии с их номерами, начиная с OB с наименьшим номером.

Каждый OB запуска содержит информацию о запуске, чтобы вы могли выяснить действительность сохраняемых данных и часов реального времени. Вы можете запрограммировать внутри OB запуска команды для проверки этих значений запуска и принятия соответствующих мер. OB запуска поддерживают следующие адреса запуска:

Вход	Тип данных	Описание
LostRetentive	BOOL	Этот бит принимает значение истина, если область хранения сохраняемых данных потеряна
LostRTC	BOOL	Этот бит принимает значение истина, если потеряны часы реального времени

Во время запуска CPU выполняет также следующие задачи.

- В фазе запуска прерывания ставятся в очередь, но не обрабатываются
- В фазе запуска отсутствует контроль времени цикла
- При запуске может быть изменена конфигурация модулей быстрых счетчиков (high-speed counter, HSC), широтно-импульсной модуляции (pulse-width modulation, PWM) и двухточечной связи (point-to-point communication, PtP)
- Фактическое функционирование модулей HSC, PWM и PtP происходит только в режиме RUN

По окончании выполнения OB запуска CPU переходит в режим RUN и обрабатывает задачи управления в непрерывном цикле.

Обработка цикла в режиме RUN

В каждом цикле CPU производит запись в выходы, считывает входы, выполняет программу пользователя, обновляет коммуникационные модули, выполняет внутренние задачи обслуживания и отвечает на события, связанные с прерываниями пользователя, и коммуникационные запросы. Коммуникационные запросы регулярно обрабатываются в течение цикла.

Эти действия (за исключением событий, связанных с прерываниями пользователя) непрерывно обрабатываются циклически. Активизированные события, связанные с прерываниями пользователя, обрабатываются в соответствии с приоритетом в том порядке, в котором они возникают.

Система гарантирует, что цикл будет завершен за интервал времени, называемый максимальным временем цикла; в противном случае генерируется ошибка времени.

- Каждый цикл начинается опросом текущих значений цифровых и аналоговых выходов в образе процесса и записью этих значений в физические выходы CPU, SB и SM, которые сконфигурированы для автоматического обновления входов/выходов (конфигурация по умолчанию). Если команда обращается к физическому выходу, то обновляется как выход в образе процесса, так и сам физический выход.
- Цикл продолжается считыванием текущих значений цифровых и аналоговых входов из CPU, SB и SM, сконфигурированных для автоматического обновления входов/выходов (конфигурация по умолчанию), и последующей записью этих значений в образ процесса. Если команда обращается к физическому входу, то значение физического входа изменяется, но вход в образе процесса не обновляется.
- После считывания входов программа пользователя исполняется от первой до последней команды. Она включает в себя все OB программного цикла плюс все связанные с ними FC и FB. OB программного цикла выполняются постоянно в порядке номеров OB, начиная с OB с наименьшим номером.

Обработка коммуникаций происходит периодически в течение цикла, прерывая, если это возможно, исполнение программы пользователя.

К самодиагностике относятся периодические проверки системы и опрос состояния модулей ввода/вывода.

Прерывания могут возникнуть в любой части цикла, они управляются событиями. Когда происходит событие, CPU прерывает выполнение цикла и вызывает OB, который был спроектирован для обработки этого события. Когда OB заканчивает обработку события, CPU возобновляет исполнение программы пользователя с места, в котором произошло прерывание.

Организационные блоки (OB)

OB управляют исполнением программы пользователя. Каждый OB должен иметь уникальный номер. Некоторые номера ниже 200 зарезервированы для определенных OB. Все остальные OB должны обладать номерами больше 200.

Исполнение организационного блока инициализируется определенными событиями в CPU. OB не могут вызывать друг друга или вызываться из FC или FB. Только стартовое событие, например, диагностическое прерывание или интервал времени, может запустить исполнение OB. CPU обрабатывает OB в соответствии с их классами приоритета, причем в первую очередь обрабатываются OB с боле высоким классом приоритета. Самым низким классом приоритета является 1 (для главного программного цикла), а наивысшим классом приоритета является 27 (для ошибок, связанных с временем).

OB управляют следующими процессами:

- OB программного цикла исполняются циклически, когда CPU находится в режиме RUN. Основной блок программы является OB программного цикла. Он содержит команды для управления вашим приложением, и из него вызываются другие пользовательские блоки. Допустимы несколько OB программного цикла, они выполняются в порядке номеров. OB 1 является стандартным блоком. Другие OB программного цикла должны быть обозначены как OB 200 или выше.
- OB запуска выполняются один раз, когда режим работы CPU меняется из STOP в RUN, при запуске в режим RUN и в случае предписанного перехода из STOP в RUN. Затем начинается исполнение OB программного цикла. Допустимы несколько OB запуска. Стандартным блоком является OB 100. Все остальные OB должны иметь номера, начиная с 200.
- OB прерываний с задержкой исполняются с определенным запаздыванием после события, сконфигурированного в команде запуска прерывания (SRT_DINT). Время задержки указывается во входном параметре расширенной команды SRT_DINT. OB прерываний с задержкой прерывает нормальное исполнение циклической программы, когда истекает указанное время задержки. Вы можете сконфигурировать до 4 событий типа "Задержка времени" в любой момент, причем для каждого такого события допустим только один OB. OB прерываний с задержкой должен иметь номер 200 или выше.
- OB циклических прерываний исполняются через определенные интервалы времени. OB циклических прерываний прерывает исполнение циклической программы через интервалы, определенные пользователем, например, каждые 2 секунды. Вы можете сконфигурировать до 4 событий типа "Циклическое прерывание", причем для каждого такого события допустим только один OB. Этот OB должен иметь номер 200 или выше.
- OB аппаратных прерываний исполняются, когда происходит соответствующее событие в аппаратуре, например, нарастающий или падающий фронт на встроенном цифровом входе или событие, связанное с HSC. OB аппаратных прерываний прерывает нормальное исполнение циклической программы в ответ на сигнал от события в аппаратуре. Эти события определяются в свойствах конфигурации аппаратуры. Для каждого сконфигурированного события в аппаратуре допустим один OB. Этот OB должен иметь номер 200 или выше.
- OB ошибок времени исполняются при обнаружении такой ошибки. OB ошибок времени прерывает нормальное исполнение циклической программы, если превышено максимальное время цикла. Максимальное время цикла определяется в свойствах ПЛК. Для ошибок времени допустим исключительно OB 80. Вы можете определить, что должно произойти, если OB 80 отсутствует: игнорировать ошибку или перейти в STOP.
- OB диагностических прерываний исполняется, когда обнаруживается диагностируемая ошибка, и о ней поступает сообщение. OB диагностических прерываний прерывает нормальное исполнение циклической программы, если модуль, обладающий диагностическими свойствами, распознает ошибку (если диагностическое прерывание активизировано для этого модуля). Для диагностических прерываний допустим только OB 82. Если в программе нет OB 82, то вы можете настроить CPU, чтобы игнорировать ошибку или перейти в STOP.

3.1.2 Приоритеты и очереди для исполнения событий

CPU осуществляет обработку под управлением событий. События запускают исполнение OB прерываний. OB прерываний для события определяется при создании блока, при конфигурировании устройства или командой ATTACH или DETACH. Некоторые события происходят на регулярной основе, например, программный цикл или циклические события. Другие события, например, запуск или событие "Задержка времени" происходят однократно. Некоторые события происходят, когда имеет место изменение, инициированное аппаратурой, например, появление фронта на входе или событие "Скоростной счетчик". Имеются также такие события, как "Диагностируемая ошибка" или "Ошибка времени", которые появляются только в случае ошибки. Приоритеты событий, группы приоритетов и очереди используются для определения порядка обработки OB прерываний.

Событие типа "Программный цикл" возникает один раз в каждом программном цикле (или цикле сканирования). Во время программного цикла CPU осуществляет запись в выходы, считывает входы и выполняет организационные блоки программного цикла. Событие типа "Программный цикл" необходимо и всегда активизируется. Для этого события вы можете не иметь ни одного OB программного цикла или иметь несколько таких OB. После запуска события типа "Программный цикл" исполняется OB программного цикла с наименьшим номером (обычно OB1). Другие OB программного цикла исполняются последовательно в порядке возрастания номеров внутри программного цикла.

События типа "Циклическое прерывание" дают вам возможность организовать исполнение OB прерываний через сконфигурированный интервал времени. Этот интервал времени устанавливается при создании OB и организуется как OB циклических прерываний. Циклические события прерывают программный цикл и исполняют OB циклических прерываний (циклическое событие находится в группе с более высоким приоритетом, чем событие типа "Программный цикл"). Циклическому событию может быть поставлен в соответствие только один OB циклических прерываний. CPU поддерживает четыре события типа "Циклическое прерывание". OB циклических прерываний обладают свойством сдвига по фазе, так что исполнение циклических прерываний с одним и тем же периодом может быть смещено относительно друг друга на величину фазового сдвига.

Событие типа "Запуск" происходит один раз при переходе из STOP в RUN и вызывает на исполнение OB запуска. Для этого события может быть выбрано несколько OB. OB запуска выполняются в порядке возрастания номеров.

События типа "Задержка времени" дают вам возможность организовать исполнение OB прерываний по истечении заданного интервала времени. Время задержки задается командой SRT_DINT. События типа "Задержка времени" прерывают программный цикл для исполнения OB прерываний с задержкой. Событию типа "Задержка времени" может быть поставлен в соответствие только один OB прерываний с задержкой. CPU поддерживает четыре события типа "Задержка времени".

События типа "Аппаратное прерывание" запускаются изменением в аппаратуре, например, нарастающим или падающим фронтом на входе, или событие "Скоростной счетчик" (High Speed Counter, HSC). Для каждого события типа "Аппаратное прерывание" может быть выбран только один OB прерываний. Такие события активизируются в конфигурации устройства. OB для этого события определяются в конфигурации устройства или с помощью команды ATTACH в программе пользователя. CPU поддерживает несколько событий типа "Аппаратное прерывание". Точное количество событий зависит от модели CPU и количества входов.

События типа "Ошибка времени" и "Диагностируемая ошибка" активизируются, когда CPU обнаруживает ошибку. Эти события образуют группу с более высоким приоритетом, чем другие события, вызывающие прерывания, и могут прерывать исполнение событий типа "Задержка времени", циклическое или аппаратное прерывание. Для каждого из событий этого типа может быть задан только один OB прерываний.

Что нужно знать о приоритетах и очередях для исполнения событий

Количество ждущих очереди событий из одного источника ограничено путем назначения каждому типу событий своей очереди. Как только предельное число стоящих в очереди событий достигнуто, следующее событие теряется. Дальнейшую информацию о переполнении очереди вы найдете в разделе "Что нужно знать о событиях типа «Ошибка времени»".

Каждое событие CPU имеет приоритет, и приоритеты событий объединены в классы приоритетов. В следующей таблице представлены длины очередей, классы приоритетов и приоритеты для событий, поддерживаемых CPU.

Указание

Приоритет или класс приоритетов и длина очереди не могут быть изменены.

В общем случае события обрабатываются в соответствии с их приоритетом (в первую очередь наивысший приоритет). События, имеющие одинаковый приоритет, обрабатываются в порядке поступления.

Тип события (OB)	Количество	Допустимые номера OB	Длина очереди	Класс приоритета	Приоритет
Программный цикл	1 событие типа " Программный цикл " Допускается несколько OB	1 (по умолчанию) 200 или больше	1	1	1
Запуск	1 событие типа "Запуск" [1] Допускается несколько OB	100 (по умолчанию) 200 или больше	1		1
Задержка времени	4 события типа "Задержка времени" 1 OB на событие	200 или больше	8	2	3
Циклическое	4 циклических события 1 OB на событие	200 или больше	8		4
Фронты	16 нарастающих фронтов 16 падающих фронтов 1 OB на событие	200 или больше	32		5
HSC	6 событий CV = PV 6 изменений направления счета 6 событий внешнего сброса 1 OB на событие	200 или больше	16		6
Диагностируемая ошибка	1 событие	только 82	8		9
Ошибка времени/ Максимальное время цикла	1 событие типа "Ошибка времени " 1 событие типа "Максимальное время цикла"	только 80	8	3	26
Двойное максимальное время цикла	1 событие типа " Двойное максимальное время цикла "	OB не вызывается	-	3	27

[1] Специальные случаи для события типа "Запуск"

- События "Запуск" и "Программный цикл" никогда не происходят одновременно, так как запуск завершается раньше, чем начинается программный цикл (управляется операционной системой).
- Ни одному из событий не разрешается прерывать запуск. События, которые происходят во время запуска, ставятся в очередь для дальнейшей обработки после завершения запуска.

После запуска OB его обработка не может быть прервана другим событием того же самого или меньшего приоритета. Такие события ставятся в очередь для дальнейшей обработки, давая возможность завершиться текущему OB.

Однако событие из группы с боле высоким приоритетом прерывает текущий OB, и CPU после этого исполняет OB для события с более высоким приоритетом. После обработки этого OB с более высоким классом приоритета CPU исполняет OB для других событий, которые находятся в очереди этого более высокого класса приоритета, а именно в соответствии с приоритетом внутри этого класса. Если в этом классе приоритета больше нет событий, стоящих в очереди, CPU возвращается к боле низкому классу приоритета и возобновляет обработку прерванного OB с того места, где его обработка была остановлена.

Латентный период

Латентный период события (т.е. время между сообщением CPU о возникновении события и началом исполнения первой команды в OB, который обслуживает это событие) составляет примерно 210 мкс, если в момент появления этого события активен в качестве программы обработки только один OB программного цикла.

Что нужно знать о событиях типа "Ошибка времени"

Появление любой ошибки, связанной с временем, приводит к событию типа "Ошибка времени". Поддерживаются следующие ошибки времени:

- Превышение максимального времени цикла
- Запрошенный OB не может быть запущен
- Переполнение очереди

Ошибка "Превышение максимального времени цикла" возникает, если программный цикл не завершается в течение заданного максимального времени цикла. Дальнейшую информацию об ошибке "Превышение максимального времени цикла", об установке максимального времени цикла и о сбросе времени цикла вы найдете в разделе "Контроль времени цикла (стр. 50)".

Ошибка "Запрошенный OB не может быть запущен" возникает, если OB запрашивается циклическим прерыванием или прерыванием с задержкой, но этот OB уже исполняется.

Ошибка "Переполнение очереди" появляется, если прерывания возникают быстрее, чем они могут быть обработаны. Количество событий, находящихся в очереди, ограничено назначением каждому событию собственной очереди ожидания. Если событие происходит, когда соответствующая очередь заполнена, то генерируется событие типа "Ошибка времени".

Все события типа "Ошибка времени" запускают выполнение OB 80, если он существует. Если OB 80 не существует, то CPU игнорирует эту ошибку. Если ошибка "Превышение максимального времени цикла" возникает дважды в одном и том же программном цикле без сброса времени цикла, то CPU переходит в STOP, независимо от того, существует ли OB 80. См. по этому вопросу раздел "Контроль времени цикла" (стр. 50).

OB 80 содержит информацию о запуске, с помощью которой вы можете определить, какое событие и какой OB сгенерировал ошибку времени. Вы можете запрограммировать команды внутри OB 80, чтобы исследовать эти значения и принять необходимые меры. OB 80 поддерживает следующие адреса запуска:

Вход	Тип данных	Описание
fault_id	BYTE	16#01 - превышение максимального времени цикла 16#02 - затребованный OB не может быть запущен 16#07 и 16#09 – произошло переполнение очереди
csg_OBnr	OB_ANY	Количество OB, которые исполнялись, когда произошла ошибка
csg_prio	UINT	Приоритет OB, вызвавшего ошибку

Когда вы создаете новый проект, OB 80 в нем отсутствует. Если необходимо, вставьте OB 80 в свой проект, дважды щелкнув в дереве проекта под "Program blocks [Программные блоки]" на "Add new block [Добавить новый блок]", затем выберите "Organization block [Организационный блок]", а затем "Time error interrupt [Прерывание по ошибке времени]".

Что нужно знать о событиях типа "Диагностируемая ошибка"

Некоторые устройства обладают способностью диагностировать ошибки и сообщать о них. Возникновение или исчезновение различных диагностируемых ошибок приводит к событию типа "Диагностируемая ошибка". Поддерживаются следующие диагностируемые ошибки:

- Отсутствие напряжения у пользователя
- Нарушение верхнего граничного значения
- Нарушение нижнего граничного значения
- Обрыв провода
- Короткое замыкание

Все события типа "Диагностируемая ошибка" вызывают исполнение OB 82, если он существует. Если OB 82 не существует, то CPU игнорирует эту ошибку. При создании нового проекта OB 82 не существует. При желании вы можете добавить OB 82 в свой проект, дважды щелкнув в дереве проекта под "Program blocks [Программные блоки]" на "Add new block [Добавить новый блок]", затем выберите "Organization block [Организационный блок]", а затем "Diagnostic error interrupt [Прерывание по диагностируемой ошибке]".

OB 82 содержит информацию о запуске, с помощью которой вы можете определить, обязано ли это событие появлению или исчезновению ошибки, а также, какое устройство и какой канал сообщили об ошибке. Вы можете запрограммировать команды внутри OB 82, чтобы исследовать эти значения и принять необходимые меры. OB 82 поддерживает следующие адреса запуска:

Вход	Тип данных	Описание
IOstate	WORD	Состояние входов/выходов устройства
laddr	HW_ANY	Идентификатор аппаратуры устройства или функциональной единицы, которая сообщила об ошибке
channel	UINT	Номер канала
multierror	BOOL	TRUE, если имеется несколько ошибок (*в ранних версиях не поддерживается*)

Бит 4 в IO_state было ли событие запущено появление или исчезновением ошибки. Бит 4 равен 1, если ошибка присутствует (пример: обрыв провода) и равен 0, если ошибки больше нет.

Вход в контактном плане содержит аппаратный идентификатор (HW ID) устройства или функциональной единицы, которая выдала ошибку. HW ID назначается автоматически, когда компоненты вставляются в отображение набора устройств или сети и появляется во вкладке Constants [Константы] переменных ПЛК. Идентификатору HW ID также автоматически присваивается имя. Эти записи во вкладке Constants переменных ПЛК не могут быть изменены.

Номер канала начинается с 0 для первого входа (аналогового или цифрового) и с 64 для первого выхода (аналогового или цифрового). Это смещение необходимо для того, чтобы отличить входы от выходов у устройств, имеющих и то, и другое. Если ошибка затрагивает все устройство или функциональную единицу, например, отсутствие напряжения у пользователя, то устанавливается самый старший бит в слове с номером канала (номер канала 32768).

Контроль времени цикла

Время цикла – это время, которое необходимо операционной системе CPU для исполнения циклической фазы режима RUN. CPU предоставляет два способа для контроля времени цикла:

- Максимальное время цикла
- фиксированное минимальное время цикла

Контроль времени цикла начинается после завершения запуска. Запроектировать эту функцию в CPU можно через "Device Configuration > Cycle time [Конфигурация устройств > Время цикла]".

CPU постоянно контролирует время цикла и реагирует, если максимальное время цикла превышено. Если установленное максимальное время цикла превышено, то генерируется ошибка, которая обрабатывается одним из следующих двух способов:

- Если OB 80 отсутствует, то CPU генерирует ошибку и продолжает исполнять программу пользователя
- Если OB 80 присутствует, то CPU исполняет OB 80

Команда RE_TRIGR (перезапустить контроль времени цикла) позволяет сбросить таймер, измеряющий время цикла. Однако эта команда действует только в том случае, если она исполняется в OB программного цикла; команда RE_TRIGR игнорируется, если она исполняется в OB 80. Если максимальное время цикла превышено дважды в одном программном цикле, и при этом команда RE_TRIGR не исполняется между этими двумя превышениями времени, то CPU немедленно переходит в STOP. Повторное исполнение команды RE_TRIGR может создать бесконечный или очень длинный цикл.

Обычно цикл исполняется как можно более быстро, и следующий цикл начинается, как только заканчивается предыдущий. Однако в зависимости от программы пользователя и коммуникационной нагрузки время цикла может варьироваться. Для устранения таких колебаний CPU поддерживает необязательное фиксированное минимальное время цикла (называемого также фиксированным циклом). Если эта функция активизирована, и фиксированное минимальное время цикла задано в мс, то CPU соблюдает это минимальное время цикла с точностью ±1 мс для каждого цикла.

Если CPU выполняет цикл быстрее, чем определено минимальным временем цикла, то CPU использует оставшееся время для диагностики ошибок во время исполнения и/или обработки коммуникационных запросов. Таким образом, CPU всегда занимает для каждого цикла фиксированное время.

Если цикл не завершается в течение заданного минимального времени цикла, то он выполняется нормально до конца (включая обработку коммуникационных запросов), и превышение минимального времени цикла не приводит ни к каким реакциям со стороны системы. В следующей таблице представлены диапазоны и настройки по умолчанию для функций контроля времени цикла.

Время цикла	Диапазон (мс)	Значение по умолчанию
Максимальное время цикла[1]	от 1 до 6000	150 мс
Фиксированное минимальное время цикла [2]	от 1 до максимального времени цикла	деактивировано

[1] Максимальное время цикла всегда активизировано. Установите время цикла от 1 до 6000 мс. Значение по умолчанию составляет 150 мс.

[2] Фиксированное минимальное время цикла необязательно и по умолчанию деактивировано. Если необходимо, установите это время от 1 мс до максимального времени цикла.

Конфигурирование времени цикла и коммуникационной нагрузки

В свойствах CPU в конфигурации устройств вы можете установить следующие параметры:

- Время цикла: Здесь вы можете ввести максимальное время цикла. Вы можете также определить фиксированное минимальное время цикла.

Пояснения к рисунку: Cycle time – Время цикла; Maximum cycle time – Максимальное время цикла; Enabke a fixed minimum cycle time for cyclic OBs – Разблокировать фиксированное минимальное время цикла для циклических OB

- Коммуникационная нагрузка: Вы можете установить долю времени в процентах, предназначенную для выполнения коммуникационных задач.

Пояснения к рисунку: Communication load – Коммуникационная нагрузка; Cycle load due to communication – Загрузка цикла коммуникационными задачами

Подробную информацию о цикле вы найдете в разделе "Контроль времени цикла" (стр. 50).

3.1.3 Память CPU

Управление памятью

CPU предоставляет следующие области памяти для хранения программы пользователя, данных и конфигурации:

- Загрузочная память – это энергонезависимая память для программы пользователя, данных и конфигурации. При загрузке проекта в CPU он сначала сохраняется в загрузочной памяти. Эта память находится или на карте памяти (если она имеется), или в CPU. Эта энергонезависимая память сохраняется также и при отключении питания. Карта памяти поддерживает больший объем памяти, чем память, встроенная в CPU.
- Рабочая память – это энергозависимая память для некоторых элементов проекта пользователя во время исполнения пользовательской программы. CPU копирует некоторые элементы проекта из загрузочной памяти в рабочую. Эта энергозависимая область памяти теряется при отключении питания, а при возвращении питания CPU ее восстанавливает.
- Сохраняемая память – это энергонезависимая память для ограниченного количества значений рабочей памяти. Область сохраняемой памяти служит для сохранения выбранных адресов памяти пользователя при потере питания. При исчезновении питания у CPU имеется достаточно времени для сохранения значений ограниченного числа адресов памяти. При включении питания эти сохраняемые значения восстанавливаются.

Чтобы отобразить использование памяти для текущего проекта, щелкните правой клавишей мыши на CPU (или одном из его блоков) и выберите "Resources [Ресурсы]" из контекстного меню. Для отображения использования памяти для текущего CPU дважды щелкните на "Online and diagnostics [Онлайновый режим и диагностика]", разверните "Diagnostics" и выберите "Memory [Память]".

Сохраняемая память

Можно избежать потери данных при выходе из строя питания, пометив некоторые данные как сохраняемые. В качестве сохраняемых могут быть сконфигурированы следующие данные:

- Битовая память (меркеры) (M): Вы можете определить точную ширину этой памяти для меркеров в таблице переменных ПЛК или в списке назначений. Сохраняемая битовая память всегда начинается с MB0 и непрерывно продолжается через указанное число байтов. Задайте это значение в таблице переменных ПЛК или в списке назначений, щелкнув на символе "Retain [Сохранять]" в линейке инструментов. Введите количество байтов битовой памяти, подлежащих сохранению, начиная с MB0.
- Переменные функционального блока (FB): Если FB был создан с активизированной опцией "Symbolic access only [Только символическая адресация]", то редактор интерфейса для этого FB содержит столбец "Retain [Сохранять]". В этом столбце вы можете выбрать "Retain [Сохранять]" или "Non-Retain [Не сохранять]" индивидуально для каждой переменной. Экземплярный DB, который был создан при вставке FB в редактор программ, также отображает этот столбец, но только для просмотра; вы не можете изменить состояние сохраняемости в редакторе интерфейса экземплярного DB для FB, который был создан с опцией "Symbolic access only".

  Если FB был создан с деактивированной опцией "Symbolic access only [Только символическая адресация]", то редактор интерфейса для этого FB не содержит столбца "Retain [Сохранять]". Экземплярный DB, который был создан при вставке FB в редактор программ, отображает столбец "Retain [Сохранять]", который доступен для редактирования. В этом случае выбор опции "Retain" для любой переменной приводит к выбору **всех** переменных. Аналогично, отмена выбора этой опции для любой переменной приводит к отмене этого выбора для **всех** переменных. У FB, для которого при его создании **не** была активизирована опция "Symbolic access only [Только символическая адресация]", вы можете изменять состояние сохраняемости в редакторе экземплярного DB, но при этом все переменные устанавливаются одновременно в одно и то же состояние.

  После создания FB вы больше не можете изменять опцию "Только символическая адресация ". Эта опция может быть активизирована только при создании FB. Чтобы определить, был ли существующий FB создан только для символической адресации, щелкните правой клавишей мыши на FB в дереве проекта, выберите "Properties [Свойства]", а затем выберите "Attributes [Атрибуты]".
- Переменные глобального блока данных: Поведение глобального DB относительно назначения сохраняемого состояния подобно поведению FB. В зависимости от настройки символической адресации вы можете определять состояние сохраняемости или для отдельных, или для всех переменных глобального блока данных.
  - Если атрибут "Symbolic access only [Только символическая адресация]" этого DB активизирован, то состояние сохраняемости может быть установлено для каждой переменной отдельно.
  - Если атрибут "Symbolic access only" этого DB не активизирован, то настройка сохраняемости действительна для всех переменных DB; или все переменные являются сохраняемыми, или несохраняемыми.

В целом 2048 байт данных могут быть сохраняемыми. Чтобы выяснить, сколько байт имеется в распоряжении, щелкните в таблице переменных ПЛК или в списке назначений в панели инструментов на пиктограмме "Retain [Сохранять]". Хотя здесь вы указываете сохраняемую область для битовой памяти, но вторая строка показывает остающуюся общую память для M и DB вместе.

Диагностический буфер

CPU поддерживает диагностический буфер, который содержит по одной записи для каждого диагностического события. Каждая запись содержит дату и время, в которое произошло событие, категорию события и его описание. Записи отображаются в хронологическом порядке, причем самое последнее событие находится на самом верху. Пока CPU включен, в этом буфере хранится до 50 самых последних событий. Когда буфер заполняется, новое событие заменяет в нем самое старое. При отключении питания сохраняются последние 10 событий.

В диагностический буфер записываются следующие типы событий:

- Каждое диагностическое событие в системе; например, ошибки CPU и модулей
- Каждое изменение состояния CPU (каждый запуск, переход в STOP или в RUN)

Для доступа к диагностическому буферу вы должны находиться в режиме онлайн. Этот буфер вы найдете через "Online & diagnostics / Diagnostics / Diagnostics buffer [Онлайновый режим и диагностика / Диагностика /Диагностический буфер]". Дальнейшую информацию по поиску и устранению ошибок вы найдете в разделе "Инструментальные средства онлайнового режима и диагностики ".

Часы реального времени

CPU снабжен часами реального времени. Когда CPU выключается, эти часы получают питание от мощного конденсатора. Этот конденсатор заряжается, когда CPU включен. Если CPU был включен не менее 2 часов, то заряда конденсатора обычно хватает для работы часов в течение 10 дней.

Часы реального времени установлены на системное время, которое представляет собой координированное мировое время (Coordinated Universal Time, UTC). Системное время для часов реального времени устанавливает STEP 7 Basic. В нем имеются команды для считывания системного (RD_SYS_T) или местного (RD_LOC_T) времени. Для расчета местного времени используются часовой пояс и моменты переключения между летним и зимним временем, которые вы вводите для часов CPU в конфигурации устройств.

Часы реального времени CPU конфигурируются через свойство "Time of day [Значение времени]". Здесь вы можете также ввести переход на летнее время, указав его начало и конец. Для установки часов реального времени вы должны находиться в режиме онлайн и вызвать отображение "Online & diagnostics [Режим онлайн и диагностика]" в CPU. Используйте для этого функцию "Set time of day [Установить значение времени]".

Системная память и тактовые меркеры

В свойствах CPU вы можете активизировать байты для "системной памяти" и "тактовых меркеров (тактовых битов памяти)". В логике своей программы вы можете ссылаться на отдельные биты этих функций.

- Вы можете назначить один байт в битовой (M) памяти в качестве системной памяти. Байт системной памяти предоставляет в распоряжение следующие четыре бита, на которые вы можете ссылаться в своей пользовательской программе:
  - Бит "Always 0 (low) [Всегда 0 (сброшен)]" всегда установлен на 0.
  - Бит "Always 1 (high) [Всегда 1 (установлен)]" всегда установлен на 1.
  - Бит "Diagnostic graph changed [Диагностическая диаграмма изменена]" устанавливается в 1 на время одного цикла сканирования, после того как CPU регистрирует диагностическое событие. Так как CPU не устанавливает этот бит до конца первого исполнения OB программного цикла, то ваша пользовательская программа не может распознать, происходило ли изменение в диагностике во время исполнения OB запуска или во время первого исполнения OB программного цикла.
  - Бит "First scan [Первый цикл]" устанавливается в 1 на время первого цикла сканирования после завершения OB запуска. (После исполнения первого цикла этот бит устанавливается в 0.)
- Вы можете назначить один байт в битовой (M) памяти в качестве тактовых меркеров. Каждый бит этого байта, сконфигурированного в качестве тактового меркера, генерирует прямоугольный импульс. Байт тактовых меркеров предоставляет 8 различных частот, от 0.5 Гц (медленно) до 10 Гц (быстро). Вы можете использовать эти биты в качестве управляющих битов, особенно в соединении с командами обработки фронтов, для циклического запуска действий в программе пользователя.

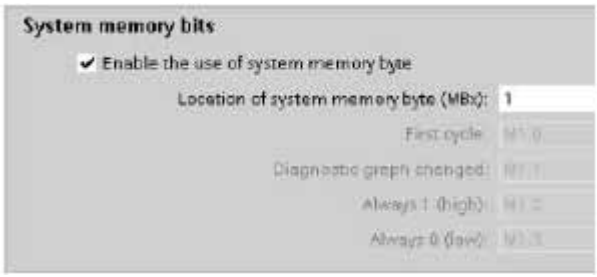
CPU инициализирует эти байты при переходе из состояния STOP в режим STARTUP (запуск). В режимах STARTUP и RUN тактовые биты памяти меняются синхронно с тактом CPU.

ОСТОРОЖНО

Наложение другой записи на биты системной памяти или тактовые биты может повредить данные в этих функциях и вызвать неправильную работу вашей программы, что может привести к повреждению оборудования и телесным повреждениям персонала.

Так как оба эти вида памяти не являются зарезервированной памятью в области битовой (M) памяти, то команды и обмен данными могут вести запись в эти адреса и повредить имеющиеся там данные.

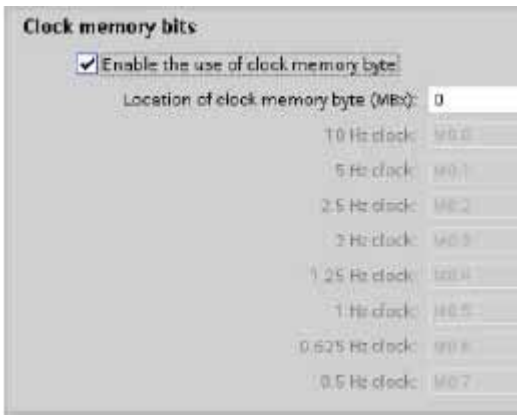
Избегайте осуществлять запись в эти адреса, чтобы обеспечить надлежащее исполнение этих функций, и всегда реализуйте схему аварийного отключения для вашего процесса или машины.



Пояснения к рисунку: System memory bits – Биты системной памяти; Enable the use of system memory byte – Разблокировать использование байта системной памяти; Location of system memory byte – Адрес байта системной памяти.

Системная память конфигурирует один байт, который включается при следующих условиях.

- Первый цикл (First cycle): Включается во время первого цикла в режиме RUN
- Диагностическая диаграмма изменена (Diagnostic graph changed)
- Всегда 1 (установлен) (Always 1 (high)): всегда включен
- Всегда 0 (сброшен) (Always 0 (low)): всегда выключен



Пояснения к рисунку: Clock memory bits – Тактовые биты памяти; Enable the use of clock memory byte - Разблокировать использование байта тактовых битов памяти; Location of clock memory byte – Адрес байта тактовых битов; 10 Hz clock – Тактовые импульсы с частотой 10 Гц

Тактовые биты памяти (тактовые меркеры) образуют байт, в котором отдельные биты включаются и выключаются через определенные промежутки времени.

Тактовые биты памяти генерируют прямоугольные импульсы. Эти биты могут использоваться как управляющие биты, особенно в соединении с командами обработки фронтов, для циклического запуска действий в программе пользователя.

Конфигурирование поведения выходных значений для состояния STOP CPU

Вы можете сконфигурировать поведение цифровых и аналоговых выходов, когда CPU находится в состоянии STOP. Для каждого выхода CPU, SB или SM вы можете заморозить его значение или использовать заменяющее значение:

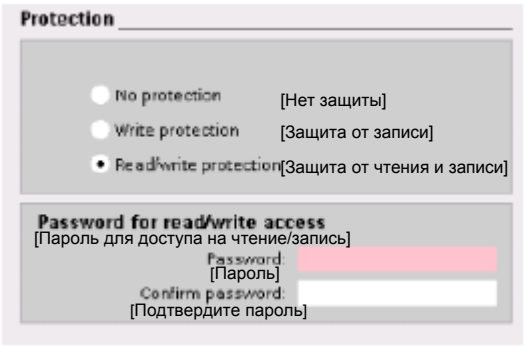
- Замена заданным выходным значением (по умолчанию): Вы вводите заменяющее значение для каждого выхода (канала) CPU, SB или SM.
  Для цифровых выходов заменяющим значением по умолчанию является ВЫКЛЮЧЕНО, а для аналоговых выходов оно равно 0.
- Замораживание выходов в последнем состоянии: Выходы сохраняют свои текущие значения на момент перехода из RUN в STOP. После запуска выходы устанавливаются на заменяющее значение по умолчанию.

Конфигурирование поведения выходов осуществляется в конфигурации устройств. Выбирайте отдельные устройства и открывайте вкладку "Properties [Свойства]", чтобы сконфигурировать выходы соответствующего устройства.

При переходе из RUN в STOP CPU сохраняет образ процесса и записывает соответствующие значения для цифровых и аналоговых выходов в соответствии с конфигурацией.

3.1.4 Защита паролем для CPU S7-1200

CPU предоставляет 3 уровня защиты для ограничения доступа к определенным функциям. Устанавливая уровень защиты и пароль для CPU, вы ограничиваете функции и области памяти, к которым можно обратиться без ввода пароля.



Для конфигурирования пароля действуйте следующим образом:

1. В разделе "Device configuration [Конфигурация устройств]" выберите CPU.
2. В окне просмотра параметров выберите вкладку "Properties [Свойства]".
3. Выберите свойство "Protection [Защита]", чтобы указать уровень защиты и ввести пароль.

Пароль чувствителен к регистру букв.

Каждый уровень защите разрешает неограниченный доступ к определенным функциям без ввода пароля. По умолчанию CPU не имеет ограничений и защиты паролем. Чтобы ограничить доступ к CPU, вы должны сконфигурировать свойства CPU и ввести пароль.

Ввод пароля через сеть не подрывает парольную защиту CPU. К CPU, защищенному паролем, в каждый момент времени имеет неограниченный доступ только один пользователь. Защита паролем не действительна для исполнения команд программы пользователя, включая коммуникационные функции. Ввод правильного пароля разрешает беспрепятственный доступ ко всем функциям.

Обмен данными между ПЛК (через коммуникационные функции в кодовых блоках) не ограничивается уровнями защиты CPU. Функции человеко-машинного интерфейса также остаются неограниченными.

Уровень защиты	Ограничения доступа
Нет защиты	Беспрепятственный доступ без защиты паролем.
Защита от записи	Доступ к устройствам человеко-машинного интерфейса и беспрепятственный обмен данными между ПЛК без защиты паролем. Пароль необходим для изменений (доступ на запись) в CPU и для изменения режима работы CPU (RUN/STOP).
Защита от чтения и записи	Доступ к устройствам человеко-машинного интерфейса и беспрепятственный обмен данными между ПЛК без защиты паролем. Пароль необходим для чтения данных в CPU, для изменений (доступ на запись) в CPU и для изменения режима работы CPU (RUN/STOP).

3.1.5 Восстановление утерянного пароля

Если вы утеряли пароль для CPU, защищенного паролем, сотрите защищенную паролем программу с помощью пустой передаточной карты. Пустая передаточная карта стирает внутреннюю загрузочную память CPU. Затем вы можете загрузить в CPU новую пользовательскую программу из STEP 7 Basic.

Информацию о создании и использовании пустой передаточной карты вы найдете в разделе Передаточная карта (стр. 70).

⚠ ПРЕДУПРЕЖДЕНИЕ

Если вставить передаточную карту в работающий CPU, то CPU перейдет в состояние STOP. Устройства управления могут выйти из строя в небезопасных условиях и вызвать вследствие этого непредсказуемое поведение управляемого оборудования. Такое непредсказуемое поведение может привести к гибели или к тяжким телесным повреждениям работающего персонала и/или материальному ущербу.

Вы должны удалить передаточную карту перед переводом CPU в режим RUN.

3.2 Память данных, области памяти и адресация

CPU предоставляет несколько возможностей для сохранения данных во время исполнения программы пользователя:

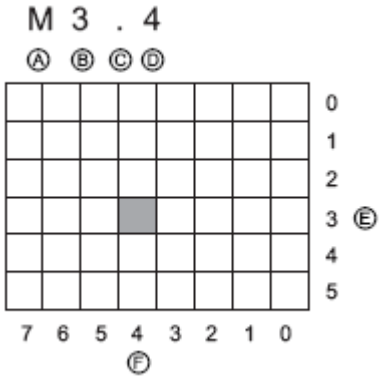
- Глобальная память: CPU предоставляет ряд специализированных областей памяти, включая входы (I), выходы (Q) и битовую память (меркеры) (M). Эта память доступна для всех кодовых блоков без ограничения
- Блок данных (DB): Вы можете включить DB в свою пользовательскую программу для сохранения данных для кодовых блоков. Эти данные сохраняются после исполнения соответствующего кодового блока. В "глобальном" DB сохраняются данные, которые могут быть использованы всеми кодовыми блоками, тогда как в экземплярном DB хранятся данные только для конкретного FB, и они структурированы в соответствии с параметрами этого FB.
- Временная память: При вызове кодового блока операционная система CPU выделяет временную, или локальную, память (L) для использования во время исполнения этого блока. Когда исполнение кодового блока заканчивается, CPU выделяет эту локальную память для исполнения другого блока

Каждое место в памяти имеет уникальный адрес. С помощью этого адреса ваша пользовательская программа может обращаться к данным, находящимся на этом месте в памяти.

Область памяти	Описание	Принудительное присваивание значений	Сохраняемая
I Образ процесса на входах I_:P (Физический вход)	В начале цикла копируется из физических входов	Нет	Нет
	Непосредственное чтение физических входов CPU, SB или SM	Да	Нет
Q Образ процесса на выходах Q_:P (Физический выход)	В начале цикла копируется в физические выходы	Нет	Нет
	Непосредственная запись в физические выходы CPU, SB, и SM	Да	Нет

Область памяти	Описание	Принудительное присваивание значений	Сохраняемая
M Битовая память	Управление и память данных	Нет	Да
L Временная память	Временные, локальные данные для блока	Нет	Нет
DB Блок данных	Память данных, а также память параметров для FB	Нет	Да

Каждое место в памяти имеет уникальный адрес. С помощью этого адреса ваша пользовательская программа может обращаться к данным, находящимся на этом месте в памяти. На следующем рисунке показан пример обращения к биту (адресация в формате "байт.бит"). В этом примере за обозначением области памяти и адресом байта (I = вход, и 3 = байт 3) следует точка ("."), отделяющая адрес бита (бит 4).



A Идентификатор области памяти
B Адрес байта: байт 3
C Разделитель ("байт.бит")
D Адрес бита в байте (бит 4 из 8)
E Байты области памяти
F Биты выбранного байта

Используя для адресации формат байт.бит, вы можете обращаться к данным в большинстве областей памяти (I, Q, M, DB и L) побайтно, пословно или используя двойные слова. Чтобы обратиться к байту, слову или двойному слову данных в памяти, вы должны указать его адрес подобно адресу бита. Вы указываете идентификатор области, размер данных и начальный байтовый адрес байта, слова или двойного слова. Обозначениями размера данных являются B (байт), W (слово) и D (двойное слово), например, IB0, MW20 или QD8. Такие адреса, как I0.3 и Q1.7 относятся к образу процесса. Для обращения к физическому входу или выходу добавьте к адресу символы ":P" (например, I0.3:P, Q1.7:P или "Stop:P").

Доступ к данным в областях памяти CPU

STEP 7 Basic облегчает символическое программирование. Обычно переменные создаются или в переменных ПЛК, или в блоке данных, или в интерфейсе в верхней части OB, FC или FB. Эти переменные включают в себя имя, тип данных, смещение и комментарий. Кроме того, в блоке данных может быть указано начальное значение. Вы можете использовать эти переменные при программировании, вводя имя переменной в качестве параметра для команды. При желании вы можете ввести в качестве параметра для команды абсолютный операнд (область памяти, размер и смещение). Примеры в следующих разделах показывают, как вводить абсолютные операнды. Перед абсолютным операндом программным редактором автоматически вводится символ %. В программном редакторе у вас есть возможность выбора между следующими представлениями: символическое, символическое и абсолютное или абсолютное.

I (образ процесса на входах): CPU опрашивает периферические (физические) входы в каждом цикле непосредственно перед исполнением циклического OB и записывает эти значения в образ процесса на входах. Вы можете обращаться к образу процесса на входах побитно, побайтно, пословно или используя двойные слова. Разрешается доступ как на чтение, так и на запись, но обычно входы образа процесса только считываются.

Бит	I[адрес байта].[адрес бита]	I0.1
Байт, слово или двойное слово	I[размер][адрес начального байта]	IB4, IW5 или ID12

Добавляя к адресу ":P", вы можете непосредственно считывать цифровые и аналоговые входы CPU, SB или SM. Доступ через I_:P отличается от доступа через I тем, что данные получаются непосредственно с входов, к которым производится обращение, а не из образа процесса на входах. Доступ через I_:P называется также прямым доступом на чтение,
Так как данные считываются прямо из источника, а не из его копии, которая была сделана при последнем обновлении образа процесса на входах.

Так как физические входы получают свои значения непосредственно из подключенных к ним полевых устройств, то запись в эти входы запрещена. То есть доступ через I_:P является доступом только на чтение, в отличие от доступа к I, который возможен как на считывание, так и на запись.

Доступ через I_:P ограничен также размером входов, поддерживаемых CPU, SB или SM, с округлением до следующего байта. Например, если входы SB с 2 DI / 2 DQ сконфигурированы так, что они начинаются с I4.0, то обратиться к этим входа можно с помощью I4.0:P и I4.1:P или IB4:P. Обращение к I4.2:P … I4.7:P не распознается как ошибка, но не имеет смысла, так как эти адреса не используются. Попытки обращения к IW4:P и ID4:P запрещены, так как они превышают байтовое смещение этой SB.

Обращение через I_:P не влияет на соответствующее значение, хранящееся в образе процесса на входах.

Бит	I[адрес байта].[адрес бита]:P	I0.1:P
Байт, слово или двойное слово	I[размер][адрес начального байта]:P	IB4:P, IW5:P, или ID12:P

Q (образ процесса на выходах): CPU копирует значения, хранящиеся в образе процесса на выходах в физические выходы. К образу процесса на выходах вы можете обращаться побитно, побайтно, пословно или используя двойные слова. К выходам образа процесса разрешается доступ как на чтение, так и на запись.

Бит	Q[адрес байта].[адрес бита]	Q1.1
Байт, слово или двойное слово	Q[размер][адрес начального байта]	QB5, QW10, QD40

Добавляя к адресу ":P", вы можете осуществлять непосредственную запись в физические цифровые и аналоговые выходы CPU, SB или SM. Доступ через Q_:P отличается от доступа через Q тем, что данные поступают непосредственно на выходы, к которым осуществляется обращение, и, кроме того, в образ процесса на выходах (запись осуществляется в оба места). Доступ через Q_:P иногда называют прямым доступом, так как данные посылаются прямо на целевой адрес, которому не приходится ждать следующего обновления образа процесса на выходах.

Так как физические выходы непосредственно управляют полевыми устройствами, подключенными к этим выходам, то чтение с этих выходов запрещено. Т.е. доступ через Q_:P является доступом только на запись, в отличие от доступа через Q, при котором возможно как чтение, так и запись.

Доступ через Q_:P ограничен также размером выходов, поддерживаемых CPU, SB, или SM (с округлением до следующего байта). Например, если выходы SB с 2 DI / 2 DQ сконфигурированы так, что они начинаются с Q4.0, то к этим выходам можно обращаться через Q4.0:P и Q4.1:P или через QB4:P. Обращение к Q4.2:P … Q4.7:P не воспринимается как ошибка, но не имеет смысла, так как эти адреса не используются. Попытки обращения к QW4:P и QD4:P запрещены, так как они превышают байтовое смещение этой SB.

Доступ через Q_:P влияет как на физический выход, так и на соответствующее значение, в образе процесса на выходах.

Бит	Q[адрес байта].[адрес бита]:P	Q1.1:P
Байт, слово или двойное слово	Q[размер][адрес начального байта]:P	QB5:P, QW10:P или QD40:P

M (область битовой памяти, M-память): Эту область памяти вы можете использовать для управляющих реле и данных, чтобы хранить промежуточные результаты операций или другую управляющую информацию. К области битовой памяти можно обращаться побитно, побайтно, пословно или используя двойные слова. Для битовой памяти возможен доступ как на чтение, так и на запись.

Бит	M[адрес байта].[адрес бита]	M26.7
Байт, слово или двойное слово	M[размер][адрес начального байта]	MB20, MW30, MD50

Temp (временная память): CPU выделяет временную память по мере необходимости. CPU выделяет временную память кодовому блоку в момент его запуска (для OB) или вызова (для FC или FB). При выделении временной памяти кодовому блоку могут повторно использоваться те же адреса временной памяти, которые перед этим были использованы другим OB, FC или FB. CPU не инициализирует временную память в момент выделения, поэтому она может содержать любые значения.

Временная память подобна M-памяти за одним важным исключением: область действия M-памяти "глобальна", а область действия временной памяти "локальна":

- M-память: Любой OB, FB и любая FC может обратиться к данным в M-памяти, т.е. данные находятся глобально в распоряжении всех элементов программы пользователя.
- Временная память: доступ к данным во временной памяти ограничен тем OB, FB или той FC, где были созданы или объявлены адреса во временной памяти. Адреса временной памяти остаются локальными и не могут быть использованы другими кодовыми блоками, даже если кодовый блок вызывает другой кодовый блок. Например: Если OB вызывает FC, то FC не может обратиться к временной памяти OB, вызвавшего эту функцию.

CPU предоставляет временную (локальную) память для каждого из трех классов приоритета OB:

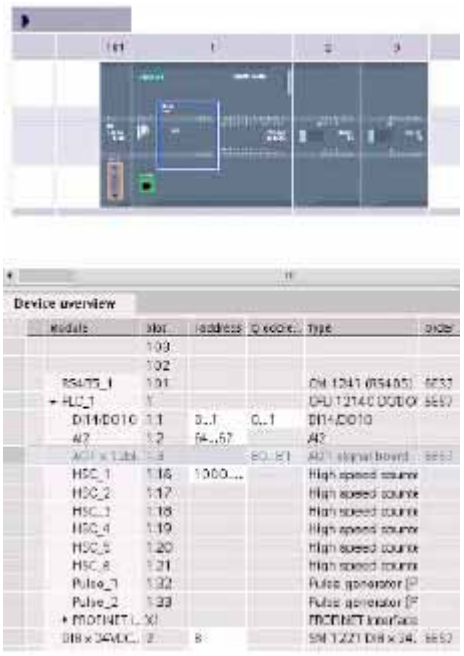
- 16 Кбайт для запуска и программного цикла, включая соответствующие FB и FC
- 4 Кбайта для стандартных событий, вызывающих прерывания, включая FB и FC
- 4 Кбайта для событий, вызывающих прерывания в связи с ошибками, включая FB и FC

к временной памяти можно обращаться только с использованием символической адресации.

DB (блок данных): Используйте блоки данных для хранения различных типов данных, включая промежуточные результаты операций или другие управляющие параметры для FB, и структуры данных, необходимые для многих команд, например, таймеров и счетчиков. Вы можете определить для блока данных доступ на чтение и запись или только на чтение. К блокам данных можно обращаться побитно, побайтно, пословно или используя двойные слова. Доступ к блокам данных, не защищенным от записи, возможен как на чтение, так и на запись. Доступ к блокам данных, защищенным от записи, возможен только на чтение.

Бит	DB[номер блока данных].DBX [адрес байта].[адрес бита]	DB1.DBX2.3
Байт, слово или двойное слово	DB[номер блока данных].DB [размер][адрес начального байта]	DB1.DBB4, DB10.DBW2, DB20.DBD8

Адресация входов/выходов в CPU и в модулях ввода/вывода



Когда вы вставляете CPU и модули ввода/вывода в экран со своей конфигурацией, то адреса I и Q назначаются автоматически.

Вы можете изменить адресацию, установленную по умолчанию, выбрав адресное поле в конфигурационном экране и введя туда новые числа. Цифровым входам и выходам адреса присваиваются в полных байтах (по 8 бит), не зависимо от того, использует ли модуль все входы и выходы или нет. Аналоговым входам и выходам адреса присваиваются группами по 2 входа или выхода в каждой группе (4 байта). В этом примере вы можете измеbнить адрес DI16 на 2..3 вместо 8..9. Инструментальное средство поможет вам, изменяя диапазоны адресов, которые имеют неправильный размер или вступают в конфликт с другими адресами.

На этом рисунке показан пример CPU 1214C с двумя SM.

3.3 Типы данных

Типы данных используются для указания размера элемента данных, а также того, как эти данные могут быть интерпретированы. Каждый параметр команды поддерживает, по крайней мере, один тип данных, а некоторые параметры поддерживают несколько типов данных. Подведите указатель мыши к полю параметра команды, чтобы увидеть, какие типы данных поддерживаются для соответствующего параметра.

Формальный параметр – это идентификатор на команде, который указывает адрес данных, подлежащих использованию командой (пример: вход IN1 команды ADD). Фактический параметр – это адрес или константа, где содержатся данные, подлежащие использованию командой (пример: %MD400 "Number_of_Widgets"). Тип данных фактического параметра, указанный вами, должен соответствовать одному из поддерживаемых типов данных формального параметра, определяемого командой.

При задании фактического параметра вы должны указать переменную (символ) или абсолютный адрес. Переменные связывают символическое имя (имя переменной) с типом данных, областью памяти, смещением в памяти, и комментарием и могут быть созданы в редакторе переменных ПЛК или в редакторе интерфейса для блока (OB, FC, FB или DB). Если вы вводите абсолютный адрес, не связанный ни с какой переменной, вы должны использовать подходящий размер, соответствующий поддерживаемому типу данных, тогда при вводе создается стандартная переменная.

Для многих входных параметров вы можете также вводить постоянное значение. В следующей таблице описаны поддерживаемые элементарные типы данных и даны примеры ввода констант. Все типы данных, кроме типа данных String [строка], доступны как в редакторе переменных ПЛК, так и в редакторах интерфейсов блоков. Тип String имеется только в редакторах интерфейсов блоков. В следующей таблице приведены элементарные типы данных.

Тип данных	**Размер (в битах)**	**Диапазон**	**Примеры ввода констант**
Bool	1	от 0 до 1	TRUE, FALSE, 0, 1
Byte	8	от 16#00 до 16#FF	16#12, 16#AB
Word	16	от 16#0000 до 16#FFFF	16#ABCD, 16#0001
DWord	32	от 16#00000000 до 16#FFFFFFFF	16#02468ACE
Char	8	от 16#00 до 16#FF	'A', 't', '@'
Sint	8	от -128 до 127	123, -123
Int	16	от -32768 до 32767	123, -123
Dint	32	от -2147483648 до 2147483647	123, -123
USInt	8	от 0 до 255	123
UInt	16	от 0 до 65,535	123
UDInt	32	0 до 4294967295	123
Real	32	от +/-1,18 x 10^{-38} до +/-3,40 x 10^{38}	123.456, -3.4, -1.2E+12, 3.4E-3
LReal	64	от +/-2,23 x 10^{-308} до +/-1,79 x 10^{308}	12345.123456789 -1.2E+40
Time	32	от T#-24d_20h_31m_23s_648ms до T#24d_20h_31m_23s_647ms Хранится как: от -2,147,483,648 мс до +2,147,483,647 мс	T#5m_30s 5#-2d T#1d_2h_15m_30x_45ms
String	переменный	от 0 до 254 символов в размере байта	'ABC'

Кроме того, командами преобразования поддерживается числовой формат BCD, хотя он и не предоставляется в распоряжение как тип данных.

Формат	Размер (в битах)	Числовой диапазон	Примеры ввода констант
BCD16	16	от -999 до 999	123, -123
BCD32	32	от -9999999 до 9999999	1234567, -1234567

Формат для вещественных чисел

Вещественные числа (или числа с плавающей точкой) представляются как 32-битовые числа с обычной точностью (Real) или 64-битовые числа с двойной точностью (LReal) в соответствии с описанием в стандарте ANSI/IEEE 754-1985. Числа с плавающей точкой обычной точности имеют точность до 6 значащих цифр, а числа с плавающей точкой двойной точности имеют точность до 15 значащих цифр. При вводе константы с плавающей точкой вы можете задать не более 6 (Real) или 15 (LReal) значащих цифр.

Расчеты, которые нуждаются в длинном ряде значений, включая очень большие и очень малые числа, могут привести к неточным результатам. Это может произойти, если числа отличаются в 10 в степени x раз, где x > 6 (Real) или 15 (LReal). Например (Real): 100 000 000 + 1 = 100 000 000.

Формат типа данных STRING

CPU поддерживает тип данных STRING для хранения последовательности однобайтовых символов. Тип данных STRING содержит общее число символов (число символов в строке) и фактическое число символов. Тип данных STRING предоставляет до 256 байтов для хранения максимального числа символов (1 байт), фактического числа символов (1 байт) и до 254 символов, каждый их которых хранится в 1 байте.

Вы можете использовать литеральные строки символов (константы) для параметров команд типа IN, используя одиночные кавычки. Например, ‘ABC’ – это строка из трех символов, которая может быть использована в качестве входа для параметра IN команды S_CONV. Вы можете создавать также строковые переменные, выбирая тип данных "String" в редакторе интерфейса блоков OB, FC, FB и DB. В редакторе переменных ПЛК создать строку символов невозможно. Вы можете указать максимальный размер строки в байтах при объявлении своей строки; например, "MyString[10]" определяет максимальный размер 10 байтов для MyString. Если вы не включаете квадратные скобки с указателем максимального размера, то принимается размер 254.

Следующий пример показывает тип данных STRING с максимальным числом символов 10 и фактическим числом символов 3. Это значит, что тип данных STRING содержит 3 однобайтовых символа, но может быть расширен до 10 однобайтовых символов.

Общее число символов	Фактическое число символов	Символ 1	Символ 2	Символ 3	...	Символ 10
10	3	'C' (16#43)	'A' (16#41)	'T' (16#54)	...	-
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	...	Byte 11

Массивы

Вы можете создать массив, содержащий несколько элементов одного элементарного типа. Массивы могут создаваться в редакторах интерфейса блоков OB, FC, FB и DB. Массив невозможно создать в редакторе переменных ПЛК.

Для создания массива в редакторе интерфейса блока выберите тип данных "Array [lo.. hi] of type", затем отредактируйте "lo", "hi" и "type" следующим образом:

- lo - начальный (наименьший) индекс для вашего массива
- hi - конечный (наибольший) индекс для вашего массива
- type – один из элементарных типов данных, например, BOOL, SINT, UDINT

Отрицательные индексы тоже поддерживаются. Вы можете дать имя массиву в столбце Name редактора интерфейса блока. В следующей таблице показаны примеры массивов в том виде, как они отображаются в редакторе интерфейса блоков:

Имя	Тип данных	Комментарий
My_Bits	Array [1.. 10] of BOOL	Этот массив содержит 10 булевых значений
My_Data	Array [-5.. 5] of SINT	Этот массив содержит 11 значений типа SINT, включая индекс 0

К элементам массива вы обращаетесь в своей программе, используя следующий синтаксис:

- Array_name[*i*], где *i* – желаемый индекс.

Примеры из редактора программ для ввода параметров:

- #My_Bits[3] – ссылается на третий бит массива "My_Bits"
- #My_Data[-2] - ссылается на четвертый элемент типа SINT массива "My_Data"

Символ # вставляется автоматически редактором программ.

Тип данных DTL (Data and Time Long)

Тип данных DTL – это структура из 12 байтов, которая хранит информацию о дате и времени в предопределенной структуре. Вы можете определить тип данных DTL во временной памяти блока или в DB.

Длина (байты)	Формат	Диапазон значений	Пример ввода значения
12	Время и календарь (год-месяц-день-час: минута:секунда.наносекунды)	мин.: DTL#1970-01-01-00:00:00.0 макс.: DTL#2554-12-31-23:59:59.999 999 999	DTL#2008-12-16-20:30:20.250

Каждый компонент типа данных DTL содержит свой тип данных и диапазон значений. Тип данных задаваемого значения должен совпадать с типом данных соответствующего компонента.

Байт	Компонент	Тип данных	Диапазон значений
0	Год	UINT	от 1970 до 2554
1			
2	Месяц	USINT	от 1 до 12
3	День	USINT	от 1 до 31
4	День недели	USINT	от 1(воскресенье) до 7(суббота) День недели в записи значения не учитывается.
5	Час	USINT	от 0 до 23
6	Минута	USINT	от 0 до 59
7	Секунда	USINT	от 0 до 59
8	Наносекунды	UDINT	от 0 до 999 999 999
9			
10			
11			

3.4 Использование карты памяти

ВНИМАНИЕ
CPU поддерживает только предварительно отформатированную карту памяти SIMATIC (стр. 370). Если вы снова отформатируете карту памяти SIMATIC под Windows, то CPU не сможет использовать эту карту памяти. Перед копированием программ на отформатированную карту памяти удалите с карты памяти все ранее сохраненные на ней программы.

Вы можете использовать карту памяти как передаточную карту или как программную карту. Каждая программа, которую вы копируете на карту памяти, содержит все кодовые блоки и блоки данных, все технологические объекты и 4конфигурацию устройств. Программа **не содержит** принудительно заданных значений.

- С помощью передаточной карты вы копируете программу во внутреннюю загрузочную память CPU без использования STEP 7 Basic. После того как вы вставили передаточную карту, CPU сначала удаляет из внутренней загрузочной памяти программу пользователя и все принудительно заданные значения, а затем копирует программу из передаточной карты во внутреннюю загрузочную память. Когда процесс передачи завершен, вы должны удалить передаточную карту.

  С помощью пустой передаточной карты вы можете получить доступ к CPU, защищенному паролем, если пароль был утерян или забыт (стр. 58). Вставка пустой передаточной карты удаляет программу, защищенную паролем, из внутренней загрузочной памяти CPU. После этого вы можете загрузить в CPU новую программу.

- Программная карта используется как внешняя загрузочная память для CPU. Вставка программной карты в CPU стирает всю внутреннюю загрузочную память CPU (программу пользователя и все принудительно задаваемые значения). Затем CPU исполняет программу, находящуюся во внешней загрузочной памяти (на программной карте). При загрузке в CPU с программной картой обновляется только внешняя загрузочная память (программная карта).

  Так как внутренняя загрузочная память CPU была стерта при вставке программной карты, то программная карта **должна** оставаться в CPU. Если вы удалите программную карту, CPU перейдет в состояние STOP. (Светодиод ошибки мигает, чтобы показать, что программная карта удалена.)

Программа на карте памяти содержит кодовые блоки, блоки данных, технологические объекты и конфигурацию устройств. Карта памяти **не содержит** принудительно задаваемых значений. Принудительно задаваемые значения не являются частью программы, но хранятся в загрузочной памяти, будь то внутренняя загрузочная память CPU или внешняя загрузочная память (программная карта). Если программная карта вставлена в CPU, то STEP 7 Basic применяет принудительно задаваемые значения только к внешней загрузочной памяти на программной карте.

3.4.1 Вставка карты памяти в CPU

⚠ ПРЕДУПРЕЖДЕНИЕ

Если вставить карту памяти (независимо от того, используется ли она как передаточная карта или как программная карта) в работающий CPU, то CPU немедленно переходит в состояние STOP. Устройства управления могут выходить из строя в небезопасных рабочих состояниях и вызвать из-за этого неконтролируемое поведение управляемого оборудования. Такое непредсказуемое поведение системы автоматизации может привести к гибели людей, тяжким телесным повреждениям и/или материальному ущербу. Всегда устанавливайте устройство аварийного отключения для своего приложения или процесса.

ОСТОРОЖНО

Электростатические разряды могут повредить карту памяти или предназначенное для нее гнездо в CPU.

При работе с картой памяти вы всегда должны находиться на токопроводящей заземленной площадке и/или носить заземленный браслет. Храните карту памяти в токопроводящем контейнере.

Чтобы вставить карту памяти, откройте верхнюю крышку CPU и вставьте карту памяти в гнездо. Штепсельный разъем позволяет легко вставлять и удалять карту памяти. Карта памяти имеет такую форму, что она может быть вставлена только надлежащим образом.

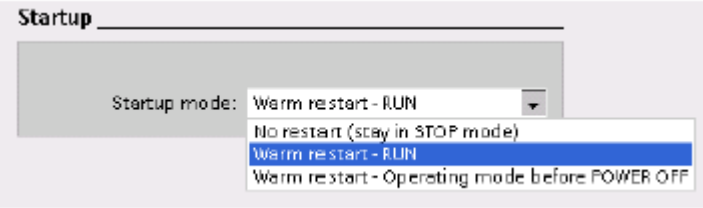
Обеспечьте, чтобы карта памяти не была защищена от записи. Для этого сдвиньте защитный переключатель из положения "Lock [Заблокировать]".

Указание

Если вставить карту памяти в CPU, находящийся в состоянии STOP, то диагностический буфер отображает сообщение о том, что начат анализ карты памяти. Проигнорируйте, пожалуйста, это сообщение. Анализ карты памяти начинается только тогда, когда вы переводите CPU в режим RUN, проводите полное стирание памяти CPU через MRES или выключаете, а затем снова включаете CPU.

3.4.2 Настройка параметров запуска CPU перед копированием проекта в карту памяти

Когда вы копируете программу на передаточную или программную карту, то эта программа содержит параметры запуска для CPU. Перед копированием программы на карту памяти всегда проверяйте, сконфигурирован ли режим работы CPU после выключения и последующего включения питания. Вы можете выбрать состояние (STOP, RUN или последнее перед выключением питания), в которое будет переходить CPU после восстановления питания.



Пояснения к рисунку: Startup – Запуск; Startup mode – Режим запуска; Warm restart – Теплый пуск; No restart (stay in STOP mode) – Нет запуска (оставаться в состоянии STOP); Warm restart –Operating mode before POWER OFF – Теплый пуск – Режим работы перед выключением питания.

3.4.3 Передаточная карта

ОСТОРОЖНО
Электростатические разряды могут повредить карту памяти или предназначенное для нее гнездо в CPU. При работе с картой памяти вы всегда должны находиться на токопроводящей заземленной площадке и/или носить заземленный браслет. Храните карту памяти в токопроводящем контейнере.

Создание передаточной карты

Никогда не забывайте сконфигурировать параметры запуска CPU (стр. 70) перед копированием программы на передаточную карту. Для создания передаточной карты действуйте следующим образом:

1. Вставьте пустую карту памяти в устройство для считывания карт, присоединенное к вашему устройству программирования.

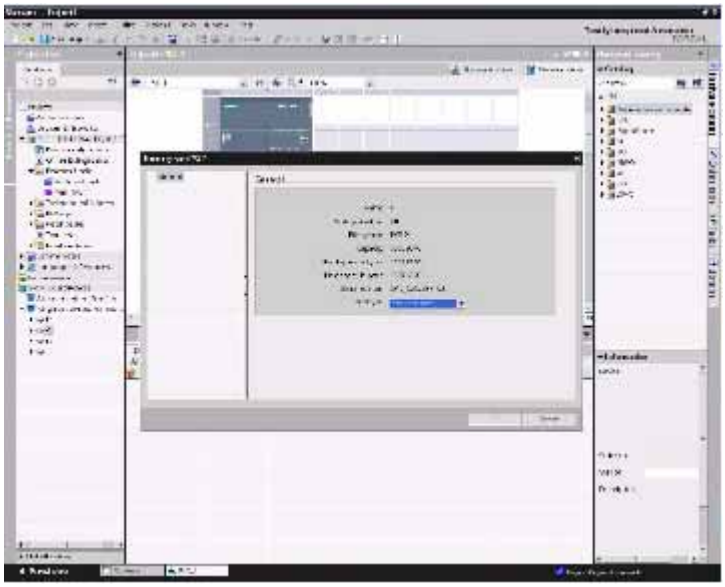
   (Если карта памяти не пуста, удалите папку "SIMATIC.S7S" и файл "S7_JOB.S7S", находящиеся на карте памяти, используя, например, проводник Windows.)

2. В дереве проекта (в проектном представлении) раскройте папку "SIMATIC Card Reader [Считыватель карт SIMATIC]" и выберите свое считывающее устройство.

3. Отобразите диалоговое окно "Memory Card [Карта памяти]", щелкнув правой клавишей мыши на считывающем устройстве и выбрав в контекстном меню опцию "Properties [Свойства]".

4. В диалоговом окне "Memory Card [Карта памяти]" выберите из ниспадающего меню пункт "Transfer [Передача]".

   После этого STEP 7 Basic создает пустую передаточную карту. Если вы создаете пустую передаточную карту, например, для восстановления утерянного пароля CPU (стр. 58), удалите передаточную карту из считывающего устройства.



5. Добавьте программу, выбрав CPU (например, ПЛК _1 [CPU 1214 DC/DC/DC]) в дереве проекта и перетащив CPU на карту памяти. (Другой способ - скопировать CPU и вставить в карту памяти.) Копирование CPU в карту памяти открывает диалоговое окно "Load preview [Загрузить предварительный просмотр]".

6. В диалоговом окне "Load preview" щелкните на кнопке "Load [Загрузить]", чтобы скопировать CPU в карту памяти.

7. Когда в диалоговом окне отобразится сообщение о том, что загрузка CPU (программы) произошла без ошибок, щелкните на кнопке "Finish [Закончить]".

Использование передаточной карты

Для передачи программы в CPU действуйте следующим образом:

1. Вставьте передаточную карту в CPU (стр. 69). Если CPU находится в режиме RUN, то он перейдет в состояние STOP. (Светодиод обслуживания мигает, чтобы показать, что карта памяти должна быть проанализирована.)
2. Для анализа карты памяти используйте один из следующих способов:
   - Выключите CPU и включите его снова.
   - Выполните переход из STOP в RUN.
   - Выполните полное стирание памяти (MRES).
3. После перезагрузки и анализа карты памяти CPU копирует программу во внутреннюю загрузочную память CPU. Когда процесс копирования завершен, светодиод обслуживания на CPU мигает, чтобы показать, что передаточная карта может быть удалена.
4. Удалите передаточную карту из CPU.
5. Чтобы проанализировать новую, перенесенную во внутреннюю загрузочную память программу, действуйте одним из следующих способов:
   - Выключите CPU и включите его снова.
   - Выполните переход из STOP в RUN.
   - Выполните полное стирание памяти (MRES).

После этого CPU переходит в режим (RUN или STOP), который вы сконфигурировали для проекта.

Указание

Перед переводом CPU в режим RUN вы должны вытащить передаточную карту.

3.4.4 Программная карта

ОСТОРОЖНО
Электростатические разряды могут повредить карту памяти или предназначенное для нее гнездо в CPU. При работе с картой памяти вы всегда должны находиться на токопроводящей заземленной площадке и/или носить заземленный браслет. Храните карту памяти в токопроводящем контейнере.

Обеспечьте, чтобы карта памяти не была защищена от записи. Для этого сдвиньте защитный переключатель из положения "Lock [Заблокировать]".

Перед копированием элементов программы в программную карту удалите с карты памяти все ранее сохраненные на ней программы.

Создание программной карты

При использовании в качестве программной карты карта памяти действует как внешняя загрузочная память CPU. Если удалить программную карту, то внутренняя загрузочная память CPU будет пустой.

Указание

Если вставить пустую карту памяти в CPU и выполнить ее анализ, выключив и включив CPU, выполнив переход из STOP в RUN или полное стирание памяти (MRES), то программа и принудительно задаваемые значения из внутренней загрузочной памяти CPU скопируются на карту памяти. (Карта памяти теперь является программной картой.) По окончании копирования программа во внутренней загрузочной памяти CPU стирается. После этого CPU переходит в режим, сконфигурированный для запуска (RUN или STOP).

Никогда не забывайте сконфигурировать параметры запуска CPU (стр. 70) перед копированием программы на программную карту. Для создания программной карты с помощью STEP 7 Basic действуйте следующим образом:

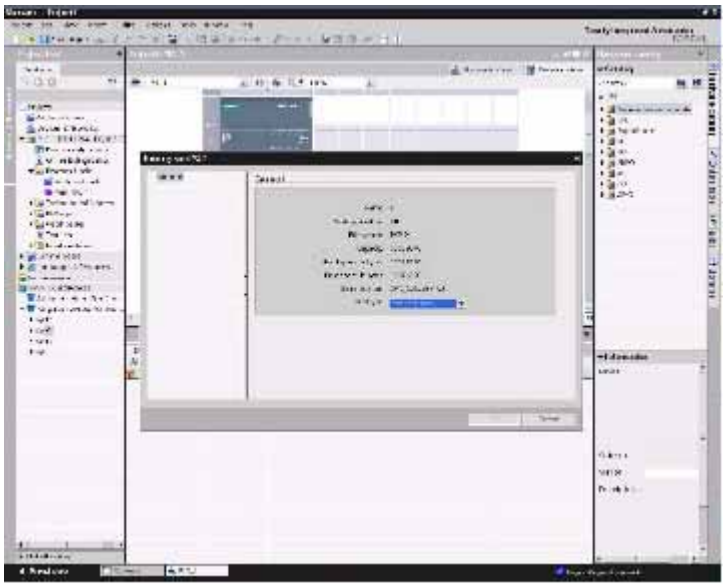
1. Вставьте пустую карту памяти в устройство для считывания карт, присоединенное к вашему устройству программирования.

   (Если карта памяти не пуста, удалите папку "SIMATIC.S7S" и файл "S7_JOB.S7S", находящиеся на карте памяти, используя, например, проводник Windows.)

2. В дереве проекта (в проектном представлении) раскройте папку "SIMATIC Card Reader [Считыватель карт SIMATIC]" и выберите свое считывающее устройство.

3. Отобразите диалоговое окно "Memory Card [Карта памяти]", щелкнув правой клавишей мыши на считывающем устройстве и выбрав в контекстном меню опцию "Properties [Свойства]".

4. В диалоговом окне "Memory Card [Карта памяти]" выберите из ниспадающего меню пункт "Program [Программа]".



5. Добавьте программу, выбрав CPU (например, ПЛК _1 [CPU 1214 DC/DC/DC]) в дереве проекта и перетащив CPU на карту памяти. (Другой способ - скопировать CPU и вставить в карту памяти.) Копирование CPU в карту памяти открывает диалоговое окно "Load preview [Загрузить предварительный просмотр]".

6. В диалоговом окне "Load preview" щелкните на кнопке "Load [Загрузить]", чтобы скопировать CPU в карту памяти.

7. Когда в диалоговом окне отобразится сообщение о том, что загрузка CPU (программы) произошла без ошибок, щелкните на кнопке "Finish [Закончить]".

Использование программной карты в качестве внешней загрузочной памяти для CPU

ОСТОРОЖНО

Если вставить пустую карту памяти в CPU, то CPU переходит в состояние STOP. Если вы выключите и включите снова CPU, переведете CPU из состояния STOP в режим RUN или выполните полный сброс памяти CPU (MRES), то CPU скопирует свою внутреннюю загрузочную память в карту памяти (благодаря чему карта памяти конфигурируется как программная карта) и сотрет программу из внутренней загрузочной памяти. Если вы удалите программную карту, то во внутренней загрузочной памяти CPU программы не будет.

Чтобы использовать программную карту со своим CPU, действуйте следующим образом:

1. Вставьте программную карту в CPU. Если CPU находится в режиме RUN, то он переходит в состояние STOP. Светодиод обслуживания мигает, показывая, что программная карта должна быть проанализирована
2. Для анализа программной карты действуйте одним из следующих способов:
   - Выключите CPU и включите его снова.
   - Выполните переход из STOP в RUN.
   - Выполните полное стирание памяти (MRES).
3. CPU перезагружается. После перезагрузки и анализа программной карты CPU стирает свою внутреннюю загрузочную память.

Затем CPU переходит в тот режим для запуска (RUN или STOP), который вы для него сконфигурировали.

Программная карта должна оставаться в CPU. Удаление программной карты оставляет CPU без программы во внутренней загрузочной памяти.

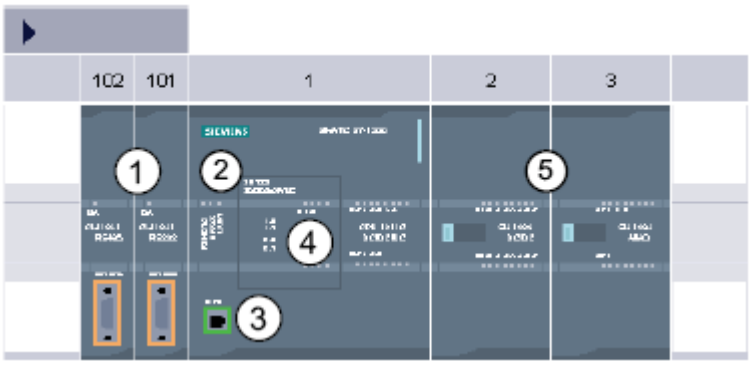
⚠ ПРЕДУПРЕЖДЕНИЕ

Если удалить программную карту, то CPU теряет свою внешнюю загрузочную память и генерирует ошибку. CPU переходит в состояние STOP, а светодиод ошибки мигает.

Устройства управления могут выходить из строя в небезопасных рабочих состояниях и вызвать из-за этого неконтролируемое поведение управляемого оборудования. Такое непредсказуемое поведение системы автоматизации может привести к гибели людей, тяжким телесным повреждениям и/или материальному ущербу.

4 Конфигурация устройств

Конфигурация устройств для вашего ПЛК создается добавлением CPU и других модулей в ваш проект.



① Коммуникационный модуль (CM): до 3, в слотах 101, 102, и 103
② CPU: слот 1
③ Порт Ethernet на CPU
④ Сигнальная плата (SB): макс. 1, вставляется в CPU
⑤ Сигнальный модуль (SM) для цифровых или аналоговых входов/выходов: до 8, вставляются в слоты со 2 по 9
(CPU 1214C допускает 8, CPU 1212C допускает 2, CPU 1211C не допускает ни одного)

Для создания конфигурации устройств сначала добавьте в свой проект одно устройство.

- В портальном представлении выберите портал "Devices & Networks [Устройства и сети]" и щелкните на "Add device [Добавить устройство]".

- В проектном представлении под именем проекта дважды щелкните на "Add new device [Добавить новое устройство]".

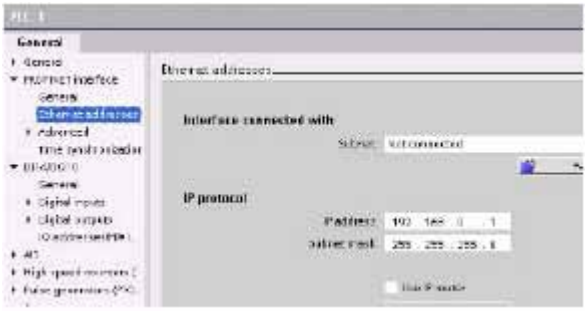
4.1 Вставка CPU

Для создания своей конфигурации устройств вставьте в свой проект CPU. Выбор CPU в диалоговом окне "Add a new device [Добавить новое устройство]" создает стойку и CPU.

Диалоговое окно "Add a new device"

Отображение набора устройств в конфигурации аппаратных средств.

Выбор CPU в отображении набора устройств выводит параметры CPU в окне просмотра параметров.



Указание

У CPU нет заранее сконфигурированного IP-адреса. Вы должны вручную назначить IP-адрес для CPU при создании конфигурации устройств. Если ваш CPU подключен к маршрутизатору в сети, то вы должны также ввести IP-адрес для маршрутизатора.

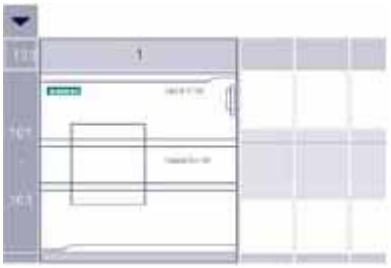
4.2 Выявление конфигурации для заранее не заданного CPU

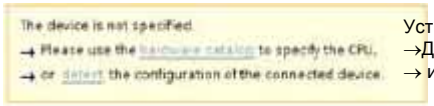
Загрузка существующей конфигурации аппаратуры очень проста

Если вы подключены к CPU, то вы можете загрузить конфигурацию этого CPU, включая возможно имеющиеся модули, в свой проект. Просто создайте для этого новый проект и выберите вместо определенного CPU "unspecified CPU [неопределенный CPU]". (Вы можете также полностью опустить создание конфигурации устройств, выбрав "Create a PLC program [Создать программу ПЛК]" через "First steps [Первые шаги]", после чего STEP 7 Basic автоматически создает неопределенный CPU.)

В программном редакторе в меню "Online" выберите команду "Hardware detection [Распознавание аппаратуры]".

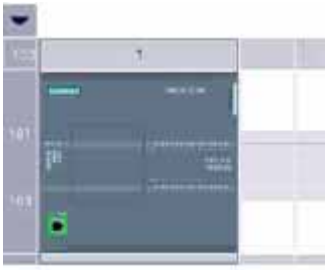
В редакторе конфигурации устройств выберите опцию для распознавания конфигурации подключенного устройства.





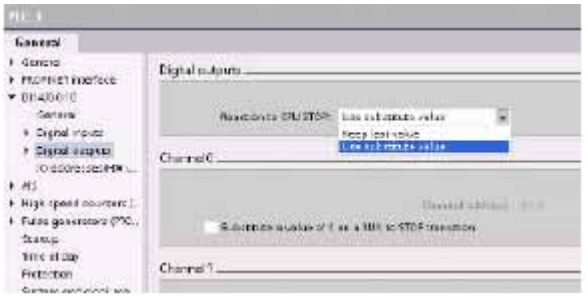
Устройство не определено
→Для определения CPU используйте, пожалуйста, каталог аппаратуры
→ или выполните распознавание конфигурации подключенного устройства

После того как вы выбрали CPU в диалоговом окне Online, STEP 7 Basic загружает конфигурацию аппаратуры из CPU, включая возможные модули (SM, SB или CM). Затем вы можете конфигурировать параметры для CPU и модулей.



4.3 Конфигурирование работы CPU

Для конфигурирования рабочих параметров CPU выберите CPU в отображении набора устройств (синяя рамка вокруг всего CPU) и откройте вкладку "Properties [Свойства]" в окне просмотра параметров.



В окне свойств вы можете установить следующие параметры:

- Интерфейс PROFINET: Установка IP-адреса для CPU и синхронизации времени
- DI, DO, и AI: Настройка поведения локальных (встроенных) цифровых и аналоговых входов и выходов
- Скоростные счетчики и генераторы импульсов: Активизация и настройка быстрых счетчиков (HSC) и генераторов импульсов, используемых для операций с последовательностями импульсов (pulse-train operations, PTO) и широтно-импульсной модуляции (pulse-width modulation, PWM)

  Когда вы конфигурируете выходы CPU или сигнальной платы в качестве генераторов импульсов (для использования с PWM или основными командами управления перемещениями), соответствующие адреса выходов (Q0.0, Q0.1, Q4.0 и Q4.1) удаляются из памяти выходов (Q) и не могут быть использованы для других целей в вашей пользовательской программе. Если ваша пользовательская программа запишет какое-либо значение в выход, используемый в качестве генератора импульсов, то CPU не запишет это значение в физический выход.
- Запуск: Настройка поведения CPU после выключения и последующего включения, например, для запуска в состоянии STOP или перехода в режим RUN после теплого пуска
- Время суток: Установка времени, часового пояса и переключения между летним и зимним временем
- Защита: Установка защиты от чтения/записи и пароля для доступа к CPU
- Системная и тактовая битовая память (тактовые меркеры): Установка байта для функций "системной памяти" (для битов "первый цикл", "всегда включен" и "всегда выключен") и установка байта для функций "тактовой памяти" (где каждый бит включается и выключается с заранее заданной частотой).
- Время цикла: Установка максимального времени цикла или фиксированного минимального времени цикла
- Коммуникационная нагрузка: Назначение процентной доли времени CPU для коммуникационных задач

4.4 Добавление модулей к конфигурации

Для добавления модулей к CPU используется каталог аппаратуры. Имеется три типа модулей:

- Сигнальные модули (SM) предоставляют дополнительные цифровые или аналоговые входы и выходы. Эти модули подключаются с правой стороны от CPU.
- Сигнальные платы (SB) предоставляют лишь ограниченное число входов и выходов для CPU. SB устанавливается с передней стороны CPU.
- Коммуникационные модули (CM) предоставляют дополнительный коммуникационный порт (RS232 или RS485) для CPU. Эти модули подключаются с левой стороны от CPU.

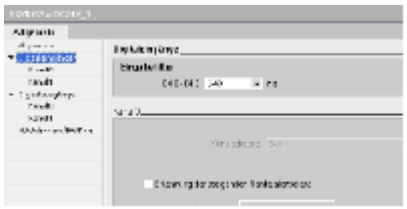
Для вставки модуля в конфигурацию аппаратуры выберите модуль в каталоге аппаратуры и дважды щелкните на нем или перетащите модуль в отмеченный слот.

Модуль	Выберите модуль	Вставьте модуль	Результат
SM			
SB			
CM			

4.5 Конфигурирование параметров модулей

Для конфигурирования рабочих параметров модулей выберите модуль в отображении набора устройств и откройте вкладку "Properties [Свойства]" в окне просмотра параметров.

Конфигурирование сигнального модуля (SM) или сигнальной платы (SB)



- Цифровые входы и выходы: Входы могут быть сконфигурированы для обнаружения нарастающих или падающих фронтов (связывая каждый из них с событием и аппаратным прерыванием), а также "захвата импульсов" (вход остается включенным после импульса) вплоть до следующего обновления образа процесса на входах. Выходы могут использовать заменяющие значения или быть заморожены.
- Аналоговые входы и выходы: Для отдельных входов вы конфигурируете параметры, например, вид измерения (напряжение или ток), диапазон и сглаживание, а также разблокирование диагностики положительного и отрицательного переполнения. Выходы предоставляют такие параметры, как, например, вид выхода (напряжение или ток) и диагностика, например, короткое замыкание (для потенциальных выходов) или диагностика нарушения верхних или нижних граничных значений.
- диагностические адреса входов и выходов: Конфигурирование начальных адресов для входов и выходов модуля

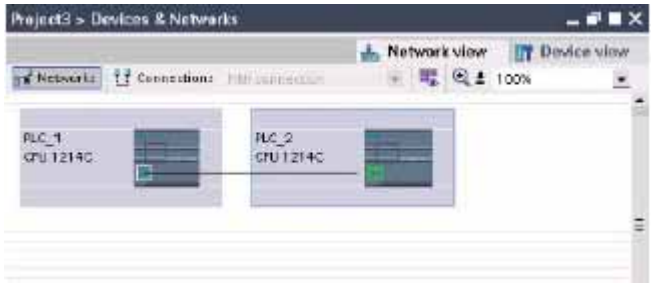
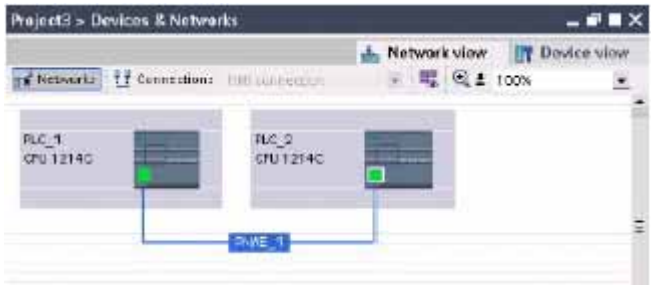
Конфигурирование коммуникационного модуля (CM)

- Конфигурирование порта: Конфигурирование коммуникационных параметров, например, скорость передачи, контроль четности, биты данных, стоповые биты, управление потоком, Символы XON и XOFF и время ожидания
- Конфигурирование передаваемого сообщения: Разблокирование и конфигурирование опций, связанных с передачей
- Конфигурирование принимаемого сообщения: Разблокирование и конфигурирование параметров для начала и окончания сообщения

Эти параметры конфигурации могут быть изменены вашей программой.

4.6 Создание сетевого соединения

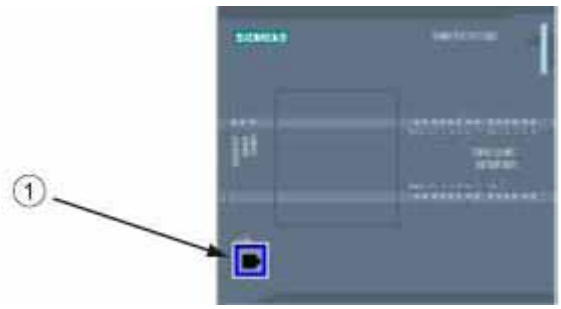
Для создания сетевых соединений между устройствами в вашем проекте используйте "Network view [Отображение сети]" в конфигурации устройств. После создания сетевого соединения вы можете во вкладке "Properties [Свойства]" окна просмотра параметров сконфигурировать параметры сети.

Действие	**Результат**
Выберите "Network view [Отображение сети]" для отображения устройств, подлежащих соединению.	
Выберите порт на одном устройстве и протащите соединение к порту второго устройства.	
Отпустите кнопку мыши для создания соединения.	

4.7 Конфигурирование IP-адреса в вашем проекте

Конфигурирование интерфейса PROFINET

После того как вы сконфигурировали стойку с CPU (стр. 80) , вы можете сконфигурировать параметры интерфейса PROFINET. Для этого щелкните на зеленом поле PROFINET на CPU, чтобы выбрать порт PROFINET. Во вкладке "Properties [Свойства]" окна просмотра параметров отображается порт PROFINET.



① порт PROFINET

Конфигурирование IP-адреса

Адрес Ethernet (MAC-адрес): В сети PROFINET каждому устройству для идентификации производителем назначается адрес управления доступом к среде передачи данных (MAC-адрес, Media Access Control address). MAC-адрес состоит из шести групп по две шестнадцатеричных цифры в каждой, отделенных друг от друга дефисами (-) или двоеточиями (:) в порядке передачи (например, 01-23-45-67-89-AB или 01:23:45:67:89:AB).

IP-адрес: Каждое устройство должно также иметь протокольный адрес Интернет (Internet Protocol address, IP-адрес). Этот адрес позволяет устройству поставлять данные через боле сложные, маршрутизированные сети.

Каждый IP-адрес делится на четыре сегмента по 8 бит в каждом и представляется в десятичном формате с разделительными точками (например, 211.154.184.16). Первая часть IP-адреса является идентификатором сети ID (в какой сети вы находитесь?), а вторая часть адреса является идентификатором хоста (уникален для каждого устройства в сети). IP-адрес 192.168.x.y является стандартным обозначением, распознаваемым как часть частной сети, которая не находится в Интернете.

Маска подсети: Подсеть – это логическая группировка связанных между собой сетевых устройств. Абоненты (узлы) подсети обычно находятся в физической близости друг от друга в одной локальной сети (Local Area Network, LAN). Маска (сетевая маска или маска подсети) определяет границы подсети IP.

Маска подсети 255.255.255.0 обычно пригодна для малой локальной сети. Это значит, что все IP-адреса в этой сети должны имеет одинаковые первые 3 октета, и различные устройства в этой сети идентифицируются последним октетом (8-битовым полем). Примером этого является назначение маски подсети 255.255.255.0 и IP-адресов от 192.168.2.0 до 192.168.2.255 устройствам в малой локальной сети.

Единственное соединение между различными подсетями осуществляется через маршрутизатор. Если используются подсети, то должен использоваться IP-маршрутизатор.

IP-маршрутизатор: Маршрутизаторы являются связующим звеном между локальными сетями. С помощью маршрутизатора компьютер в локальной сети может посылать сообщения в другие сети, за которыми, возможно, есть другие локальные сети. Если получатель данных не находится в этой локальной сети, то маршрутизатор передает данные дальше в другую сеть или группу сетей, где они могут быть доставлены получателю.
Для передачи и приема пакетов данных маршрутизаторам нужны IP-адреса.

Свойства IP-адресов: В окне свойств (Properties) выберите запись "Ethernet address [Адрес Ethernet]". Портал комплексной автоматизации (TIA-портал) отображает диалоговое окно для конфигурирования адреса Ethernet, в котором вы проекту программного обеспечения ставите в соответствие IP-адрес CPU, в который загружается проект.

Указание

У CPU нет заранее сконфигурированного IP-адреса. Поэтому IP-адрес для CPU вы должны назначить вручную. Если ваш CPU подключен к маршрутизатору или к сети, то вы должны также ввести IP-адрес маршрутизатора. Все IP-адреса конфигурируются при загрузке проекта.

Дальнейшую информацию вы найдете в разделе "Назначение IP-адресов устройству программирования и сетевым устройствам (стр. 252)".

Следующая таблица содержит параметры для IP-адреса:

Параметр	Описание	
Subnet [Подсеть]	Имя подсети, к которой подключено устройство. Чтобы создать новую подсеть, щелкните на кнопке "Add new subnet [Добавить новую подсеть]". По умолчанию "Not connected [Не подключено]". Возможны два типа соединения: • Настройка по умолчанию "Not connected" предоставляет локальное соединение. • Подсеть необходима, если ваша сеть содержит два или более устройств.	
IP protocol	IP address	Назначенный IP-адрес CPU
	Subnet mask [Маска подсети]	Назначенная маска подсети
	Use IP router [Использовать IP-маршрутизатор]	Щелкните на этой триггерной кнопке, если используется IP-маршрутизатор
	Router address [Адрес маршрутизатора]	Назначенный IP-адрес маршрутизатора, если имеется

5 Основы программирования

5.1 Указания по проектированию системы с ПЛК

При проектировании системы с ПЛК у вас есть возможность выбора из ряда методов и критериев. Следующие общие указания применимы ко многим проектам. Разумеется, вы должны придерживаться процедур, принятых в вашей компании и учитывать собственный опыт.

Рекомендуемые шаги	**Задачи**
Разделите ваш процесс или установку на части	Разделите ваш процесс или установку на части, не зависящие друг от друга. Эти части определяют границы между контроллерами и влияют на перечень функциональных описаний и распределение ресурсов.
Создайте перечень функциональных описаний	Сделайте описания функций для каждой части процесса или установки, например, входы и выходы, функциональное описание процесса, состояния, которые могут быть достигнуты, прежде чем может начать реагировать исполнительное устройство (например, электромагнитный клапан, двигатель или привод), описание интерфейса оператора и всех интерфейсов с другими частями процесса или установки.
Проектирование цепей аварийной защиты	Определите устройства, которым для обеспечения безопасности может потребоваться схемно-реализованная логика. Помните, что устройства управления могут входить из строя небезопасным образом, что может привести к неожиданному запуску или изменению в работе оборудования. Там, где неожиданная или неправильная работа оборудования может привести к телесным повреждениям людей или существенному материальному ущербу, подумайте о введении электромеханических блокирующих устройств (которые действуют независимо от ПЛК) для предотвращения опасных режимов. Для разработки цепей аварийной защиты действуйте следующим образом: • Выявите, где возможно ненадлежащее или неожиданное функционирование исполнительных устройств, которое может вызвать опасное состояние. • Определите условия, при которых эксплуатация оборудования безопасна, и определите, как обнаружить эти условия независимо от ПЛК. • Определите, как ПЛК влияет на процесс, когда включается и снова отключается напряжение, а также определите, как и где могут быть обнаружены ошибки. Используйте эту информацию только для проектирования нормальных и ожидаемых ненормальных режимов работы. Из соображений безопасности не полагайтесь на этот сценарий "наилучшего случая". • Спроектируйте цепи ручной или электромеханической защиты, с помощью которых опасные процессы блокируются независимо от ПЛК. • Обеспечьте передачу соответствующей информации о состоянии от независимых цепей в ПЛК, так чтобы программа и интерфейсы операторов обладали этой информацией. • Определите другие требования техники безопасности для безопасного протекания процесса.
Определите размещение станций оператора	На основе требований, содержащихся в перечне функциональных описаний, разработайте следующие планы станций оператора: • Обзорный чертеж, показывающий расположение каждой станции оператора относительно процесса или установки. • Чертеж расположения устройств для станции оператора, например, дисплея, переключателей и ламп. • Электрические чертежи с соответствующими входами и выходами ПЛК и сигнальных модулей.

Рекомендуемые шаги	Задачи
Разработайте конфигурационные чертежи	На основе требований, содержащихся в перечне функциональных описаний, разработайте конфигурационные чертежи управляющего оборудования: • Обзорный чертеж, показывающий расположение каждого ПЛК относительно процесса или установки. • Чертеж механического расположения каждого ПЛК и всех модулей ввода/вывода, включая все шкафы и другое оборудование. • Электрические чертежи для каждого ПЛК и всех модулей ввода/вывода, включая номера моделей устройств, коммуникационные адреса и адреса входов и выходов.
Создайте список символических имен	Создайте список символических имен для абсолютных адресов. Укажите не только физические входы и выходы, но также и другие элементы (например, имена переменных), которые вы используете в своей программе.

5.2 Структурирование программы пользователя

При создании пользовательской программы для решения задачи автоматизации команды для программы вставляются в кодовые блоки:

- Организационный блок (OB) реагирует на определенное событие в CPU и может прервать исполнение программы пользователя. Стандартный блок для исполнения программы пользователя (OB 1) предоставляет основную структуру для вашей пользовательской программы и является единственным кодовым блоком, необходимым для пользовательской программы. Если вы вставите другие OB в свою программу, то эти OB прерывают исполнение OB 1. Другие OB выполняют специфические функции, например, для задач запуска, для обработки прерываний и ошибок или для исполнения конкретного программного кода через определенные интервалы времени.

- Функциональный блок (FB) – это подпрограмма, которая исполняется при вызове из другого кодового блока (OB, FB или FC). Вызывающий блок передает параметры в FB, а также определяет некоторый блок данных (DB), который сохраняет данные для этого вызова или экземпляра этого FB. Изменение экземплярного DB позволяет родовому FB управлять работой группы устройств. Например, один FB может управлять несколькими насосами или вентилями с помощью различных экземплярных DB, содержащих конкретные рабочие параметры для каждого насоса или вентиля.

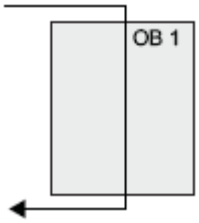
- Функция (FC) – это подпрограмма, которая исполняется при вызове из другого кодового блока (OB, FB или FC). У FC нет связанного с ней экземплярного кодового DB. Вызывающий блок передает параметры в FC. Выходные значения FC должны быть записаны в адреса памяти или в глобальный DB.

Выбор структуры для программы пользователя

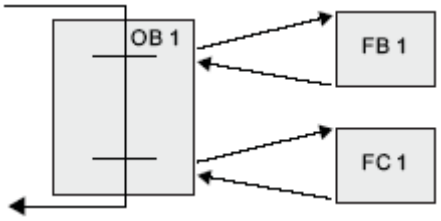
В зависимости от требований вашего приложения вы можете выбрать для своей пользовательской программы линейную или модульную структуру:

- Линейная программа выполняет все команды для ваших задач автоматизации последовательно друг за другом. Обычно в линейной программе все команды находятся в ОВ цикла (ОВ 1).
- Модульная программа вызывает специальные кодовые блоки, которые выполняют конкретные задачи. Для создания модульной структуры сложная задача автоматизации делится на небольшие подзадачи, соответствующие технологическим функциям процесса. Каждый кодовый блок содержит сегмент программы для соответствующей подзадачи. Вы структурируете свою программу, вызывая один кодовый блок из другого.

Линейная структура:



Модульная структура:



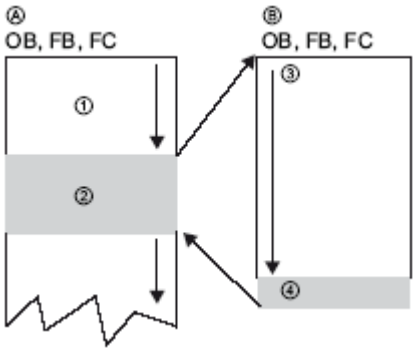
Разработку и реализацию программы пользователя можно упростить, создавая типовые кодовые блоки, которые могут быть многократно использованы в программе. Использование типовых кодовых блоков имеет ряд преимуществ:

- Вы можете создавать повторно используемые кодовые блоки для стандартных задач, например, для управления насосом или двигателем. Вы можете также хранить эти типовые кодовые блоки в библиотеке, которая может быть использована для различных приложений или решений.
- Когда вы структурируете пользовательскую программу на модульные компоненты, соответствующие функциональным задачам, ваша программа становится более наглядной, и с ней легче обращаться. Модульные компоненты не только помогают стандартизовать разработку программы, но также ускоряют и облегчают ее адаптацию и модификацию.
- Создание модульных компонентов упрощает отладку вашей программы. Если вся программа разделена на последовательность модульных программных сегментов, вы можете тестировать функции каждого кодового блока непосредственно во время разработки.
- Создание модульных компонентов, относящихся к конкретным технологическим функциям, упрощает и сокращает ввод в эксплуатацию всего приложения.

5.3 Использование блоков для структурирования вашей программы

Модульные кодовые блоки вы создаете путем разработки функциональных блоков и функций для выполнения типовых задач. Затем вы структурируете свою задачу, вызывая повторно используемые кодовые блоки из других кодовых блоков. Вызывающие блоки передают параметры конкретного устройства в вызываемый блок.

A Вызывающий блок
B Вызываемый (или прерывающий) блок
① Исполнение программы
② Операция, которая вызывает другой блок
③ Исполнение программы
④ Конец блока (возврат в вызывающий блок)

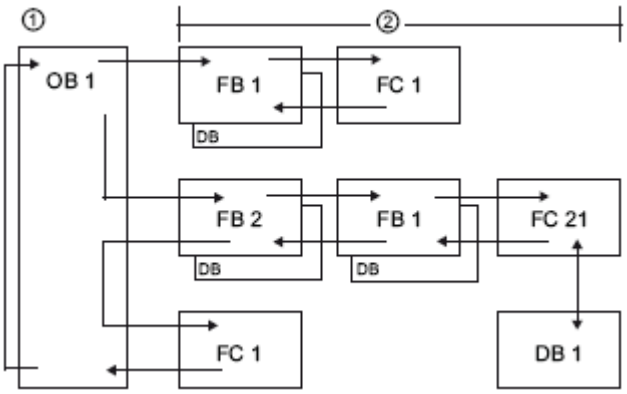


Если кодовый блок вызывается другим кодовым блоком, CPU исполняет программный код в вызванном блоке. После того как вызванный блок обработан, CPU возобновляет исполнение вызывающего блока.

Обработка продолжается исполнением команды, следующей за вызовом блока.

Вызовы блоков могут быть вложены друг в друга, делая структуру еще более модульной.

① Начало цикла
② Глубина вложения



Создание повторно используемых кодовых блоков

Для создания OB, FB, FC и глобальных DB используйте диалоговое окно "Add new block [Добавить новый блок]", которое открывается через "Program blocks [Программные блоки]" в дереве проекта.

Когда вы создаете кодовый блок, вы должны выбрать язык программирования для этого блока. Не выбирайте язык программирования для DB, так как он только хранит данные.

5.3.1 Организационный блок (OB)

Организационные блоки служат для структурирования вашей программы. Они образуют интерфейс между операционной системой и программой пользователя. OB управляются событиями. Событие, например, диагностическое прерывание или интервал времени, побуждает CPU к исполнению OB. Некоторые OB имеют предопределенные стартовые события и поведение.

OB программного цикла содержит вашу главную программу. Вы можете включить в свою пользовательскую программу боле одного OB программного цикла. В режиме RUN OB программного цикла выполняются с наименьшим уровнем приоритета и могут быть прерваны всеми другими видами обработки программы. OB запуска не прерывает OB программного цикла, так как CPU выполняет OB запуска до перехода в режим RUN.

После обработки OB программного цикла CPU немедленно исполняет этот OB снова. Циклическая обработка является "нормальным" видом обработки для программируемых логических контроллеров. Во многих приложениях вся пользовательская программа содержится в одном OB программного цикла.

Вы можете создавать другие OB для выполнения определенных функций, например, при запуске, для обработки прерываний и ошибок или для исполнения определенного программного кода через определенные интервалы времени. Эти OB прерывают исполнение OB программного цикла.

Для создания новых OB в своей пользовательской программе используйте диалоговое окно "Add new block [Добавить новый блок]".

В зависимости от уровня своего приоритета один OB может прерывать работу другого OB. Обработка прерываний всегда управляется событиями. Когда такое событие происходит, CPU прерывает исполнение программы пользователя и вызывает OB, который был спроектирован для обработки этого события. После завершения прерывающего OB CPU возобновляет исполнение программы пользователя с точки прерывания.

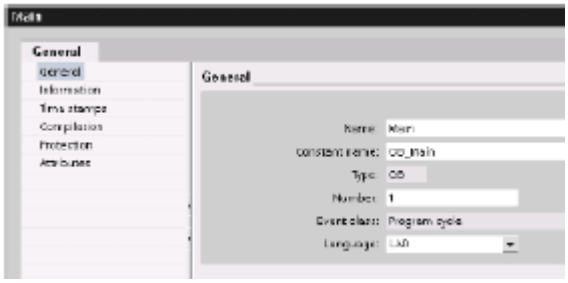
CPU определяет порядок обработки прерывающих событий на основе приоритетов, назначенных отдельным OB. Каждое событие имеет определенный приоритет обслуживания. Несколько прерывающих событий могут быть объединены в классы приоритета. За дальнейшей информацией обратитесь к разделу главы Основы ПЛК об исполнении программы пользователя (стр. 39).

Создание дополнительного OB внутри класса OB

Вы можете создать несколько OB для своей пользовательской программы, в том числе для классов OB программного цикла и OB запуска. Для создания OB используйте диалоговое окно "Add new block [Добавить новый блок]". Введите имя для вашего OB и номер OB, который должен быть больше 200.

Если вы создаете несколько OB программного цикла для своей пользовательской программы, то CPU исполняет отдельные OB программного цикла в порядке возрастания номеров, начиная с OB главного программного цикла (по умолчанию это OB 1). Например: после первого OB программного цикла (OB1) CPU исполняет второй OB программного цикла (например, OB 200).

Конфигурирование режима функционирования OB



Вы можете изменять параметры функционирования OB. Например, вы можете установить параметр времени для OB с запаздыванием или для циклического OB.

5.3.2 Функция (FC)

Функция (FC) – это кодовый блок, который обычно выполняет определенную операцию с набором входных значений. FC сохраняет результаты этой операции в определенных местах памяти.

Вы можете использовать FC для выполнения следующих задач:

- Стандартные и многократно выполняемые операции, например, математические расчеты.
- Выполнения технологических функций, например, для отдельных процессов управления, использующих двоичную логику.

FC может также вызываться несколько раз в различных местах программы. Это повторное использование упрощает программирование часто повторяющихся задач.

У FC нет связанного с ним экземплярного блока данных (DB). FC использует стек локальных данных для временных данных, используемых для расчета. Временные данные не сохраняются. Для длительного хранения данных выходная величина должна быть присвоена адресу в глобальной памяти, например, в M-памяти или в глобальном DB.

5.3.3 Функциональный блок (FB)

Функциональный блок (FB) – это кодовый блок, который использует экземплярный блок данных для своих параметров и статических данных. FB имеют переменную память, которая находится в блоке данных (DB) или в "экземплярном" DB. Экземплярный DB предоставляет блок памяти, связанный с экземпляром (или вызовом) FB, и хранит данные после исполнения FB. Вы можете назначить различным вызовам FB различные экземплярные DB. Экземплярный DB позволяет вам использовать один типовой FB для управления несколькими устройствами. Вы можете структурировать свою программу, тем что один кодовый блок вызывает FB и экземплярный DB. Затем CPU исполняет программный код в этом FB и сохраняет параметры блока и статические локальные данные в экземплярном DB. Когда исполнение FB заканчивается, CPU продолжает исполнение кодового блока, который вызвал FB. Экземплярный DB сохраняет значения для этого экземпляра FB. Эти значения находятся в распоряжении последующих вызовов этого функционального блока в том же самом или в других циклах сканирования.

Повторно используемые кодовые блоки со связанной с ними памятью

Обычно FB используется для управления последовательностью действий или устройствами, которые не заканчивают свою работу в течение одного цикла. Для сохранения рабочих параметров так, чтобы к ним можно было быстро обращаться от одного цикла к другому, каждый FB в вашей пользовательской программе имеет один или несколько экземплярных DB. Вызывая FB, вы также указываете экземплярный DB, содержащий параметры блока и статические локальные данные для этого вызова или "экземпляра" FB. Экземплярный DB сохраняет эти значения по окончании исполнения FB.

Спроектировав FB для типовых задач управления, вы можете повторно использовать этот FB для нескольких устройств, выбирая различные экземплярные DB для различных вызовов FB.

FB сохраняет в экземплярном DB входные (IN), выходные (OUT) и изменяемые (IN_OUT) параметры.

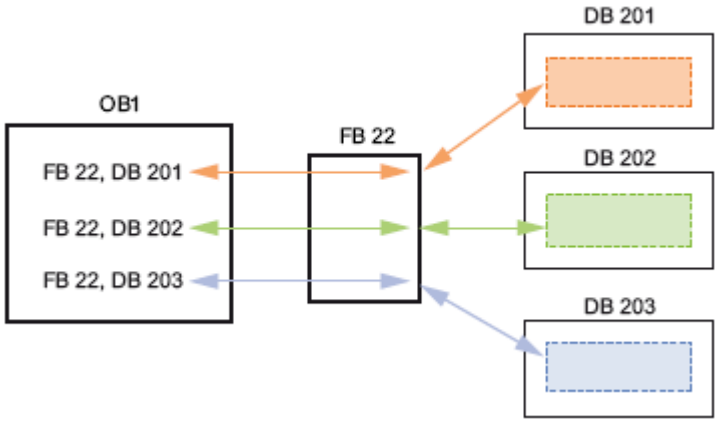
Присваивание начальных значений

Если входным, выходным или изменяемым параметрам функционального блока (FB) не присвоены значения, то используются значения, хранящиеся в экземплярном блоке данных (DB). В некоторых случаях вы должны назначить параметры.

Вы можете назначить параметрам начальные значения в интерфейсе FB. Эти значения передаются в соответствующий экземплярный DB. Если вы не назначаете параметры, то используются значения, хранящиеся в настоящее время в экземплярном DB.

Использование одного FB с несколькими DB

На следующем рисунке показан OB, который трижды вызывает FB, используя разные блоки данных для каждого вызова. Эта структура позволяет одному и тому же FB управлять несколькими одинаковыми устройствами, например, двигателями, назначая каждому вызову того или иного устройства свой экземплярный блок данных. Каждый экземплярный DB хранит данные (например, скорость, длительность запуска и общее время работы) для отдельного устройства. В этом примере FB 22 управляет тремя отдельными устройствами, причем DB 201 хранит эксплуатационные данные для первого устройства, DB 202 хранит эксплуатационные данные для второго устройства, и DB 203 хранит эксплуатационные данные для третьего устройства.



5.3.4 Блок данных (DB)

Вы можете создавать в своей пользовательской программе блоки данных (DB) для сохранения данных для кодовых блоков. Все программные блоки в программе пользователя могут обращаться к данным в глобальном DB, но экземплярный DB хранит данные для конкретного функционального блока (FB). Вы можете определить DB как предназначенный только для чтения.

Данные, хранящиеся в DB, не удаляются, когда заканчивается исполнение соответствующего кодового блока. Имеется два вида DB:

- Глобальный DB хранит данные для кодовых блоков в вашей программе. Каждый OB, FB или FC может обратиться к данным в глобальном DB.
- Экземплярный DB хранит данные для конкретного FB. Структура данных в экземплярном DB отражает параметры (Input, Output и InOut) и статические данные для FB. (Временная память для FB в экземплярном DB не храниться.)

Указание

Хотя экземплярный DB хранит данные для конкретного FB, любой кодовый блок может получить доступ к данным в экземплярном DB.

5.4 Согласованность данных

CPU поддерживает согласованность данных для всех элементарных типов данных (например, Word или DWord) и всех определяемых системой структур (например, IEC_TIMERS или DTL). Процесс чтения или записи такого значения не может быть прерван. (Например, CPU защищает доступ к значению типа DWord, пока все четыре байта DWord не будут прочитаны или записаны.) Чтобы гарантировать невозможность одновременной записи в одно и то же место памяти со стороны OB программного цикла и OB прерываний, CPU не выполняет OB прерываний, пока операция чтения или записи в OB программного цикла не будет завершена.

Если в вашей пользовательской программе несколько значений в памяти используются OB программного цикла и OB прерываний, то ваша пользовательская программа должна также обеспечить, чтобы эти значения изменялись или считывались согласованно. Чтобы защитить доступ к совместно используемым значениям, вы можете использовать в своем OB программного цикла команды DIS_AIRT и EN_AIRT.

- Вставьте в кодовый блок DIS_AIRT, чтобы не допустить исполнения OB прерываний во время операции чтения или записи.
- Вставьте команды чтения или записи значений, к4оторые может изменить OB прерываний.
- Вставьте команду EN_AIRT в конце этой последовательности, чтобы отменить DIS_AIRT и разрешить исполнение OB прерываний.

Коммуникационный запрос от устройства человеко-машинного интерфейса или другого CPU также может прервать исполнение OB программного цикла. Коммуникационные запросы также могут привести к проблемам с согласованностью данных. CPU гарантирует, что элементарные типы данных всегда будут считываться и записываться согласованно командами программы пользователя. Так как программа пользователя периодически прерывается коммуникационными запросами, то нет возможности гарантировать, чтобы несколько значений в CPU не обновлялись одновременно устройствами человеко-машинного интерфейса. Например, значения, отображаемые на экране устройства человеко-машинного интерфейса, могут происходить из различных циклов сканирования CPU.

Команды двухточечной связи (Point-to-Point, PtP) и команды PROFINET (например, TSEND_C и TRCV_C) передают буфера данных, которые могут быть прерваны. Обеспечивайте согласованность данных для этих буферов, избегая операций чтения или записи в эти буфера как в OB программного цикла, так и в OB прерываний. Если же необходимо изменить значения в буфере для этих команд в OB прерываний, воспользуйтесь командой DIS_AIRT, чтобы отложить прерывание (OB прерываний или коммуникационное прерывание от устройства человеко-машинного интерфейса или другого CPU), пока не будет выполнена команда EN_AIRT.

Указание

Использование команды DIS_AIRT задерживает обработку OB прерываний до тех пор, пока не будет выполнена команда EN_AIRT, оказывает воздействие на латентность прерывания (интервал времени от возникновения события до исполнения OB прерываний) в вашей пользовательской программе.

5.5 Выбор языка программирования

В качестве языка программирования вы можете выбрать цепную логическую схему (ladder logic, LAD), называемую также контактным планом, или функциональную блок-схему (Function Block Diagram, FBD), называемую также функциональным планом.

Язык программирования LAD

LAD – это графический язык программирования. Это представление основано на схемах электрических соединений.

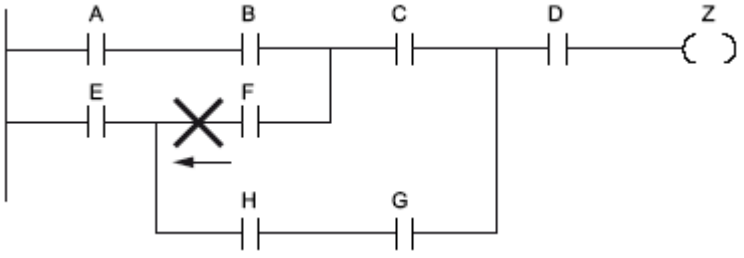
Элементы схемы, например, замыкающий и размыкающий контакты, и катушки реле соединены в сети.

Для создания логики для сложных операций вы можете вставлять разветвления для формирования параллельных цепей. Параллельные цепи открываются вниз или подключаются непосредственно к шине электропитания. Разветвления оканчиваются вверх.

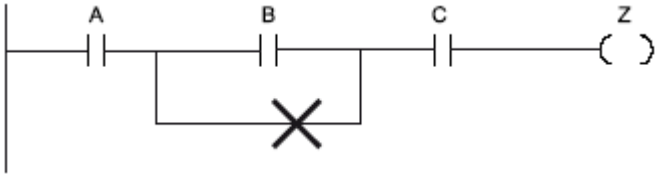
LAD предоставляет команды в виде блоков для ряда функций, например, арифметических операций, таймеров, счетчиков и пересылок.

При создании сетей LAD примите во внимание следующие правила:

- Каждая сеть LAD должна завершаться катушкой или блоковой командой. Не завершайте сеть ни командой сравнения, ни командой распознавания фронта (падающего или нарастающего).
- Вы не можете создавать разветвления, которые могут привести к изменению направления потока сигнала.



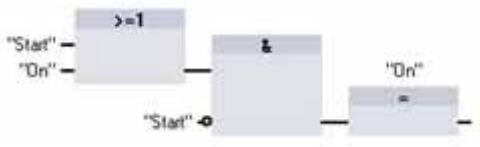
- Вы не можете создать разветвление, вызывающее короткое замыкание.



Язык программирования FBD

Как и LAD, FBD также является графическим языком программирования. Представление логики здесь основано на графических логических символах, используемых в булевой алгебре.

Математические и другие сложные функции могут быть представлены непосредственно в соединении с логическими блоками. Чтобы создать логику для сложных операций, вставляйте параллельные ветви между блоками.



Значение EN и ENO для блоковых команд

Как LAD, так и FBD, используют для некоторых блоковых команд понятие "поток сигнала" (EN и ENO). Некоторые команды (например, арифметические операции и команды пересылки) отображают параметры для EN и ENO. Эти параметры относятся к потоку сигнала и определяют, выполняется ли команда в этом цикле.

- EN (Enable In = разблокировать вход) является булевым входом для блоков в LAD и FBD. Поток сигнала (EN = 1) должен присутствовать на этом входе, чтобы блоковая команда выполнялась. Если вход EN блока LAD присоединен непосредственно к левой шине электропитания, то блок всегда будет исполняться.
- ENO (Enable Out = разблокировать выход) является булевым выходом для блоков в LAD и FBD. Если у блока имеется поток сигнала на входе EN и блок выполняет свою функцию без ошибок, то выход ENO передает поток сигнала (ENO = 1) следующему элементу. Если в исполнении блоковой команды обнаружена ошибка, то поток сигнала прерывается (ENO = 0) у блоковой команды, которая вызвала ошибку.

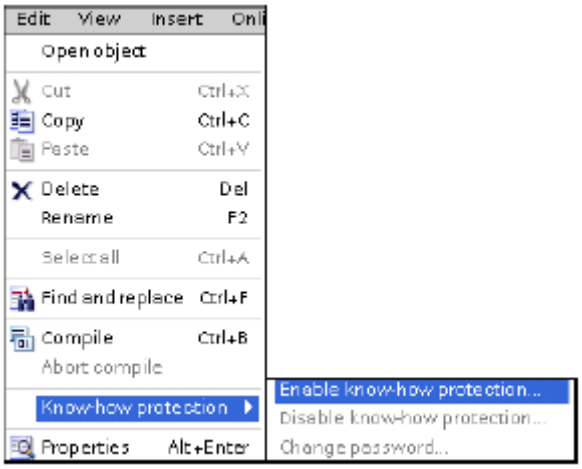
Программный редактор	Входы/выходы	Операнды	Тип данных
LAD	EN, ENO	Поток сигнала	BOOL
FBD	EN	I, I:P, Q, M, DB, Temp, Поток сигнала	BOOL
	ENO	Поток сигнала	BOOL

5.6 Защита от копирования

С помощью защиты от копирования или защиты "ноу-хау" вы можете защитить один или несколько кодовых блоков (OB, FB или FC) в своей программе от несанкционированного доступа. Для ограничения доступа к кодовому блоку вы можете ввести пароль.

Если вы сконфигурируете блок для защиты "ноу-хау", то код в этом блоке будет доступен только после ввода пароля.

Для создания защиты от копирования для блока выберите команду "Know how protection [Защита ноу-хау]" в меню "Edit [Редактирование]". Затем вы вводите пароль, который разрешает доступ к блоку.



Защита паролем предотвращает несанкционированное чтение или изменение кодового блока. Без пароля вы можете считывать только следующую информацию о кодовом блоке:

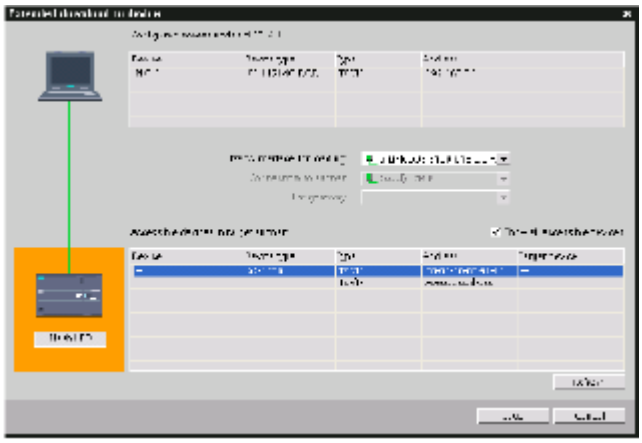
- Название, комментарий и свойства блока
- Параметры передачи (IN, OUT, IN_OUT, Return)
- Структура вызова программы
- Глобальные переменные в перекрестных ссылках (без информации о месте использования), однако локальные переменные скрыты

5.7 Загрузка элементов вашей программы в CPU

Вы можете загрузить элементы своего проекта из устройства программирования в CPU. При загрузке проекта CPU сохраняет программу пользователя (OB, FC, FB и DB) в постоянной памяти.

Вы можете загрузить свой проект из устройства программирования в CPU из любого из следующих мест:

- "Дерево проекта": Щелкните правой клавишей мыши на элементе программы, а затем выберите в контекстном меню пункт "Download [Загрузить]".
- Меню "Online": Щелкните на опции "Download to device [Загрузить в устройство]".
- Панель инструментов: Щелкните на символе "Загрузить в устройство ".



5.8 Загрузка элементов вашей программы из CPU

Вы можете загрузить все программные блоки и таблицу переменных из находящегося в режиме онлайн CPU в оффлайновый проект, но вы не сможете загрузить конфигурацию устройств и наблюдательные таблицы. Невозможна загрузка из онлайнового CPU в пустой проект; для этого вам нужен CPU, находящийся в режиме оффлайн. Вы не можете загрузить отдельный блок; загрузить можно только всю программу. Если вы загружаете данные из CPU, то оффлайновый CPU перед загрузкой и после вашего подтверждения "сбрасывается" (все блоки и таблица переменных удаляются). Вы не можете редактировать блок в онлайновой области; вы должны сначала загрузить его в офлайновую область, там вы можете его изменить, а затем загрузить обратно в ПЛК.

Загрузка из CPU возможна двумя способами: путем перетаскивания в дереве проекта или синхронизацией в редакторе сравнения.

Перетаскивание в дереве проекта

1. Создайте новый проект.
2. Добавьте CPU, который соответствует тому CPU, из которого выполняется загрузка.
3. Раскройте узел CPU так, чтобы стала видна папка "Program blocks [Программные блоки]".
4. В дереве проекта раскройте узел "Online access [Онлайновый доступ]", раскройте узел для желаемой сети и дважды щелкните на "Update accessible devices [Обновить доступные устройства]".
5. После того как на экран будут выведены имеющиеся CPU, раскройте узел для интересующего вас CPU.
6. Щелкните левой клавишей мыши в области "Online access [Онлайновый доступ]" на папке "Program blocks [Программные блоки]", удерживайте кнопку нажатой и перетащите эту папку к папке "Program blocks" в оффлайновой области, затем отпустите левую кнопку мыши. Указатель мыши должен превратиться в ‘+’, когда вы находитесь над правильной областью.
7. Теперь открывается диалоговое окно "Upload preview [Предварительный просмотр загрузки]". Щелкните в поле для "Continue [Продолжить]", а затем щелкните "Upload from device [Загрузить из устройства]".
8. Дайте загрузке закончиться. После этого в оффлайновой области отобразятся все программные блоки, технологические блоки и переменные.
9. Так как конфигурация устройств не может быть загружена из CPU, выполните настройку свойств CPU в конфигурации устройств вручную, включая желаемый IP-адрес, и вставьте другие устройства в оффлайновый проект.

Вы можете также перетаскивать с помощью мыши данные из онлайновой области в область "Program blocks [Программные блоки]" существующей программы. Т.е. оффлайновая область программных блоков не должна быть пустой. В этом случае существующая программа удаляется и заменяется онлайновой программой.

Синхронизация в редакторе сравнения

1. Откройте проект, содержащий программу.
2. В дереве проекта выберите оффлайновый CPU для сравнения.
3. Откройте редактор "Compare [Сравнение]", щелкнув правой клавишей мыши на оффлайновом CPU, или выбрав команду "Compare offline/online [Сравнить offline/online]" из меню "Tools [Инструменты]".
4. Редактор сравнения дает список различий в папке "Program blocks [Программные блоки]". Щелкните на символе в столбце "Action [Действие]". Чтобы загрузить проект из CPU, выберите "Upload from device [Загрузить из устройства]".
5. Щелкните на кнопке "Synchronize online and offline [Синхронизировать online и offline]", чтобы скопировать проект из онлайнового CPU в оффлайновый CPU.

5.9 Отладка и тестирование программы

Для контроля и изменения значений программы пользователя, исполняемой CPU, находящимся в режиме онлайн, можно использовать "наблюдательные таблицы". Вы можете создавать и сохранять в своем проекте различные наблюдательные таблицы для поддержки ряда сред тестирования. Это дает вам возможность воспроизводить тесты при вводе в эксплуатацию или для целей обслуживания и текущего ремонта.

С помощью наблюдательной таблицы вы можете контролировать, как CPU исполняет программу пользователя, и вмешиваться в этот исполнение. Вы можете отображать и изменять значения не только переменных кодовых блоков и блоков данных, но также и областей памяти CPU, включая входы и выходы (I и Q), периферийные входы и выходы (I:P и Q:P), битовую память (M) и блоки данных (DB).

С помощью наблюдательной таблицы вы можете разблокировать физические выходы (Q:P) CPU, находящегося в состоянии STOP. Например, вы можете назначать конкретные значения выходам при тестировании проводки для CPU.

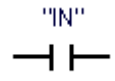
Наблюдательная таблица дает вам также возможность "форсировать" или устанавливать переменную на определенное значение. Дальнейшую информацию о принудительной установке ("форсировании") значений вы найдете в соответствующем разделе (стр. 323) в главе "Инструментальные средства онлайнового режима и диагностики ".

6 Руководство по программированию

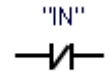
6.1 Основные команды

6.1.1 Двоичная логика

Контакты LAD



Замыкающий



Размыкающий

Вы можете соединять контакты друг с другом, создавая свою собственную комбинационную логику. Если указанный вами входной бит использует идентификатор памяти I (вход) или Q (выход), то значение этого бита считывается из регистра образа процесса. Физические сигналы с контактов в вашем управляющем процессе подключены к клеммам I на ПЛК. CPU опрашивает подключенные входные сигналы и непрерывно обновляет соответствующие значения состояний в образе процесса на входах.

Прямое считывание физического входа задается с помощью символов ":P", следующих за адресом входа (пример: "%I3.4:P"). При прямом считывании значения битовых данных считываются непосредственно с физического входа, а не из образа процесса. Прямое считывание не обновляет образа процесса.

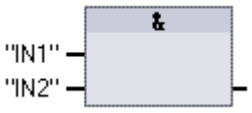
Параметр	Тип данных	Описание
IN	Bool	Назначенный бит

- Замыкающий (нормально открытый) контакт замкнут (ON), если значение назначенного бита равно 1.
- Размыкающий (нормально замкнутый) контакт замкнут (ON), если значение назначенного бита равно 0.
- Контакты, соединенные последовательно, образуют логические соединения типа И (AND).
- Контакты, соединенные параллельно, образуют логические соединения типа ИЛИ (OR).

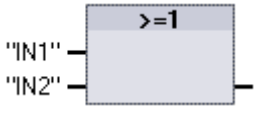
Блоки FBD, AND, OR и XOR

При программировании на языке FBD сети с LAD-контактами преобразуются в сети блоков И (AND, &), ИЛИ (OR, >=1) и исключающее ИЛИ (XOR, x), в которых вы можете задавать значения битов для входов и выходов блока. Вы можете также создавать соединения с другими логическими блоками, образуя свои собственные логические комбинации. После того как блок помещен в вашу сеть, вы можете перетащить инструментальное средство "Insert binary input [Вставить двоичный вход]" из панели инструментов "Favorites [Фавориты]" или из дерева команд к стороне входов блока, чтобы добавить дополнительные входы. Вы можете также щелкнуть правой клавишей мыши на стороне входов блока и выбрать "Insert input [Вставить вход]".

Входы и выходы блока могут быть соединены с другим логическим блоком, или вы можете ввести адрес бита или символическое имя бита для неподключенного входа. При выполнении блоковой команды текущие состояния входов прилагаются к двоичному входу логического блока и, если все верно, то выход блока тоже принимает значение истина.



Логическое соединение И (AND)



Логическое соединение ИЛИ (OR)

Логическое соединение исключающее ИЛИ (XOR)

Параметр	Тип данных	Описание
IN1, IN2	Bool	Входной бит

- Чтобы выход блока И (AND) принял значение ИСТИНА, на всех его входах должна быть ИСТИНА.
- Чтобы выход блока ИЛИ (OR) принял значение ИСТИНА, на любом его входе должна быть ИСТИНА.
- Чтобы выход блока исключающее ИЛИ (XOR) принял значение ИСТИНА, на нечетном числе его входов должна быть ИСТИНА.

Логическое отрицание NOT

При программировании на языке FBD вы можете перетащить инструментальное средство "Negate binary input [Инвертировать двоичный вход]" из панели инструментов "Favorites [Фавориты]" или из дерева команд на вход или выход, чтобы создать логическое отрицание для этого элемента блока.

LAD: инвертирующий контакт NOT

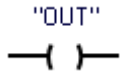
FBD: блок И (AND) с одним инвертированным логическим входом

FBD: блок И (AND) с инвертированным логическим входом и выходом

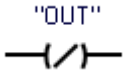
В LAD контакт NOT инвертирует логическое состояние входящего потока сигнала.

- Если на контакт NOT не поступает поток сигнала, то поток сигнала есть на его выходе.
- Если на контакт NOT поступает поток сигнала, то его нет на выходе.

Выходная катушка LAD



Выходная катушка



Инвертированная выходная катушка

Команда для выхода катушки реле записывает значение в выходной бит. Если указанный вами выходной бит использует идентификатор памяти Q, то CPU включает или выключает выходной бит в регистре образа процесса, устанавливая указанный бит равным состоянию потока сигнала. Выходные сигналы для исполнительных устройств вашего контроллера подключены к выходным клеммам S7-1200. В режиме RUN CPU непрерывно опрашивает входные сигналы вход, обрабатывает входные состояния в соответствии с логикой вашей программы, а затем реагирует на них, устанавливая новые значения для выходных состояний в регистре образа процесса на выходах. После каждого цикла исполнения программы CPU передает хранящуюся в регистре образа процесса новую реакцию на выходное состояние на подключенные выходные клеммы.

Вы можете задать прямую запись в физический выход с помощью символов ":P" после адреса Q (пример: "%Q3.4:P"). При прямой записи значения бита заносятся в выход образа процесса и непосредственно в физический выход.

Параметр	Тип данных	Описание
OUT	Bool	Назначенный бит

- Если имеется поток сигнала через выходную катушку, то выходной бит устанавливается в 1.
- Если нет потока сигнала через выходную катушку, то выходной бит устанавливается в 0.
- Если имеется поток сигнала через инвертированную выходную катушку, то выходной бит устанавливается в 0.
- Если нет потока сигнала через инвертированную выходную катушку, то выходной бит устанавливается в 1.

Блок FBD для назначения выхода

При программировании на языке FBD катушки LAD преобразуются в блоки назначений (= и /=), где вы указываете адрес бита для выхода блока. Входы и выходы блока могут быть соединены с другими логическими блоками, или вы можете ввести битовый адрес.

Назначение выхода

Инвертированное назначение выхода

Назначение выхода с его инверсией

Параметр	Тип данных	Описание
OUT	Bool	Назначенный бит

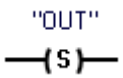
- Если вход выходного блока равен 1, то бит OUT устанавливается в 1.
- Если вход выходного блока равен 0, то бит OUT устанавливается в 0.
- Если вход инвертированного выходного блока равен 1, то бит OUT устанавливается в 0.
- Если вход инвертированного выходного блока равен 0, то бит OUT устанавливается в 1.

6.1.1.1 Команды установки и сброса

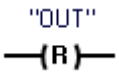
S и R: Установка и сброс 1 бита

- Если S (Set [Установить]) активизирован, то значение данных на адресе OUT устанавливается в 1. Если S не активизирован, то OUT не изменяется.
- Если R (Reset [Сбросить]) активизирован, то значение данных на адресе OUT устанавливается в 0. Если R не активизирован, то OUT не изменяется.
- Эти команды могут быть вставлены в любом месте сети.

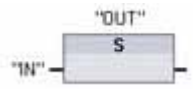
LAD: Установить



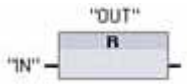
LAD: Сбросить



FBD: Установить



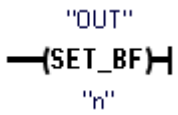
FBD: Сбросить



Параметр	Тип данных	Описание
IN (или соедините с контактной логикой или логикой устройства управления)	Bool	Адрес бита, подлежащего контролю
OUT	Bool	Адрес бита, подлежащего установке или сбросу

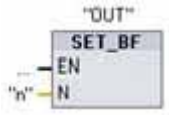
SET_BF и RESET_BF: Установка и сброс битового поля

LAD: SET_BF

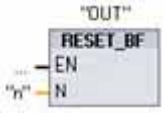


LAD: RESET_BF

FBD: SET_BF



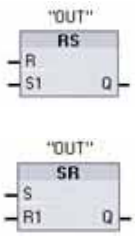
FBD: RESET_BF



Параметр	Тип данных	Описание
n	Константа	Число битов, подлежащих записи
OUT	Элемент булева массива	Начальный элемент булева массива, подлежащего установке или сбросу Пример: #MyArray[3]

- Если SET_BF активизирован, то значение 1 назначается "n" битам, начиная с адреса OUT. Если SET_BF не активизирован, то OUT не изменяется.
- RESET_BF записывает значение 0 в "n" битов, начиная с адреса OUT. Если RESET_BF не активизирован, то OUT не изменяется.
- Эти команды должны располагаться на самом правом краю ветви.

RS и SR: Триггер с преимуществом установки и с преимуществом сброса



RS – это триггер с преимуществом установки. Если сигналы установки (S1) и сброса (R) одновременно принимают значение истина, то выходной адрес OUT устанавливается в 1.

SR – это триггер с преимуществом сброса. Если сигналы установки (S) и сброса (R1) одновременно принимают значение истина, то выходной адрес OUT устанавливается в 0.

Параметр OUT указывает адрес бита, который устанавливается или сбрасывается. Не обязательно используемый выход Q отражает сигнальное состояние адреса "OUT".

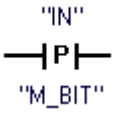
Параметр	Тип данных	Описание
S, S1	BOOL	Установить вход; 1 указывает на доминирование
R, R1	BOOL	Сбросить вход; 1 указывает на доминирование
OUT	BOOL	Назначенный бит выход "OUT"
Q	BOOL	Следует состоянию бита "OUT"

Команда	S1	R	Бит "OUT"
RS	0	0	Предыдущее состояние
	0	1	0
	1	0	1
	1	1	1
	S	**R1**	
SR	0	0	Предыдущее состояние
	0	1	0
	1	0	1
	1	1	0

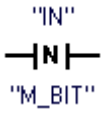
6.1.1.2 Команды распознавания нарастающего и падающего фронта

Распознавание нарастающего и падающего фронта

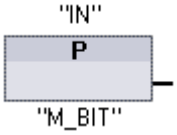
Контакт P: LAD



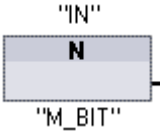
Контакт N: LAD



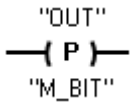
Блок P: FBD



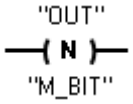
Блок N: FBD



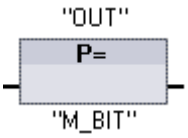
Катушка P: LAD



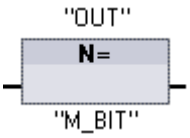
Катушка N: LAD



Блок P=: FBD



Блок N=: FBD



P_TRIG: LAD\FBD

N_TRIG: LAD\FBD

Параметр	Тип данных	Описание
M_BIT	Bool	Бит памяти, в котором сохраняется предыдущее состояние
IN	Bool	Входной бит, фронт которого необходимо распознать
OUT	Bool	Выходной бит, который указывает, что фронт был обнаружен
CLK	Bool	Поток сигнала или входной бит, фронт которого необходимо распознать
Q	Bool	Выход, который указывает, что фронт был обнаружен

Контакт P: LAD — Этот контакт принимает состояние ИСТИНА, когда на назначенном бите "IN" обнаруживается нарастающий фронт (переход из ВЫКЛ во ВКЛ). Логическое состояние этого контакта затем логически сопрягается с состоянием потока сигнала на входе, чтобы установить состояние потока сигнала на выходе. Контакт P может быть расположен в любом месте сети, кроме конца ветви.

Контакт N: LAD — Этот контакт принимает состояние ИСТИНА, когда на назначенном входном бите обнаруживается падающий фронт (переход из ВКЛ в ВЫКЛ). Логическое состояние этого контакта затем логически сопрягается с состоянием потока сигнала на входе, чтобы установить состояние потока сигнала на выходе. Контакт N может быть расположен в любом месте сети, кроме конца ветви.

Блок P: FBD — Выход принимает логическое состояние ИСТИНА, когда на назначенном входном бите обнаруживается нарастающий фронт (переход из ВЫКЛ во ВКЛ). Блок P может быть расположен только в начале ветви.

Блок N: FBD	Выход принимает логическое состояние ИСТИНА, когда на назначенном входном бите обнаруживается падающий фронт (переход из ВКЛ в ВЫКЛ). Блок N может быть расположен только в начале ветви.
Катушка P: LAD	Назначенный бит "OUT" принимает значение ИСТИНА, когда нарастающий фронт (переход из ВЫКЛ во ВКЛ) обнаружен в потоке сигнала, поступающего в катушку. Состояние потока сигнала на входе всегда проходит через катушку как состояние потока сигнала на выходе. Катушка P может находиться в любом месте сети.
Катушка N: LAD	Назначенный бит "OUT" принимает значение ИСТИНА, когда падающий фронт (переход из ВКЛ в ВЫКЛ) обнаружен в потоке сигнала, поступающего в катушку. Состояние потока сигнала на входе всегда проходит через катушку как состояние потока сигнала на выходе. Катушка N может находиться в любом месте сети.
Блок P=: FBD	Назначенный бит "OUT" принимает значение ИСТИНА, когда нарастающий фронт (переход из ВЫКЛ во ВКЛ) обнаружен в логическом состоянии входа блока или в назначенном входном бите, если блок находится в начале ветви. Логическое состояние на входе всегда проходит через блок как логическое состояние на выходе. Блок P= может находиться в любом месте ветви.
Блок N=: FBD	Назначенный бит "OUT" принимает значение ИСТИНА, когда падающий фронт (переход из ВКЛ в ВЫКЛ) обнаружен в логическом состоянии входа блока или в назначенном входном бите, если блок находится в начале ветви. Логическое состояние на входе всегда проходит через блок как логическое состояние на выходе. Блок N= может находиться в любом месте ветви .
P_TRIG: LAD/FBD	Поток сигнала или логическое состояние на выходе Q принимает значение ИСТИНА, когда нарастающий фронт (переход из ВЫКЛ во ВКЛ) обнаружен в состоянии на входе CLK (FBD) или в потоке сигнала на входе CLK (LAD). В LAD команда P_TRIG не может находиться в начале или в конце сети. В FBD команда P_TRIG может находиться в любом месте кроме конца ветви.
N_TRIG (LAD/FBD)	Поток сигнала или логическое состояние на выходе Q принимает значение ИСТИНА, когда падающий фронт (переход из ВКЛ в ВЫКЛ) обнаружен в состоянии на входе CLK (FBD) или в потоке сигнала на входе CLK (LAD). В LAD команда N_TRIG не может находиться в начале или в конце сети. В FBD команда N_TRIG может находиться в любом месте кроме конца ветви.

Все команды обнаружения фронта используют бит памяти (M_BIT) для хранения предыдущего состояния подлежащего контролю входного сигнала. Фронт обнаруживается путем сравнения состояния входа с состоянием этого бита памяти. Если эти состояния указывают на изменение сигнала на входе в интересующем нас направлении, то о появлении фронта сообщается установкой выхода в состояние ИСТИНА. Иначе выход устанавливается в состояние ЛОЖЬ.

Указание

Команды обнаружения фронтов анализируют значения входа и бита памяти при каждом исполнении команды, включая первое исполнение. Вы должны учитывать в своей программе начальные состояния входа и бит памяти, чтобы допускать или не допускать распознавание фронта в первом цикле.

Так как этот бит памяти должен сохраняться от одного исполнения команды до другого, то для каждой команды обнаружения фронта вы должны использовать уникальный бит, и вы не должны использовать этот бит ни в каком другом месте своей программы. Вам также следует избегать использования временной памяти, а также памяти, на которую могут оказывать влияние другие системные функции, например, обновление входов и выходов. Используйте для назначения бита M_BIT только битовую (M) память, глобальные DB или статическую память (в экземплярном DB).

6.1.2 Таймеры

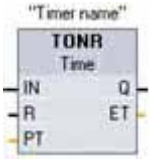
С помощью таймерных команд вы можете создавать программируемые запаздывания:

- TP: Импульсный таймер генерирует импульс заданной длительности.
- TON: Выход Q таймера с запаздыванием включения устанавливается в состояние ВКЛЮЧЕНО по истечении заранее заданного времени.
- TOF: Выход Q таймера с запаздыванием выключения устанавливается в состояние ВЫКЛЮЧЕНО по истечении заранее заданного времени.
- TONR: Выход запоминающего таймера с запаздыванием включения устанавливается в состояние ВКЛЮЧЕНО по истечении заранее заданного времени. Истекшее время накапливается в течение нескольких интервалов выдержки таймера, пока вход R не будет использован для сброса истекшего времени.
- RT: Сбрасывает таймер, стирая данные о времени, хранящиеся в заданном экземплярном блоке данных таймера.

Каждый таймер использует структуру, хранящуюся в блоке данных, для сохранения данных о времени. Вы назначаете блок данных, когда вы вставляете таймерную команду в редакторе.

Когда вы помещаете таймерные команды в функциональном блоке, вы можете выбрать вариант многоэкземплярного блока данных. Имена структур таймеров могут быть различными у различных структур данных, но данные таймеров находятся в одном единственном блоке данных и не требуют отдельного блока данных для каждого таймера. Это уменьшает время обработки и объем памяти данных, необходимой для управления таймерами. Между структурами данных таймеров в совместно используемом многоэкземплярном блоке данных нет взаимодействия.

Таймеры TP, TON и TOF имеют одни и те же входные и выходные параметры.



Таймер TONR имеет дополнительный параметр R для входа сброса.

Создайте свое собственное имя таймера ("Timer name") для обозначения блока данных таймера и описания назначения этого таймера в вашем процессе.

"Имя таймера"
----[RT]----

Команда RT сбрасывает данные указанного таймера.

Параметр	Тип данных	Описание
IN	Bool	Разблокирующий вход таймера
R	Bool	Сброс с ноль истекшего времени таймера TONR
PT	Bool	Вход предустановленного значения времени
Q	Bool	Выход таймера
ET	Time	Выход истекшего времени
Блок данных таймера	DB	Указывает, какой таймер должен быть сброшен командой RT

Параметр IN запускает и останавливает таймеры

- Переход с 0 на 1 параметра IN запускает таймеры TP, TON и TONR.
- Переход с 1 на 0 параметра IN запускает таймер TOF.

В следующей таблице показано влияние изменений значения в параметрах PT и IN.

Таймер	**Изменения в параметрах PT и IN**
TP	• Изменение PT не оказывает влияния во время работы таймера. • Изменение IN не оказывает влияния во время работы таймера .
TON	• Изменение PT не оказывает влияния во время работы таймера. • Изменение IN на ЛОЖЬ, когда таймер работает, сбрасывает и останавливает таймер.
TOF	• Изменение PT не оказывает влияния во время работы таймера. • Изменение IN на значение ИСТИНА, когда таймер работает, сбрасывает и останавливает таймер.
TONR	• Изменение PT не оказывает влияния во время работы таймера, но оказывает влияние, когда таймер возобновляет работу. • Изменение IN на ЛОЖЬ, когда таймер работает, останавливает таймер, но не сбрасывает его. Изменение IN обратно на значение ИСТИНА заставляет таймер работать, начиная с накопленного значения времени.

Значения TIME

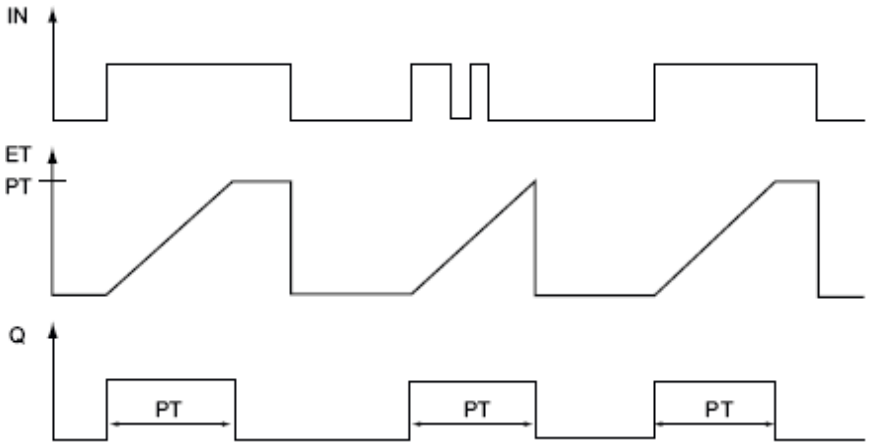
Значения PT (preset time [предустановленное время]) и ET (elapsed time [истекшее время]) хранятся в памяти как двойные целые со знаком, которые представляют миллисекунды. Тип данных TIME использует идентификатор T# и может быть введен как простая единица времени "T#200ms" или в виде комбинированных единиц времени "T#2s_200ms".

Тип данных	Размер	Допустимые диапазоны значений
TIME	32 бита Хранится как	от T#-24d_20h_31m_23s_648ms до T#24d_20h_31m_23s_647ms от -2 147 483 648 мс до +2 147 483 647 мс

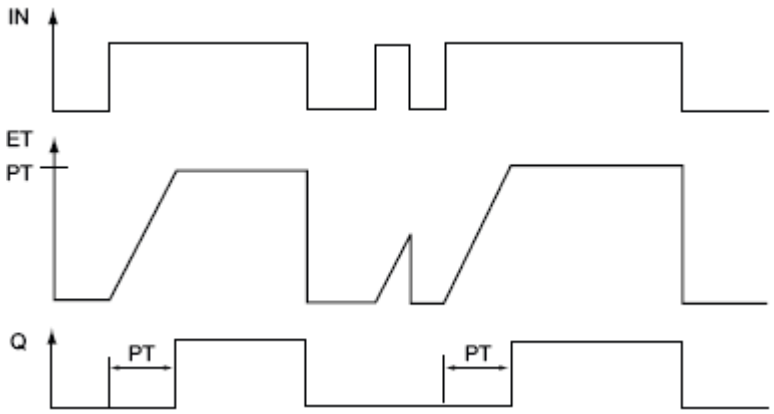
Указание

Отрицательный диапазон типа данных TIME, показанный выше, не может быть использован с таймерными командами. Отрицательные значения PT (предустановленное время) при исполнении таймерной команды сбрасываются в ноль. ET (истекшее время) всегда имеет положительное значение.

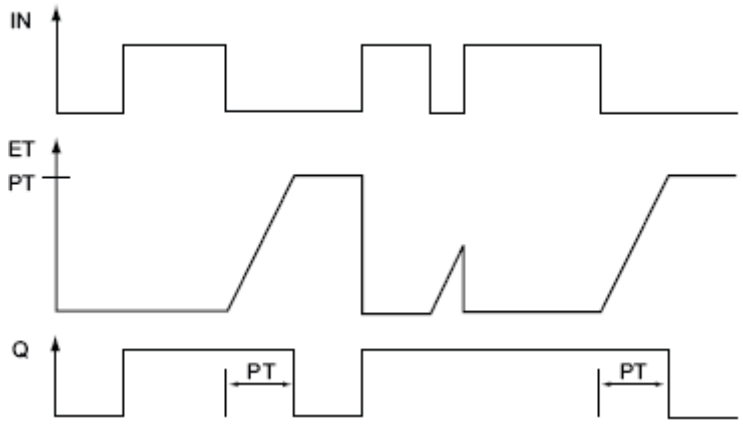
TP: Временная диаграмма Импульс



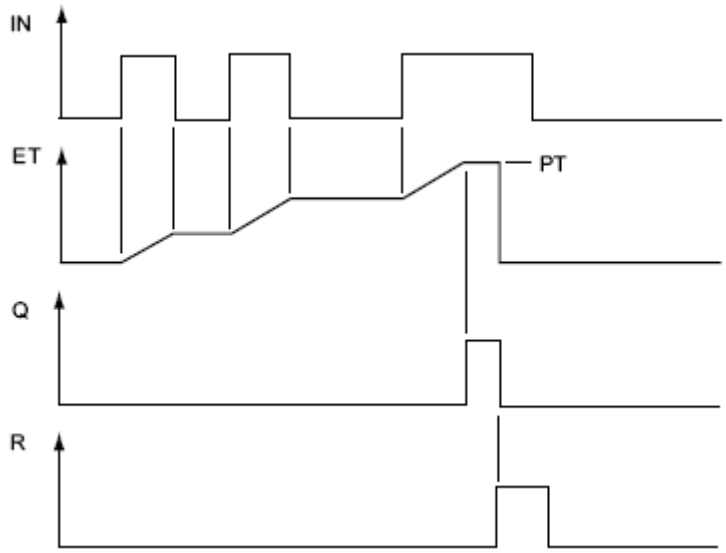
TON: Временная диаграмма Запаздывание включения



TOF: Временная диаграмма Запаздывание выключения



TONR: Временная диаграмма Запаздывание включения с запоминанием



6.1.3 Счетчики

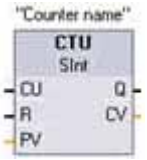
6.1.3.1 Счетчики

С помощью команд счета вы можете подсчитывать события внутри программы и внешние события в процессе:

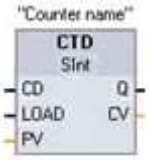
- CTU – это суммирующий счетчик.
- CTD – это вычитающий счетчик.
- CTUD – это реверсивный счетчик.

Каждый счетчик использует структуру, хранящуюся в блоке данных, для сохранения данных счета. Вы назначаете блок данных, когда вы помещаете счетчик в редактор. Эти команды используют программные счетчики, максимальная скорость счета которых ограничена скоростью исполнения OB, в котором они находятся. OB, в котором размещены эти команды, должен исполняться достаточно часто, чтобы обнаруживать все изменения входов CU или CD. Для более быстрого счета используйте команду CTRL_HSC.

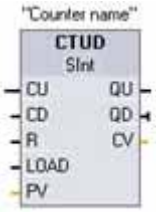
Когда вы помещаете команды счета в функциональный блок, вы можете выбрать вариант многоэкземплярного блока данных. Имена структур счетчиков могут быть различными у различных структур данных, но данные счетчиков находятся в одном единственном блоке данных и не требуют отдельного блока данных для каждого счетчика. Это уменьшает время обработки и объем памяти данных, необходимой для счетчиков. Между структурами данных счетчиков в совместно используемом многоэкземплярном блоке данных нет взаимодействия.



Выберите тип данных для значений счетчика из ниспадающего списка под именем блока.



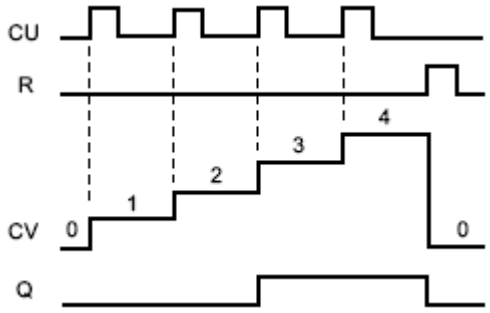
Создайте свое собственное имя счетчика ("Counter name") для обозначения блока данных счетчика и описания функции этого счетчика в вашем процессе.



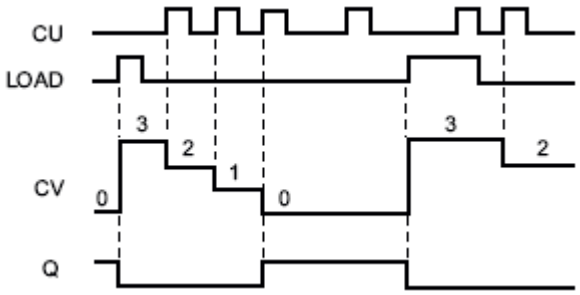
Параметр	Тип данных	Описание
CU, CD	Bool	Счет вверх или вниз, каждый раз на одну единицу
R (CTU, CTUD)	Bool	Сброс значения счетчика в ноль
LOAD (CTD, CTUD)	Bool	Управление загрузкой для предустановленного значения
PV	SInt, Int, DInt, USInt, UInt, UDInt	Предустановленное значение
Q, QU	Bool	Истина, если CV >= PV
QD	Bool	Истина, если CV <= 0
CV	SInt, Int, DInt, USInt, UInt, UDInt	Текущее значение счетчика

Числовой диапазон значений счетчика зависит от выбранного вами тира данных. Если значение счетчика – целое без знака, то вы можете считать в обратном направлении до нуля, а в прямом направлении до границы диапазона. Если значение счетчика – целое со знаком, вы можете считать в обратном направлении до нижней границы, а в прямом направлении до верхней границы.

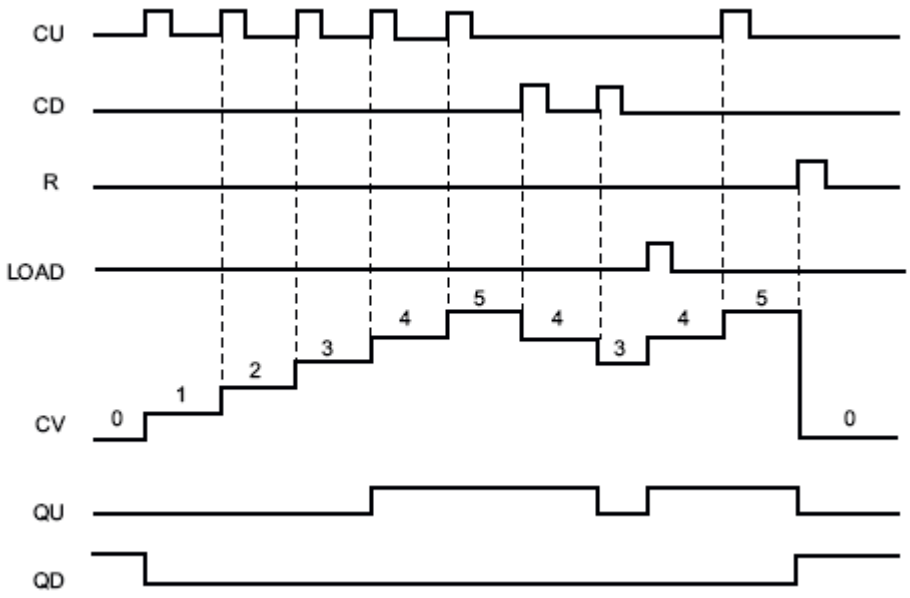
CTU: CTU увеличивает значение на 1, когда значение параметра CU изменяется с 0 на 1. Если значение параметра CV (текущее значение счетчика) больше или равно значению параметра PV (предустановленное значение счетчика), то выходной параметр счетчика
Q = 1. Если значение параметра сброса R изменяется с 0 на 1, то текущее значение счетчика сбрасывается в 0. На следующем рисунке показана временная диаграмма CTU со значением счетчика типа целое без знака (где PV = 3).



CTD: CTD уменьшает значение на 1, когда значение параметра CD изменяется с 0 на 1. Если значение параметра CV (текущее значение счетчика) меньше или равно 0, то выходной параметр счетчика Q = 1. Если значение параметра LOAD изменяется с 0 на 1, то значение параметра PV (предустановленное значение) загружается в счетчик как новое CV (текущее значение счетчика). На следующем рисунке показана временная диаграмма CTD со значением счетчика типа целое без знака (где PV = 3).



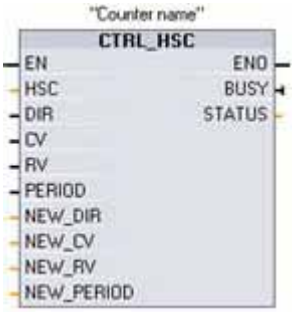
CTUD: CTUD увеличивает или уменьшает значение на 1, когда происходит переход с 0 на 1 соответственно на входе прямого (CU) или обратного (CD) счета. Если значение параметра CV (текущее значение счетчика) больше или равно значению параметра PV (предустановленное значение), то выходной параметр счетчика QU = 1. Если значение параметра CV меньше или равно нулю, то выходной параметр счетчика QD = 1. Если значение параметра LOAD изменяется с 0 на 1, то значение параметра PV (предустановленное значение) загружается в счетчик как новое CV (текущее значение счетчика). Если значение параметра сброса R изменяется с 0 на 1, то текущее значение счетчика сбрасывается в 0. На следующем рисунке показана временная диаграмма CTUD со значением счетчика типа целое без знака (где PV = 4).



6.1.3.2 Команда CTRL_HSC

Команда CTRL_HSC управляет скоростными счетчиками, которые используются для счета событий, происходящих чаще, чем частота исполнения OB. Частота выполнения операций счета командами CTU, CTD и CTUD ограничена частотой исполнения OB, в которых они находятся. Максимальные входные тактовые частоты для HSC вы найдете в технических данных CPU (стр. 335).

Типичным использованием скоростных счетчиков является счет импульсов, датчиком скорости вращения в системах управления перемещением.



Каждая команда CTRL_HSC использует для сохранения данных структуру, хранящуюся в блоке данных. Вы назначаете блок данных, когда команда CTRL_HSC вставляется в редактор.

Создайте свое собственное имя счетчика ("Counter name") для обозначения блока данных счетчика и описания функции этого счетчика в вашем процессе.

Параметр	Тип параметра	Тип данных	Описание
HSC	IN	HW_HSC	Идентификатор HSC
DIR	IN	Bool	1 = Запросить новое направление
CV	IN	Bool	1 = Запрос на установку нового значения счетчика
RV	IN	Bool	1= Запрос на установку нового эталонного значения
PERIOD	IN	Bool	1 = Запрос на установку нового интервала времени (только при измерении частоты)
NEW_DIR	IN	Int	Новое направление: 1= вперед -1= назад
NEW_CV	IN	DInt	Новое направление счета
NEW_RV	IN	DInt	Новое эталонное значение
NEW_PERIOD	IN	Int	Новое значение интервала времени в секундах:0,01; 0,1 или 1 (только при измерении частоты)
BUSY	OUT	Bool	Функция занята
STATUS	OUT	Word	Код условия исполнения

Прежде чем скоростные счетчики можно будет использовать в программе, вы должны их создать в настройках проекта для конфигурации устройств ПЛК. При конфигурировании устройств HSC устанавливаются режимы счета, интерфейсы ввода/вывода, назначение прерываний и работа в качестве скоростного счетчика или в качестве устройства для измерения частоты импульсов. Скоростной счетчик может функционировать с программным управлением или без него.

Многие конфигурационные параметры скоростных счетчиков устанавливаются только в конфигурации устройств проекта. Некоторые параметры скоростных счетчиков инициализируются в конфигурации устройств проекта, но позже могут быть изменены под управлением программы.

Параметры команды CTRL_HSC обеспечивают программное управление процессом счета:

- Установка направления счета на новое значение NEW_DIR
- Установка текущего значения счетчика на новое значение NEW_CV
- Установка эталонного значения на новое значение NEW_RV
- Установка значения интервала времени (для режима измерения частоты) на новое значение NEW_PERIOD

Если при исполнении команды CTRL_HSC следующие биты установлены в 1, то соответствующее значение NEW_xxx загружается в счетчик. Несколько запросов (более одного бита установлено одновременно) обрабатываются за одно исполнение команды CTRL_HSC.

- DIR = 1 – это запрос на загрузку значения NEW_DIR, 0 = нет изменения
- CV = 1 – это запрос на загрузку значения NEW_CV, 0 = нет изменения
- RV = 1 – это запрос на загрузку значения NEW_RV, 0 = нет изменения
- PERIOD = 1 – это запрос на загрузку значения NEW_PERIOD, 0 = нет изменения

Команда CTRL_HSC обычно помещается в OB аппаратных прерываний, который исполняется, когда запускается аппаратное прерывание, связанное со счетчиком. Например, если событие CV=RV запускает прерывание, связанное со счетчиком, то кодовый блок OB аппаратных прерываний исполняет команду CTRL_HSC и может изменить эталонное значение, загрузив значение NEW_RV.

Текущее значение счетчика отсутствует среди параметров CTRL_HSC. Адрес образа процесса, в котором сохраняется текущее значение счетчика, назначается при конфигурировании счетчика. Вы можете непосредственно считывать значение счетчика с помощью программной логики, и значение, возвращаемое в вашу программу, будет правильным значением счетчика для того момента, когда счетчик считывался. Но счетчик продолжает счет быстрых событий. Поэтому текущее значение счетчика может измениться, прежде чем ваша программа завершит процесс со старым значением.

Подробности для параметров CTRL_HSC:

- Если нет запроса на обновление параметра, то соответствующие входные значения игнорируются.
- Параметр DIR действителен только тогда, когда для сконфигурированного направления счета задано программное управление "User program (internal direction control [Программа пользователя (внутреннее управление направлением счета)])". Вы задаете способ использования этого параметра в конфигурации HSC.
- Для HSC S7-1200 в CPU или на сигнальной плате параметр BUSY всегда имеет значение 0.

Коды условий: В случае ошибки ENO устанавливается в 0, а выход STATUS содержит код условия.

Значение STATUS (W#16#...)	Описание
0	Нет ошибки
80A1	Идентификатор HSC не обращается к HSC
80B1	Недопустимое значение в NEW_DIR
80B2	Недопустимое значение в NEW_CV
80B3	Недопустимое значение в NEW_RV
80B4	Недопустимое значение в NEW_PERIOD

6.1.3.3 Принцип действия скоростных счетчиков

Скоростной счетчик (HSC) может быть использован как вход для углового шагового датчика. Угловой шаговый датчик делает определенное число отсчетов на оборот, а также по одному импульсу сброса на каждый оборот. Генератор или генераторы тактовых импульсов и импульс сброса от углового шагового датчика поставляют входную информацию для HSC.

HSC загружается первым из нескольких предустановленных значений, а выходы активизируется в течение интервала времени, когда текущее значение счетчика меньше текущего предустановленного значения. HSC запускает прерывание, когда текущее значение счетчика равно предустановленному, когда происходит сброс, а также когда меняется направление.

Когда текущее значение счетчика становится равным предустановленному значению, вызывая прерывание, в счетчик загружается новое предустановленное значение, а для выходов устанавливается следующее состояние. Когда прерывание вызывается сбросом, то загружается первое предустановленное значение и устанавливаются первые состояния выходов, и цикл повторяется.

Так как частота прерываний значительно меньше частоты счета HSC, то может быть реализовано точное управление скоростными операциями при относительно малом влиянии на цикл CPU. Метод закрепления прерываний позволяет каждую загрузку нового предустановленного значения выполнять в отдельной программе прерываний, облегчая управление состоянием. (В качестве альтернативы вы можете все прерывания обрабатывать в одной программе прерываний.)

Выбор функционального назначения для HSC

Все HSC при одинаковом режиме счета работают одинаково. Имеются четыре основных типа быстрых счетчиков:

- Однофазный счетчик с внутренним управлением направлением
- Однофазный счетчик с внешним управлением направлением
- Двухфазный счетчик с 2 тактовыми входами
- Квадратурный A/B-счетчик

HSC любого типа можно использовать с входом сброса или без него. При активизации входа сброса (с некоторыми ограничениями, см. следующую таблицу) текущее значение сбрасывается и остается сброшенным до деактивизации вход сброса.

- Функция частоты: Некоторые режимы HSC допускают такое конфигурирование HSC (Type of counting [Тип счета]), чтобы счетчик выдавал частоту вместо текущего числа импульсов. Имеются три различных интервала измерения частоты: 0,01; 0,1 или 1,0 секунда.

  Интервал измерения частоты определяет, как часто HSC вычисляет и выдает новое значение частоты. Выданная частота является средним значением, определяемым общим числом отсчетов за последний интервал измерения. Если частота изменяется быстро, то выданное значение будет промежуточной величиной между максимальной и минимальной частотой на интервале измерения. Значение частоты всегда сообщается в герцах (импульсах в секунду) независимо от установленного интервала измерения частоты.

- Режимы работы и входы счетчика: В следующей таблице показаны входы, используемые для таких функций, как генератор тактовых импульсов, управление направлением и сброс HSC.

  Один и тот же вход не может быть использован для двух разных функций, но любой вход, не используемый текущим режимом работы HSC, может быть использован для другой цели. Например, если HSC1 находится в режиме, который использует встроенные входы, но не использует внешний сброс (I0.3), то I0.3 можно использовать для прерываний по обнаружению фронта или для HSC2.

Описание			**Назначение входа по умолчанию**			**Функция**
HSC	HSC1	Встроенные или сигнальная плата или контроль PTO 0[1]	I0.0 I4.0 PTO 0 Импульс	I0.1 I4.1 PTO 0 Направление	I0.3 I4.3 -	
	HSC:	Встроенные или сигнальная плата или контроль PTO 1[1]	I0.2 I4.2 PTO 1 Импульс	I0.3 I4.3 PTO 1 Направление	I0.1 I4.1 -	
	HSC3[2]	Встроенные	I0.4	I0.5	I0.7	
	HSC4[3]	Встроенные	I0.6	I0.7	I0.5	
	HSC5[4]	Встроенные или сигнальная плата	I1.0 I4.0	I1.1 I4.1	I1.2 I4.3	
	HSC6[4]	Встроенные или сигнальная плата	I1.3 I4.2	I1.4 I4.3	I1.5 I4.1	

Описание		Назначение входа по умолчанию			Функция
Режим	Однофазный счетчик с внутренним управлением направлением	Такт	-	-	Счет или частота
				Сброс	Счет
	Однофазный счетчик с внешним управлением направлением	Такт	Направление	-	Счет или частота
				Сброс	Счет
	Двухтактный счетчик с 2 тактовыми входами	Такт вперед	Такт назад	-	Счет или частота
				Сброс	Счет
	Квадратурный A/B-фазный счетчик	Фаза A	Фаза B	-	Счет или частота
				Фаза Z	Счет
	Контроль выходов последовательности импульсов (PTO)[1]	Такт	Направление	-	Счет

[1] Контроль выходов последовательности импульсов всегда использует генератор тактовых импульсов и направление. Если соответствующий выход PTO сконфигурирован только для импульсов, то выход направления, как правило, следует использовать для счета вперед.

[2] HSC3 с входом сброса невозможен для CPU 1211C, который поддерживает только 6 встроенных входов.

[3] HSC4 невозможен для CPU 1211C, который поддерживает только 6 встроенных входов.

[4] HSC5 и HSC6 поддерживаются CPU 1211C и CPU 1212C только при установке сигнальной платы.

Обращение к текущему значению HSC

CPU сохраняет текущее значение каждого HSC в адресе входа (I). В следующей таблице показаны адреса, назначенные по умолчанию текущему значению каждого HSC. Вы можете изменить I-адреса для текущего значения, изменив свойства CPU в конфигурации устройств.

Скоростной счетчик	Тип данных	Адрес по умолчанию
HSC1	DInt	ID1000
HSC2	DInt	ID1004
HSC3	DInt	ID1008
HSC4	DInt	ID1012
HSC5	DInt	ID1016
HSC6	DInt	ID1020

Значения цифровых входов/выходов, назначенных HSC, не могут быть принудительно изменены

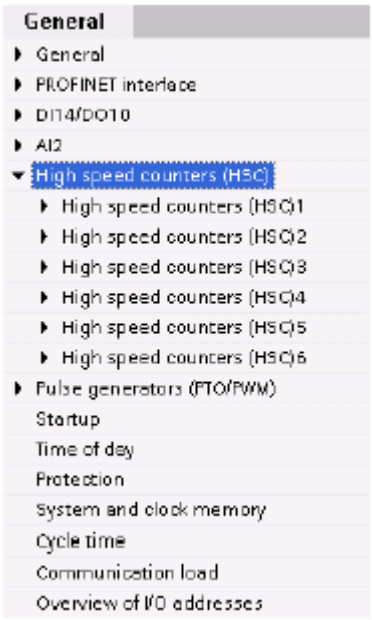
Цифровые входы и выходы, используемые скоростными счетчиками, назначаются при конфигурировании устройств. Если адреса входов/выходов назначены этим устройствам, то значения этих назначенных адресов не могут быть изменены функцией принудительного присваивания значений в таблице наблюдения.

6.1.3.4 Конфигурирование скоростного счетчика

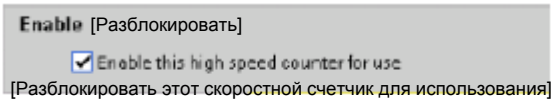
CPU дает вам возможность сконфигурировать до 6 скоростных счетчиков. Конфигурирование параметров каждого отдельного HSC осуществляется через свойства ("Properties") CPU.

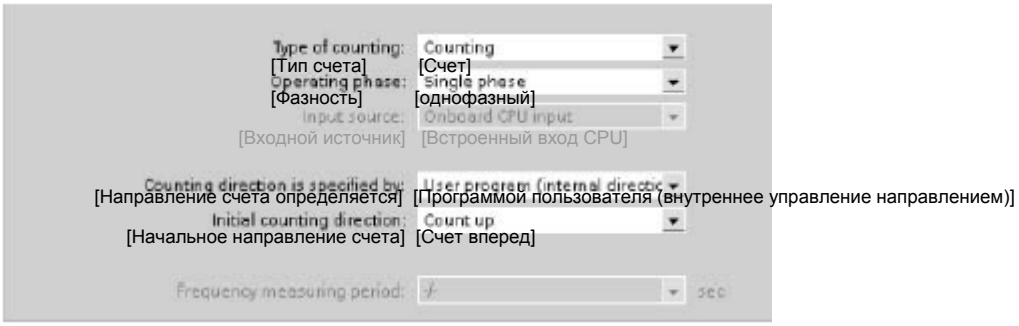
После активизации HSC сконфигурируйте другие параметры, например, функцию счетчика, начальные значения, возможности сброса и прерывающие события.

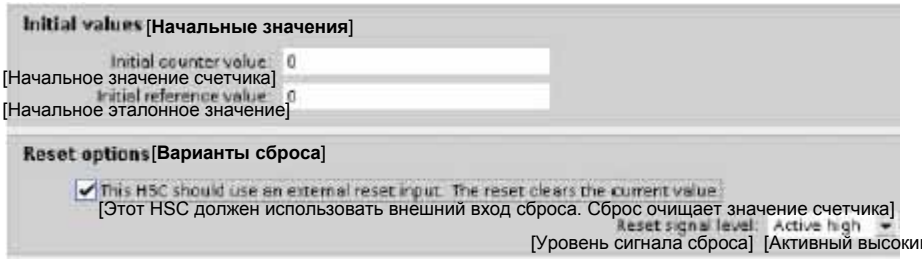
После конфигурирования HSC вы можете с помощью команды CTRL_HSC управлять работой HSC в своей пользовательской программе.

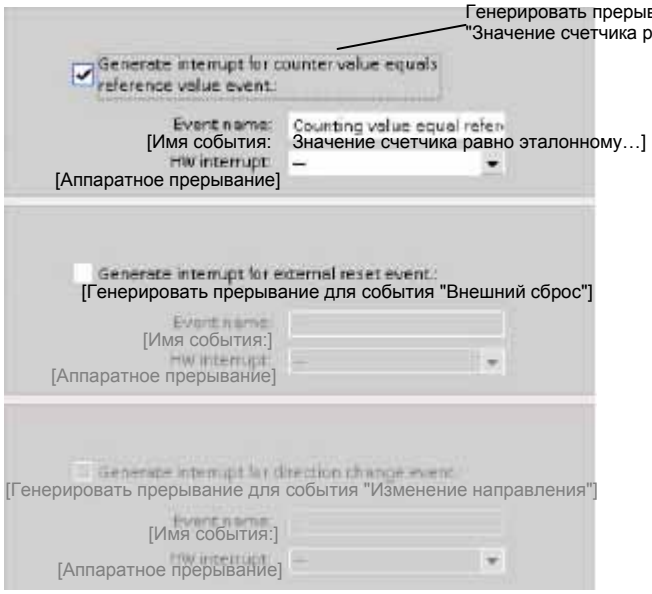


Пояснение к рисунку: High speed counters – Скоростные счетчики;

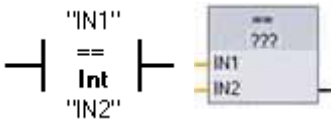








6.1.4 Сравнение



LAD **FBD**

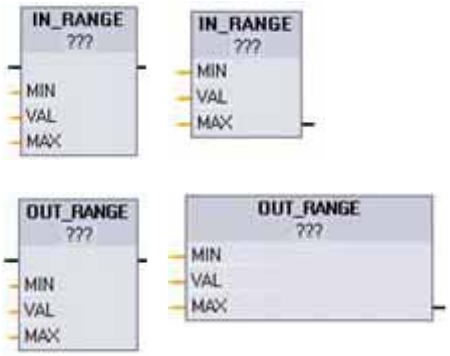
Команды сравнения используются для сравнения двух величин, относящихся к одному и тому же типу данных. Если сравнение в виде контакта в LAD имеет значение ИСТИНА, то контакт активизирован. Если сравнение в виде блока в FBD имеет значение ИСТИНА, то выход блока имеет значение ИСТИНА.

Щелкнув на команде в программном редакторе, вы можете выбрать тип сравнения и тип данных из ниспадающих меню.

Тип отношения	Сравнение истинно, если:
==	IN1 равно IN2
<>	IN1 не равно IN2
>=	IN1 больше или равноIN2
<=	IN1 меньше или равно IN2
>	IN1 больше, чем IN2
<	IN1 меньше, чем IN2

Параметр	Тип данных	Описание
IN1, IN2	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, String, Char, Time, DTL, Constant	Величины, подлежащие сравнению

Команды IN_RANGE и OUT_RANGE



LAD **FBD**

С помощью команд IN_RANGE и OUT_RANGE вы можете проверить, находится ли входное значение внутри или вне заданного диапазона значений. Если результатом сравнения является ИСТИНА, то выход блока принимает значение ИСТИНА.

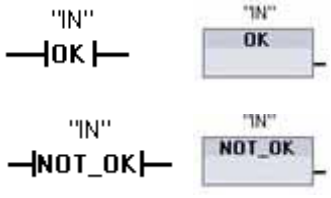
Входные параметры MIN, VAL и MAX должны относиться к одному и тому же типу данных.

Щелкнув на команде в программном редакторе, вы можете выбрать тип данных из ниспадающих меню.

Тип отношения	Результатом сравнения является ИСТИНА, если:
IN_RANGE	MIN <= VAL <= MAX
OUT_RANGE	VAL < MIN или VAL > MAX

Параметр	Тип данных	Описание
MIN, VAL, MAX	SInt, Int, DInt, USInt, UInt, UDInt, Real, Constant	Входы компаратора

Команды OK и Not OK



LAD **FBD**

С помощью команд OK и NOT_OK можно проверить, действительно ли является эталонное значение входных данных вещественным числом в соответствии со спецификацией IEEE 754. Если контакт в LAD принимает значение ИСТИНА, то он активизирован и пропускает через себя поток сигнала. Если блок FBD принимает значение ИСТИНА, то выход блока тоже принимает значение ИСТИНА.

Значение типа Real или LReal является недопустимым, если оно равно +/- INF (бесконечность), NaN (Not a Number [Не число]), или оно денормализовано. Денормализованное число – это число, очень близкое к нулю. При расчетах CPU вместо денормализованного числа подставляет ноль.

Команда	Проверка, является ли число вещественным, принимает значение ИСТИНА, если:
OK	Входная величина действительно является вещественным числом
NOT_OK	Входная величина не является вещественным числом

Параметр	Тип данных	Описание
IN	Real, LReal	Входные данные

6.1.5 Арифметические команды

Команды сложения, вычитания, умножения и деления

Блоковые арифметические команды используются для программирования основных арифметических операций:

- ADD: Сложение (IN1 + IN2 = OUT)
- SUB: Вычитание (IN1 - IN2 = OUT)
- MUL: Умножение (IN1 * IN2 = OUT)
- DIV: Деление (IN1 / IN2 = OUT)

  При целочисленном делении дробная часть частного отбрасывается, что приводит к появлению целочисленного выходного значения.

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Указание

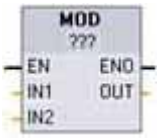
Параметры основных арифметических команд IN1, IN2 и OUT должны относиться к одному и тому же типу данных.

Параметр	Тип данных	Описание
IN1, IN2	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Constant	Входы арифметических команд
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Выход арифметических команд

Если арифметическая команда активизирована (EN = 1), то она выполняет указанную операцию над входными значениями (IN1 и IN2) и сохраняет результат по адресу, указанному в выходном параметре (OUT). После успешного выполнения операции команда устанавливает ENO = 1.

Состояние ENO	Описание
1	Нет ошибки
0	Результирующе значение арифметической операции находится вне допустимого диапазона значений для выбранного типа данных. Возвращается наименьшая значащая часть результата, которая подходит по размеру целевой величине.
0	Деление на 0 (IN2 = 0): Результат неопределен, и возвращается ноль.
0	Real/LReal: Если одна из входных величин является NaN (не число), то возвращается NaN.
0	ADD Real/LReal: Если обе входных величины (IN) равны бесконечности (INF) с разными знаками, то эта операция недопустима, и возвращается NaN.
0	SUB Real/LReal: Если обе входных величины (IN) равны бесконечности (INF) с одинаковым знаком, то эта операция недопустима, и возвращается NaN.
0	MUL Real/LReal: Если одна из входных величин (IN) равна нулю, а другая INF, то эта операция недопустима и возвращается NaN.
0	DIV Real/LReal: Если обе входных величины (IN) равны нулю или INF, то эта операция недопустима и возвращается NaN.

6.1.5.1 Команда MOD (получение остатка от деления)



Команда MOD (modulo) используется для выполнения операции IN1 modulo IN2.
Операция IN1 MOD IN2 = IN1 - (IN1 / IN2)*IN2 = параметр OUT.
Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Указание

Параметры IN1, IN2 и OUT должны относиться к одному и тому же типу данных.

Параметр	Тип данных	Описание
IN1 и IN2	Int, DInt, USInt, UInt, UDInt, Constant	Входы блока
OUT	Int, DInt, USInt, UInt, UDInt	Выход блока

Состояние ENO	Описание
1	Нет ошибки
0	Значение IN2 = 0, параметру OUT присваивается значение ноль

Команда NEG

Команда NEG (отрицание) используется для изменения знака параметра IN и сохранения результата в параметре OUT.

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Указание

Параметры IN и OUT должны относиться к одному и тому же типу данных.

Параметр	Тип данных	Описание
IN	SInt, Int, DInt, Real, LReal, Constant	Вход арифметической команды
OUT	SInt, Int, DInt, Real, LReal	Выход арифметической команды

Состояние ENO	Описание
1	Нет ошибки
0	Результирующая величина находится за пределами допустимого диапазона для выбранного типа данных. Пример для SInt: NEG (-128) дает +128, что превышает максимальное значение для этого типа данных.

Команды увеличения и уменьшения на 1

Команды INC и DEC используются для:

- увеличения на 1 целого числа со знаком или без знака

  INC (увеличение на 1): параметру IN/OUT присваивается значение IN/OUT + 1
- уменьшения на 1 целого числа со знаком или без знака

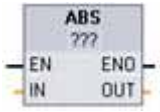
  DEC (уменьшение на 1): параметру IN/OUT присваивается значение IN/OUT – 1

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Параметр	Тип данных	Описание
IN/OUT	SInt, Int, DInt, USInt, UInt, UDInt	Вход и выход арифметической команды

Состояние ENO	Описание
1	Нет ошибки
0	Результирующая величина находится за пределами допустимого диапазона для выбранного типа данных. Пример для SInt: INC (127) дает -128, что выходит за пределы допустимого диапазона для этого типа данных.

Команда образования абсолютного значения



Команда ABS используется для получения абсолютного значения целого или вещественного числа со знаком для параметра IN и сохранения результата в параметре OUT.

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

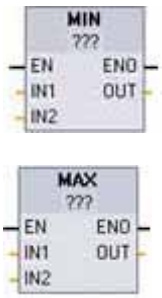
Указание

Параметры IN и OUT должны относиться к одному и тому же типу данных.

Параметр	Тип данных	Описание
IN	SInt, Int, DInt, Real, LReal	Вход арифметической команды
OUT	SInt, Int, DInt, Real, LReal	Выход арифметической команды

Состояние ENO	Описание
1	Нет ошибки
0	Результирующее значение арифметической операции находится за пределами допустимого диапазона для выбранного типа данных. Пример для SInt: ABS (-128) дает +128, что выходит за пределы допустимого диапазона для этого типа данных.

Команды MIN и MAX



Команды MIN (минимум) и MAX (максимум) используются следующим образом:

- MIN сравнивает значения двух параметров IN1 и IN2 и присваивает минимальное (меньшее) значение параметру OUT.
- MAX сравнивает значения двух параметров IN1 и IN2 и присваивает максимальное (большее) значение параметру OUT.

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Указание

Параметры IN1, IN2 и OUT должны относиться к одному и тому же типу данных.

Параметр	Тип данных	Описание
IN1, IN2	SInt, Int, DInt, USInt, UInt, UDInt, Real, Constant	Входы арифметической команды
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real	Выход арифметической команды

Состояние ENO	Описание
1	Нет ошибки
0	Только для типа данных Real: • Один или оба входа не являются вещественными числами (NaN). • Результирующий выход OUT равен +/- INF (бесконечность).

Команда проверки граничных значений

С помощью команды LIMIT вы можете проверить, находится ли значение параметра IN внутри допустимого диапазона, определяемого параметрами MIN и MAX. Значение OUT фиксируется на значении MIN или MAX, если значение IN находится вне этого диапазона.

- Если значение параметра IN находится внутри заданного диапазона, то значение IN сохраняется в параметре OUT.
- Если значение параметра IN находится вне заданного диапазона, то значению OUT присваивается значение параметра MIN (если значение IN меньше, чем значение MIN) или значение параметра MAX (если значение IN больше, чем значение MAX).

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Указание

Параметры MIN, IN, MAX и OUT должны относиться к одному и тому же типу данных.

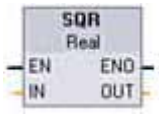
Параметр	Тип данных	Описание
MIN, IN и MAX	SInt, Int, DInt, USInt, UInt, UDInt, Real, Constant	Входы арифметической команды
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real	Выход арифметической команды

Состояние ENO	Описание
1	Нет ошибки
0	Real: Если одно или более значений для MIN, IN и MAX равно NaN (не число), то возвращается NaN.
0	Если MIN больше, чем MAX, значение IN присваивается выходу OUT.

Арифметические операции с плавающей точкой

Операции с плавающей точкой используются для программирования арифметических функций с типом данных Real или LReal:

- SQR: Квадрат (IN ^{2} = OUT)
- SQRT: Квадратный корень (√IN = OUT)
- LN: Натуральный логарифм (LN(IN) = OUT)
- EXP: Натуральная экспоненциальная функция (e ^{IN} =OUT), где основание e = 2.71828182845904523536
- SIN: Синус (sin(IN радиан) = OUT)
- COS: Cosine (cos(IN радиан) = OUT)
- TAN: Тангенс (tg(IN радиан) = OUT)
- ASIN: Арксинус (arcsin(IN) = OUT радиан), где sin(OUT радиан) = IN
- ACOS: Арккосинус (arccos(IN) = OUT радиан), где cos(OUT радиан) = IN
- ATAN: Арктангенс (arctg(IN) = OUT радиан), где tg(OUT радиан) = IN
- FRAC: Дробная часть (разряды после десятичной точки в числе с плавающей точкой IN = OUT)
- EXPT: Возведение в степень (IN1 ^{IN2} = OUT)



Щелкните под именем блока и выберите тип данных из ниспадающего меню. Параметры EXPT IN1 и OUT всегда являются вещественными числами. Для параметра экспоненты IN2 вы можете выбрать тип данных.

Параметр	Тип данных	Описание
IN, IN1	Real, LReal, Constant	Входы
IN2	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Constant	Вход EXPT
OUT	Real, LReal	Выходы

Состояние ENO	Команда	Условие	Результат (OUT)
1	Все	Нет ошибки	Допустимый результат
0	SQR	Результат выходит за пределы допустимых значений для Real/LReal	+INF
		IN равняется +/- NaN (не число)	+NaN
	SQRT	IN отрицательно	-NaN
		IN равно +/- INF (бесконечность) или +/- NaN	+/- INF или +/- NaN
	LN	IN равно 0.0, отрицательно, -INF или -NaN	-NaN
		IN равно +INF или +NaN	+INF или +NaN
	EXP	Результат выходит за пределы допустимых значений для Real/LReal	+INF
		IN равно +/- NaN	+/- NaN
	SIN, COS, TAN	IN равно +/- INF или +/- NaN	+/- INF или +/- NaN
	ASIN, ACOS	IN выходит за пределы допустимого диапазона от -1.0 до +1.0	+NaN
		IN равно +/- NaN	+/- NaN
	ATAN	IN равно +/- NaN	+/- NaN
	FRAC	IN равно +/- INF или +/- NaN	+NaN
	EXPT	IN1 равно +INF, а IN2 не равно -INF	+INF
		IN1 отрицательно или -INF	+NaN, если IN2 имеет тип Real/LReal, -INF в противном случае
		IN1 или IN2 равно +/- NaN	+NaN
		IN1 равно 0.0, а IN2 имеет тип Real/LReal (только)	+NaN

6.1.6 Команда Move

Команды передачи и блочной передачи

Команды передачи используются для копирования элементов данных в новый адрес в памяти и преобразования из одного типа данных в другой. При этом источник данных не изменяется.

- MOVE: Копирует элемент данных, хранящийся по определенному адресу в новый адрес
- MOVE_BLK: Прерываемая передача, которая копирует блок элементов данных в новый адрес
- UMOVE_BLK: Непрерываемая передача, которая копирует блок элементов данных в новый адрес

MOVE		
Параметр	**Тип данных**	**Описание**
IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Char, Array, Struct, DTL, Time	Адрес источника
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Char, Array, Struct, DTL, Time	Целевой адрес

MOVE_BLK, UMOVE_BLK		
Параметр	**Тип данных**	**Описание**
IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, Byte, Word, DWord	Начальный адрес источника
COUNT	UInt	Число элементов данных, подлежащих копированию
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, Byte, Word, DWord	Начальный адрес назначения

Указание

Правила для операций копирования данных

- Для копирования данных типа Bool используйте SET_BF, RESET_BF, R, S или выходную катушку (LAD)
- Для копирования отдельного элементарного типа данных используйте MOVE
- Для копирования массива данных элементарного типа используйте MOVE_BLK или UMOVE_BLK
- Для копирования структуры используйте MOVE
- Для копирования строки используйте S_CONV
- Для копирования отдельного символа в строке используйте MOVE
- Команды MOVE_BLK и UMOVE_BLK не могут использоваться для копирования массивов или структур в области памяти I, Q или M.

Команда MOVE копирует отдельный элемент данных из исходного адреса, указанного в параметре IN, в целевой адрес, определяемый параметром OUT.

Команды MOVE_BLK и UMOVE_BLK имеют дополнительный параметр COUNT. COUNT определяет, сколько элементов данных копируется. Число байтов на копируемый элемент зависит от типа данных, назначенных именам переменных параметров IN и OUT в таблице переменных ПЛК.

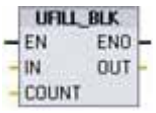
Команды MOVE_BLK и UMOVE_BLK отличаются друг от друга способом обработки прерываний:

- Во время исполнения MOVE_BLK прерывающие события **ставятся в очередь и обрабатываются**. Используйте команду MOVE_BLK, если данные по адресу назначения передачи не используются в подпрограмме внутри OB прерываний или, если используются, то целевые данные не обязательно должны быть согласованными. Если операция MOVE_BLK прерывается, то последний переданный элемент данных полностью и непротиворечиво находится по целевому адресу. Операция MOVE_BLK возобновляется по окончании исполнения OB прерываний.
- Прерывания **ставятся в очередь, но не обрабатываются**, пока не завершится исполнение UMOVE_BLK. Используйте команду UMOVE_BLK, если операция передачи должна быть завершена, а целевые данные непротиворечивы, до исполнения подпрограммы OB прерываний. Дополнительную информацию вы найдете в разделе о согласованности данных (стр. 96).

После выполнения команды MOVE параметр ENO всегда принимает значение ИСТИНА.

Состояние ENO	Условие	Результат
1	Нет ошибки	Все COUNT элементов успешно скопированы
0	Исходная область (IN) или целевая область (OUT) превышает имеющуюся в распоряжении область памяти	Подходящие по размеру имеющейся памяти элементы копируются. Часть элементов не копируется.

Команды заполнения



Команды FILL_BLK и UFILL_BLK используются следующим образом:

- FILL_BLK: Команда прерываемого заполнения заполняет определенный диапазон адресов копиями заданного элемента данных.
- UFILL_BLK: Команда непрерываемого заполнения заполняет определенный диапазон адресов копиями заданного элемента данных.

Параметр	Тип данных	Описание
IN	SInt, Int, DIntT, USInt, UInt, UDInt, Real, BYTE, Word, DWord	Адрес источника данных
COUNT	USInt, UInt	Количество элементов данных, подлежащих копированию
OUT	SInt, Int, DIntT, USInt, UInt, UDInt, Real, BYTE, Word, DWord	Целевой адрес для данных

Указание

Правила для операций заполнения

- Для заполнения данными типа BOOL используйте SET_BF, RESET_BF, R, S или выходную катушку (LAD)
- Для заполнения отдельным элементарным типом данных используйте MOVE
- Для заполнения массива элементарного типа данных используйте FILL_BLK или UFILL_BLK
- Для заполнения отдельного символа в строке используйте MOVE
- Команды FILL_BLK и UFILL_BLK не могут использоваться для заполнения массивов в областях памяти I, Q или M.

С помощью команд FILL_BLK и UFILL_BLK элемент данных источника IN копируется в место назначения, причем начальный адрес этого места определяется параметром OUT. Процесс копирования и заполнения соседних адресов продолжается до тех пор, пока количество копий не будет равно параметру COUNT.

Команды FILL_BLK и UFILL_BLK отличаются друг от друга способом обработки прерываний:

- Во время исполнения FILL_BLK прерывающие события **ставятся в очередь и обрабатываются**. Используйте команду FILL_BLK, если данные по адресу назначения не используются в подпрограмме внутри OB прерываний или, если используются, то целевые данные не обязательно должны быть согласованными.
- Прерывания **ставятся в очередь, но не обрабатываются**, пока не завершится исполнение UFILL_BLK. Используйте команду UFILL_BLK, если операция заполнения должна быть завершена, а целевые данные непротиворечивы, до исполнения подпрограммы OB прерываний.

Состояние ENO	**Условие**	**Результат**
1	Нет ошибки	Элемент IN был успешно скопирован во все COUNT целевых адресов
0	Целевая область (OUT) превышает доступную область памяти	Подходящие по размеру имеющейся памяти элементы копируются. Часть элементов не копируется.

6.1.6.1 Команда Swap (обмен байтов)

Команда SWAP используется для изменения порядка следования байтов в 2-байтовых и 4-байтовых элементах данных. Внутри каждого байта порядок битов не меняется. После выполнения команды SWAP параметр ENO всегда принимает значение ИСТИНА.

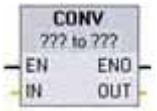
Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Параметр	Тип данных	Описание
IN	Word, DWord	Нормально расположенные байты данных в параметре IN
OUT	Word, DWord	Измененный порядок расположения байтов в параметре OUT

	Пример: Параметр IN = MB0 Перед выполнением SWAP				**Пример: Параметр OUT = MB4, После выполнения SWAP**			
Адрес	MB0	MB1			MB4	MB5		
W#16#1234	12	34			34	12		
WORD	MSB	LSB			MSB	LSB		
Адрес	MB0	MB1	MB2	MB3	MB4	MB5	MB6	MB7
DW#16# 12345678	12	34	56	78	78	56	34	12
DWORD	MSB			LSB	MSB			LSB

6.1.7 Преобразование

Команда преобразования



Команда CONVERT преобразует элемент данных из одного типа данных в другой. Щелкните под именем блока, а затем выберите типы данных для IN и OUT из ниспадающего списка.

После выбора типа данных источника (преобразовать из) в ниспадающем списке отображаются возможные преобразования (преобразовать в). Преобразования из и в BCD16 ограничены типом данных Int. Преобразования из и в BCD32 ограничены типом данных DInt.
Щелкните под именем блока и выберите типы данных из ниспадающих меню.

Параметр	Тип данных	Описание
IN	SInt, Int, DInt, USInt, UInt, UDInt, Byte, Word, DWord, Real, LReal, Bcd16, Bcd32	Значение IN
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Byte, Word, DWord, Real, LReal, Bcd16, Bcd32	Значение IN, преобразованное в новый тип данных

Состояние ENO	Описание	Результат OUT
1	Нет ошибки	Допустимый результат
0	IN равно +/- INF или +/- NaN	+/- INF или +/- NaN
0	Результат выходит за пределы допустимого диапазона для типа данных OUT	OUT устанавливается на значение, записанное в младшем байте IN

Команды Round и Truncate

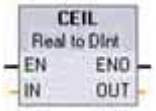
ROUND преобразует вещественное число в целое. Дробная часть вещественного числа округляется до ближайшего целого (IEEE – округление до ближайшего). Если вещественное число находится точно между двумя целыми (напр., 10.5), то вещественное число округляется до четного числа. Например, ROUND (10.5) = 10 или ROUND (11.5) = 12.

TRUNC преобразует вещественное число в целое. Дробная часть вещественного числа отбрасывается (IEEE – округление до нуля).

Параметр	Тип данных	Описание
IN	Real, LReal	Вход для числа с плавающей точкой
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Округленный или целочисленный выход

Состояние ENO	Описание	Результат OUT
1	Нет ошибки	Допустимый результат
0	IN равно +/- INF или +/- NaN	+/- INF или +/- NaN

Команды получения из вещественного числа ближайшего большего или ближайшего меньшего целого числа



Команда CEIL (от ceiling – потолок) преобразует вещественное число в наименьшее целое число, большее или равное вещественному числу (IEEE – округление до + бесконечности).

FLOOR (floor = пол) преобразует вещественное число в наибольшее целое число, не превышающее этого вещественного числа (IEEE – округление до – бесконечности).

Параметр	Тип данных	Описание
IN	Real, LReal	Вход для числа с плавающей точкой
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Преобразованный выход

Состояние ENO	Описание	Результат OUT
1	Нет ошибки	Допустимый результат
0	IN равно +/- INF или +/- NaN	+/- INF или +/- NaN

6.1.7.1 Команды масштабирования и нормализации

Команды масштабирования и нормализации

SCALE_X масштабирует нормализованный вещественный параметр VALUE, где (0,0 <= VALUE <= 1,0), в тип данных и диапазон значений, указанные в параметрах MIN и MAX:
OUT = VALUE (MAX – MIN) + MIN
Для команды SCALE_X параметры MIN, MAX и OUT должны относиться к одному и тому же типу данных.

NORM_X нормализует параметр VALUE внутри диапазона значений, указанного в параметрах MIN и MAX:
OUT = (VALUE – MIN) / (MAX – MIN), где (0,0 <= OUT <= 1,0)
Для команды NORM_X параметры MIN, VALUE и MAX должны относиться к одному и тому же типу данных.

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Параметр	Тип данных	Описание
MIN	SInt, Int, DInt, USInt, UInt, UDInt, Real	Вход для минимального значения диапазона
VALUE	SCALE_X: Real NORM_X: SInt, Int, DInt, USInt, UInt, UDInt, Real	Входное значение для масштабирования или нормализации
MAX	SInt, Int, DInt, USInt, UInt, UDInt, Real	Вход для максимального значения диапазона
OUT	SCALE_X: SInt, Int, DInt, USInt, UInt, UDInt, Real NORM_X: Real	Масштабированное или нормализованное выходное значение

Указание

Параметр VALUE команды SCALE_X должен находиться в диапазоне значений (0,0 <= VALUE <= 1,0)

Если параметр VALUE выходит за пределы этого диапазона, то:

- операция линейного масштабирования может выдавать значения OUT, меньшие параметра MIN или превышающие параметр MAX для значений OUT, находящихся внутри диапазона значений для типа данных OUT. Для этих случаев исполнение команды SCALE_X устанавливает ENO = ИСТИНА.
- возможно генерирование масштабированных чисел, которые не находятся в диапазоне допустимых значений для типа данных OUT. Тогда параметр OUT устанавливается на промежуточное значение, равное наименее значимой части масштабированного вещественного числа перед окончательным преобразованием в тип данных OUT. Для этих случаев исполнение команды SCALE_X устанавливает ENO = ЛОЖЬ.

Параметр VALUE команды NORM_X должен находиться в диапазоне значений (MIN <= VALUE <= MAX)

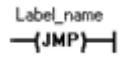
Если параметр VALUE меньше, чем MIN, или больше, чем MAX, то операция линейного масштабирования может выдавать нормализованные значения OUT, меньшие 0.0 или большие 1.0. В этом случае исполнение команды NORM_X устанавливает ENO = ИСТИНА.

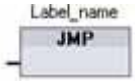
Состояние ENO	Условие	Результат OUT
1	Нет ошибки	Допустимый результат
0	Результат выходит за пределы допустимого диапазона для типа данных OUT	Промежуточный результат: Наименее значимая часть вещественного числа перед окончательным преобразованием в тип данных OUT.
0	Параметр MAX <= MIN	SCALE_X: Наименее значимая часть вещественного числа VALUE, которой заполняется участок памяти, отводимый под OUT. NORM_X: значение VALUE в типе данных VALUE, расширенное для заполнения участка памяти размером в двойное слово.
0	Параметр VALUE = +/- INF или +/- NaN	VALUE записывается в OUT

6.1.8 Управление программой

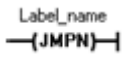
Команды перехода и метки перехода

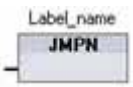
Команды управления программой используются для воздействия на последовательность исполнения программы в зависимости от тех или иных условий:





JMP: Если имеется поток сигнала к катушке JMP (LAD), или если вход блока JMP принимает значение ИСТИНА (FBD), то исполнение программы продолжается с первой команды после указанной метки.





JMPN: Если отсутствует поток сигнала к катушке JMP (LAD), или если вход блока JMP принимает значение ЛОЖЬ (FBD), то исполнение программы продолжается с первой команды после указанной метки.

LABEL [Метка]: Метка места назначения для команды перехода JMP или JMPN.

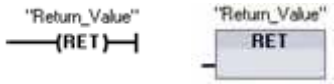
LAD **FBD**

Параметр	Тип данных	Описание
Label_name	Идентификатор метки	Идентификатор для команд перехода и соответствующая метка места перехода

Имена меток создаются непосредственным вводом команды LABEL. Имеющиеся в распоряжении имена меток для поля с именами меток перехода команд JMP и JMPN можно выбрать с помощью символа поддержки параметра. Вы можете также непосредственно впечатать имя метки в команду JMP или JMPN.

Команда управления исполнением программы Return_Value [Возвращаемое значение] (RET)

LAD **FBD**



Команда RET используется для завершения исполнения текущего блока.

Параметр	Тип данных	Описание
Return_Value	Bool	Параметр "Return_value" команды RET назначается выходу ENO вызываемого блока в вызывающем блоке.

Необязательная команда RET используется для завершения исполнения текущего блока. В том и только в том случае, если имеет место поток сигнала к катушке RET (LAD) или если вход блока RET принимает значение ИСТИНА (FBD), то исполнение программы текущего блока завершается в этом месте и команды, следующие за командой RET, не исполняются. Если текущий блок является OB, то параметр "Return_Value" игнорируется. Если текущий блок является FC или FB, то значение параметра "Return_Value " передается обратно в вызывающую программу как значение ENO вызываемого блока.

У вас нет необходимости вставлять RET в качестве последней команды в блоке; это происходит автоматически. Вы можете вставить несколько команд RET в один и тот же блок.

Образцы шагов для использования команды RET внутри кодового блока FC:

1. Создайте новый проект и вставьте FC:
2. Редактируйте FC:
   - Вставляйте команды из каталога команд.
   - Вставьте команду RET, включая один из следующих элементов для параметра "Return_Value":

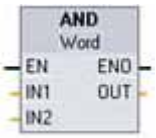
     TRUE (ИСТИНА), FALSE (ЛОЖЬ) или адрес в памяти, который указывает требуемое возвращаемое значение.
   - Вставляйте следующие команды.
3. Вызовите FC из MAIN [OB1].

Вход EN блока FC в кодовом блоке MAIN должен принять значение ИСТИНА, чтобы начать исполнение FC.

Значение, определяемое командой RET в FC, будет находиться на выходе ENO блока FC в кодовом блоке MAIN после исполнения блока FC, для которого поток сигнала к команде RET принял значение ИСТИНА.

6.1.9 Логические операции

Команды AND (И), OR (ИЛИ) и XOR (исключающее ИЛИ)



AND: Логическое И для типов данных BYTE, WORD и DWORD

OR: Логическое ИЛИ для типов данных BYTE, WORD и DWORD

XOR: Логическое исключающее ИЛИ для типов данных BYTE, WORD и DWORD

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Параметр	Тип данных	Описание
IN1, IN2	Byte, Word, DWord	Логические входы
OUT	Byte, Word, DWord	Логический выход

Выбором типа данных параметры IN1, IN2 и OUT устанавливаются на один и тот же тип данных. Соответствующие битовые значения IN1 и IN2 логически сопрягаются, формируя логический двоичный результат в параметре OUT. После выполнения этих команд ENO всегда принимает значение ИСТИНА.

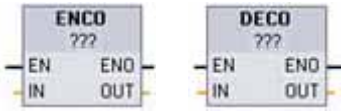
Команда инвертирования

Команда INV используется для получения двоичного дополнения до единицы параметра IN. Дополнение до единицы образуется инвертированием значения каждого бита параметра IN (замена каждого 0 на 1 и каждой 1 на 0). После выполнения этой команды ENO всегда принимает значение ИСТИНА.

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Параметр	Тип данных	Описание
IN	SInt, Int, DInt, USInt, UInt, UDInt, Byte, Word, DWord	Элемент данных, подлежащий инвертированию
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Byte, Word, DWord	Инвертированный выход

Команды кодирования и декодирования



Команда ENCO преобразует (кодирует) битовый образ в двоичное число.
Команда DECO преобразует (декодирует) двоичное число в битовый образ.
Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Параметр	Тип данных	Описание
IN	ENCO: Byte, Word, DWord DECO: UInt	ENCO: Бит образ для кодирования DECO: Значение для декодирования
OUT	ENCO: Int DECO: Byte, Word, DWord	ENCO: Закодированное значение DECO: Декодированный битовый образ

Команда ENCO преобразует параметр IN в двоичное число, соответствующее положению самого младшего установленного бита параметра IN, и возвращает результат в параметр OUT. Если параметр IN равен 0000 0001 или 0000 0000, то в OUT возвращается значение 0. Если значение параметра IN равно 0000 0000, то ENO устанавливается в ЛОЖЬ.

Команда DECO декодирует двоичное число из параметра IN, устанавливая соответствующую битовую позицию в параметре OUT в 1 (все остальные биты устанавливаются в 0). После выполнения команды DECO ENO всегда принимает значение ИСТИНА.

Выбором типа данных Byte, Word или DWord для параметра OUT команды DECO ограничивается полезный диапазон параметра IN. Если значение параметра IN выходит за пределы полезного диапазона, то выполняется операция modulo для извлечения младших значащих битов, как показано ниже.

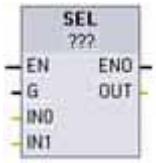
Битовый диапазон для параметра IN команды DECO:

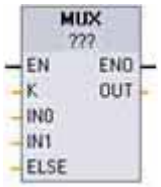
- 3 бита (значения 0-7) IN используются для установки 1 битовой позиции в байте OUT
- 4 бита (значения 0-15) IN используются для установки 1 битовой позиции в слове OUT
- 5 битов (значения 0-31) IN используются для установки 1 битовой позиции в двойном слове OUT

Значение IN для DECO		Значение OUT для DECO (декодирование позиции отдельного бита)
		Byte OUT (8 битов):
мин. IN	0	00000001
макс. IN	7	10000000
		Word OUT (16 битов):
мин. IN	0	0000000000000001
макс. IN	15	1000000000000000
		DWord OUT: (32 бита):
мин. IN	0	00000000000000000000000000000001
макс. IN	31	10000000000000000000000000000000

Состояние ENO	Условие	Результат (OUT)
1	Нет ошибки	Допустимый номер бита
0	IN равно нулю	OUT устанавливается в ноль

Команды выбора (SEL) и мультиплексирования (MUX)





- Команда SEL присваивает одно из двух входных значений параметру OUT, в зависимости от значения параметра G.
- Команда MUX присваивает одно из нескольких входных значений параметру OUT, в зависимости от значения параметра K. Если значение параметра K выходит за пределы допустимого диапазона, то параметру OUT присваивается значение параметра ELSE.

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

SEL	**Тип данных**	**Описание**
G	Bool	Селекторный переключатель: • ЛОЖЬ для IN0 • ИСТИНА для IN1
IN0, IN1	SInt, Int, DInt, USInt, UInt, UDInt, Real, Byte, Word, DWord, Time, Char	Входы
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, Byte, Word, DWord, Time, Char	Выход

MUX	**Тип данных**	**Описание**
K	UInt	Значение переключателя: • 0 для IN0 • 1 для IN1 • ...
IN0, IN1,....	SInt, Int, DInt, USInt, UInt, UDInt, Real, Byte, Word, DWord, Time, Char	Входы
ELSE	SInt, Int, DInt, USInt, UInt, UDInt, Real, Byte, Word, DWord, Time, Char	Заменяющее входное значение (факультативно)
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, Byte, Word, DWord, Time, Char	Выход

Входные переменные и выходная переменная должны быть одного и того же типа данных.

- Команда SEL всегда осуществляет выбор между двумя значениями IN.
- Команда MUX после ее вставки в программном редакторе всегда имеет два параметра IN, но она может быть расширена добавлением еще нескольких параметров IN.

Для добавления и удаления входных параметров команды MUX действуйте следующим образом:

- Для добавления входа щелкните правой клавишей мыши у одного из имеющихся параметров IN и выберите команду "Insert input [Вставить вход]".
- Для удаления входа щелкните правой клавишей мыши у одного из имеющихся параметров IN (если имеется более чем два первоначальных входа) и выберите команду "Delete [Удалить]".

Коды условий: ENO всегда принимает значение ИСТИНА после выполнения команды SEL.

Состояние ENO (MUX)	**Условие MUX**	**Результат выполнения MUX в OUT**
1	Нет ошибки	Выбранное значение IN присваивается параметру OUT
0	K больше или равно числу параметров IN	Параметр ELSE не задан: OUT не меняется ELSE задан: значение ELSE присваивается параметру OUT

6.1.10 Операции сдвига и циклического сдвига

Команда сдвига

Команды сдвига используются для смещения битового образа параметра IN. Результат присваивается параметру OUT. Параметр N определяет битовых позиций, на которые осуществляется сдвиг:

- SHR: Сдвинуть битовый образ вправо
- SHL: Сдвинуть битовый образ влево

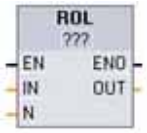
Щелкните под именем блока и выберите тип данных из ниспадающего списка.

Параметр	Тип данных	Описание
IN	Byte, Word, DWord	Битовый образ, подлежащий сдвигу
N	UInt	Число битовых позиций, на которые осуществляется сдвиг
OUT	Byte, Word, DWord	Битовый образ после операции сдвига

- При N=0 сдвиг не производится, и значение IN присваивается параметру OUT.
- Битовые позиции, которые освобождаются операцией сдвига, заполняются нулями.
- Если число позиций, на которые осуществляется сдвиг, (N) превышает число битов в целевом значении (8 для байта, 16 для слова, 32 для двойного слова), то все первоначальные значения битов бит выдвигаются и заменяются нулями (выходу OUT присваивается нулевое значение).
- Для операций сдвига ENO всегда принимает значение ИСТИНА.

Пример SHL для размера данных Word: Вставка нулей на освободившиеся позиции			
IN	1110 0010 1010 1101	Значение OUT перед первым сдвигом:	1110 0010 1010 1101
		После первого сдвига:	1100 0101 0101 1010
		После второго сдвига:	1000 1010 1011 0100
		После третьего сдвига:	0001 0101 0110 1000

Команда циклического сдвига



С помощью команд циклического сдвига вы можете циклически сдвигать битовый образ параметра IN. Результат присваивается параметру OUT. Параметр N определяет число битовых позиций, на которое осуществляется циклический сдвиг.

- ROR: Циклический сдвиг битового образа вправо
- ROL: Циклический сдвиг битового образа влево

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Параметр	Тип данных	Описание
IN	Byte, Word, DWord	Битовый образ, подлежащий циклическому сдвигу
N	UInt	Число битовых позиций, на которое должен быть произведен циклический сдвиг
OUT	Byte, Word, DWord	Битовый образ после операции циклического сдвига

- При N=0 циклический сдвиг не производится, и значение IN присваивается параметру OUT.
- Битовые данные, выдвигаемые с одной стороны целевого значения, вдвигаются с другой стороны целевого значения, так что ни одно из первоначальных битовых значений не теряется.
- Если число битовых позиций, на которое осуществляется циклический сдвиг, (N) превышает число битов в целевом значении (8 для байта, 16 для слова, 32 для двойного слова), то циклический сдвиг все равно выполняется.
- ENO всегда принимает значение ИСТИНА после выполнения команд циклического сдвига.

Пример ROR для данных размера WORD размер: Биты, выдвигаемые с правой стороны, вдвигаются с левой стороны			
IN	0100 0000 0000 0001	Значение OUT перед первым циклическим сдвигом:	0100 0000 0000 0001
		После первого циклического сдвига вправо:	1010 0000 0000 0000
		После второго циклического сдвига вправо:	0101 0000 0000 0000

6.2 Расширенные команды

6.2.1 Общие параметры ошибок для расширенных команд

Описания расширенных команд содержат сведения об ошибках этапа исполнения, которые могут произойти для каждой команды. Кроме этих ошибок, возможны также общие ошибки, приведенные ниже. Если при исполнении кодового блока происходит одна из общих ошибок, то CPU переходит в состояние STOP, если вы не используете внутри этого кодового блока команду GetError или GetErrorID для формирования запрограммированной реакции на эту ошибку.

Значение кода ошибки (W#16#....)	**Описание**
8022	Область для ввода слишком мала
8023	Область для вывода слишком мала
8024	Недопустимая область ввода
8025	Недопустимая область вывода
8028	Недопустимое назначение входного бита
8029	Недопустимое назначение выходного бита
8030	Областью вывода является DB, защищенный от записи
803A	DB не существует

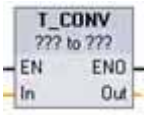
6.2.2 Команды для часов и календаря

Команды для часов и календаря

Команды для часов и календаря используются для расчетов, связанных с календарем и временем.

- T_CONV преобразует тип данных значения времени: (Time to DInt [Время в двойное целое]) или (DInt to Time [Двойное целое во время])
- T_ADD складывает значения Time и DTL: (Time + Time = Time) или (DTL + Time = DTL)
- T_SUB вычитает значения Time и DTL: (Time - Time = Time) или (DTL - Time = DTL)
- T_DIFF выдает разность между двумя значениями DTL как значение Time: DTL - DTL = Time

Тип данных	Размер (в битах)	Допустимые диапазоны
Time [Время]	32 Хранится как	от T#-24d_20h_31m_23s_648ms до T#24d_20h_31m_23s_647ms от -2 147 483 648 мс до +2 147 483 647 мс
Структура данных DTL		
Год: UInt	16	от 1970 до 2554
Месяц: USInt	8	от 1 до 12
День: USInt	8	от 1 до 31
День недели: USInt	8	от 1=воскресенье до 7=суббота
Час: USInt	8	от 0 до 23
Минута: USInt	8	от 0 до 59
Секунда: USInt	8	от 0 до 59
Наносекунды: UDInt	32	от 0 до 999,999,999



T_CONV (преобразование времени) преобразует тип данных Time в тип данных DInt, или осуществляет обратное преобразование из типа данных DInt в тип данных Time.

Параметр	Тип параметра	Тип данных	Описание
IN	IN	DInt, Time	Входное значение типа Time или Dint
OUT	OUT	DInt, Time	Преобразованное значение типа DInt или Time

Выберите типы данных IN и OUT из ниспадающих списков под именем команды.

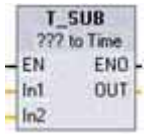
T_ADD (сложение времен) складывает входное значение IN1 (типа данных DTL или Time) с входным значением IN2 (тип Time). Параметр OUT выдает результат в виде значения типа DTL или Time.

Возможны операции с двумя типами данных:

- Time + Time = Time
- DTL + Time = DTL

Параметр	Тип параметра	Тип данных	Описание
IN1	IN	DTL, Time	Значение типа DTL или Time
IN2	IN	Time	Значение типа Time, которое нужно прибавить
OUT	OUT	DTL, Time	Сумма типа DTL или Time

Выберите тип данных IN1 из ниспадающего списка под списком команды. Выбор типа данных IN1 устанавливает также тип данных параметра OUT.



T_SUB (вычитание времени) вычитает значение типа Time в IN2 из IN1 (значение типа DTL или Time). Параметр OUT выдает значение разности, используя тип данных DTL или Time.

Возможны операции с двумя типами данных:

- Time - Time = Time
- DTL - Time = DTL

Параметр	Тип параметра	Тип данных	Описание
IN1	IN	DTL, Time	Значение типа DTL или Time
IN2	IN	Time	Значение типа Time, которое нужно вычесть
OUT	OUT	DTL, Time	Разность типа DTL или Time

Выберите тип данных IN1 из ниспадающего списка под именем команды. Выбор типа данных IN1 устанавливает также тип данных параметра OUT.

T_DIFF (разность времен) вычитает значение типа DTL в IN2 из значения типа DTL в IN1. Параметр OUT выдает значение разности, используя тип данных Time.

- DTL - DTL = Time

Параметр	Тип параметра	Тип данных	Описание
IN1	IN	DTL	Значение типа DTL
IN2	IN	DTL	Значение типа DTL, которое нужно вычесть
OUT	OUT	Time	Разность типа Time

Коды условий: ENO = 1 означает, что ошибки не произошло. ENO = 0 и параметр OUT = 0 – ошибки:

- Недопустимое значение DTL
- Недопустимое значение Time

Команды для работы с часами

Команды для работы с часами позволяют устанавливать и считывать системные часы ПЛК. Для вывода значений даты и времени используется тип данных DTL.

Структура DTL	Размер	Допустимые диапазоны
Год: UInt	16 битов	от 1970 до 2554
Месяц: USInt	8 битов	от 1 до 12
День: USInt	8 битов	от 1 до 31
День недели: USInt	8 битов	от 1=Sunday до 7=Saturday
Час: USInt	8 битов	от 0 до 23
Минута: USInt	8 битов	от 0 до 59
Секунда: USInt	8 битов	от 0 до 59
Наносекунды: UDInt	32 бита	от 0 до 999,999,999

WR_SYS_T (запись системного времени) устанавливает часы истинного времени ПЛК с помощью значения типа данных DTL в параметре IN. Это значение времени не учитывает ни местного часового пояса, ни переходов на зимнее время и обратно.

Параметр	Тип параметра	Тип данных	Описание
IN	IN	DTL	Истинное время, подлежащее установке в системных часах ПЛК
RET_VAL	OUT	Int	Код условия выполнения

RD_SYS_T (считывание системного времени) считывает текущее системное время из ПЛК. Это значение времени не учитывает ни местного часового пояса, ни переходов на зимнее время и обратно.

Параметр	Тип параметра	Тип данных	Описание
RET_VAL	OUT	Int	Код условия выполнения
OUT	OUT	DTL	Текущее системное время ПЛК

RD_LOC_T (считывание местного времени) предоставляет текущее местное время ПЛК как тип данных DTL.

Параметр	Тип параметра	Тип данных	Описание
RET_VAL	OUT	Int	Код условия выполнения
OUT	OUT	DTL	Местное время

- Для расчета местного времени используются часовой пояс и времена переключения на зимнее время и обратно, которые вы ввели в конфигурации устройств для часов CPU.
- Часовой пояс представляет собой смещение по отношению к скоординированному универсальному времени (Universal Time Coordinated, UTC).
- Для переключения на летнее время необходимо ввести месяц, неделю, день и час, когда осуществляется перевод.
- Для переключения на зимнее время также необходимо ввести месяц, неделю, день и час, когда осуществляется перевод.
- Разность часовых поясов относительно системного времени действует всегда. Переключение на летнее время действует только тогда, когда этот переход имеет место.

Коды условий: ENO = 1 означает, что ошибок не было. ENO = 0 означает, что произошла ошибка исполнения, и на выходе RET_VAL находится код условия.

RET_VAL (W#16#....)	Описание
0000	Нет ошибки
8080	Отсутствует местное время
8081	Недопустимое значение года
8082	Недопустимое значение месяца
8083	Недопустимое значение дня
8084	Недопустимое значение часа
8085	Недопустимое значение минуты
8086	Недопустимое значение секунды
8087	Недопустимое значение наносекунды
80B0	Часы реального времени вышли из строя

6.2.3 Операции над строками и символами

6.2.3.1 Обзор данных строки

Тип данных String

Данные типа String хранятся в виде 2-байтного заголовка, за которым следует до 254 байтов символов ASCII. Заголовок для данных типа String содержит два байта для длины. Первый байт содержит максимальную длину строки, которая указывается в квадратных скобках при инициализации строки, или устанавливается на 254 по умолчанию. Второй байт заголовка – это текущая длина, равная числу действительных символов в строке. Текущая длина не должна превышать максимальную длину. Число сохраняемых байтов, занимаемых форматом типа String на 2 байта больше, чем максимальная длина.

Инициализация данных типа String

Входные и выходные данные типа String должны быть инициализированы в памяти как действительные строки перед исполнением команд над строками.

Действительные данные типа String

Действительная строка имеет максимальную длину, которая должна быть больше нуля, но меньше 255. Текущая длина не должна превышать максимальной длины.

Строки не должны назначаться областям памяти входов (I) или выходов (Q).

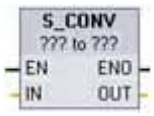
Дальнейшую информацию вы найдете под заголовком "Формат типа данных STRING" (стр. 65)

6.2.3.2 Команды преобразования строки

Преобразования строки в значения и значения в строку

С помощью этих команд вы можете преобразовывать строки цифровых символов в числовые значения и числовые значения в строки цифровых символов:

- S_CONV преобразует строку цифровых символов в числовое значение или числовое значение в строку цифровых символов
- STRG_VAL преобразует строку цифровых символов в числовое значение с возможностями форматирования
- VAL_STRG преобразует числовое значение в строку цифровых символов с возможностями форматирования



S_CONV (преобразование строки) преобразует строку символов в соответствующее число или число в соответствующую строку символов. Команда S_CONV не имеет возможностей форматирования выхода. Это делает команду S_CONV боле простой, но менее гибкой, чем команды STRG_VAL и VAL_STRG.

Выберите типы данных параметров из ниспадающих списков.

S_CONV (преобразование строки символов в числовое значение)

Параметр	Тип параметра	Тип данных	Описание
IN	IN	String	Вводимая строка символов
OUT	OUT	String, SInt, Int, DInt, USInt, UInt, UDInt, Real	Выводимое числовое значение

Преобразование параметра строки символов IN начинается с первого символа и продолжается до конца строки или до тех пор, пока не встретится первый символ, отличный от "0" – "9", "+", "-", или ".". Результирующе значение передается по адресу, указанному в параметре OUT. Если выходное числовое значение не соответствует диапазону типа данных OUT, то параметр OUT устанавливается в 0, а ENO устанавливается в ЛОЖЬ. В противном случае параметр OUT содержит действительный результат, и ENO принимает значение ИСТИНА.

Правила форматирования для ввода строки:

- Если в строке IN используется знак, разделяющий целую и дробную часть, то вы должны использовать символ ".".
- Символы "," используемые в качестве разделителей тысяч слева от десятичной точки, допускаются, но игнорируются.
- Ведущие пробелы игнорируются.
- Допускается представление чисел только с фиксированной точкой. Символы "e" и "E" не распознаются в качестве экспоненциального представления.

S_CONV (преобразование числового значения в строку символов)

Параметр	Тип параметра	Тип данных	Описание
IN	IN	String, SInt, Int, DInt, USInt, UInt, UDInt, Real	Вводимое числовое значение
OUT	OUT	String	Выводимая строка символов

Целое, целое без знака или число с плавающей точкой на входе IN преобразуется в соответствующую строку символов на выходе OUT. Параметр OUT должен ссылаться на действительную строку перед выполнением преобразования. Действительная строка содержит максимальную длину строки в первом байте, текущую длину строки во втором байте и символы текущей строки в следующих байтах. Преобразованная строка заменяет символы в строке OUT, начиная с первого символа, и согласует байт фактической длины строки со строкой OUT. Байт максимальной длины строки OUT не изменяется.

Количество заменяемых символов зависит от типа данных параметра IN и числового значения. Число заменяемых символов должно подходить к длине строки параметра OUT. Максимальная длина (первый байт) строки OUT должна быть не меньше максимально ожидаемого числа преобразуемых символов.

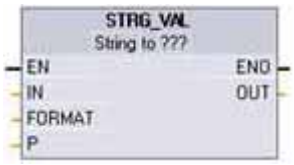
В следующей таблице показаны максимально возможные длины строк, необходимые для каждого из поддерживаемых типов данных.

Тип данных IN	Максимальное число преобразуемых символов в строке OUT	Пример	Общая длина строки, включая байты максимальной и текущей длины
USInt	3	255	5
SInt	4	-128	6
UInt	5	65535	7
Int	6	-32768	8
UDInt	10	4294967295	12
DInt	11	-2147483648	13

Правила форматирования для вывода строки:

- Значения, записываемые в параметр OUT, не используют ведущий знак "+".
- Используется представление чисел с фиксированной точкой (не экспоненциальное представление).
- Если параметр IN имеет тип данных Real, то для разделения целой и дробной части числа используется десятичная точка ".".

Команда STRG_VAL



STRG_VAL (строка в значение) преобразует строку цифровых символов в соответствующее целое число или число с плавающей точкой. Преобразование начинается во вводимой строке IN со смещения, указанного в параметре P, и продолжается до конца строки или до тех пор, пока не встретится первый символ, не являющийся "+", "-", ".", ",", "e", "E" или цифрой от "0" до "9". Результат помещается по адресу, указанному в параметре OUT.

Параметр P также возвращается как величина смещения в исходной строке символов на том месте, где заканчивается преобразование. Перед исполнением команды данные строки должны быть инициализированы в памяти как действительная строка.

Параметр	Тип параметра	Тип данных	Описание
IN	IN	String	Строка символов ASCII, подлежащая преобразованию
FORMAT	IN	Word	Варианты для формата вывода
P	IN_OUT	UInt	IN: Указатель на первый символ, подлежащий преобразованию (первый символ = 1) OUT: Указатель на следующий символ после завершения преобразования
OUT	OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real	Преобразованное числовое значение

Параметр FORMAT команды STRG_VAL

Параметр FORMAT команды STRG_VAL определен ниже. Неиспользуемые битовые позиции должны быть установлены в ноль.

Бит 16							**Бит 8**	**Бит 7**							**Бит 0**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	f	r

f = формат представления — 1= экспоненциальное представление; 0 = представление с фиксированной точкой

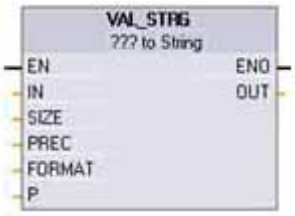
r = формат десятичной точки — 1 = "," (запятая); 0 = "." (точка)

FORMAT (W#16#)	**Формат представления**	**Представление десятичной точки**
0000 (по умолчанию)	Фиксированная точка	"."
0001		","
0002	Экспоненциальное	"."
0003		","
от 0004 до FFFF	Недопустимые значения	

Правила преобразования для STRG_VAL:

- Если для разделения целой и дробной части числа используется символ ".", то запятые "," слева от десятичной точки интерпретируются как символы разделения тысяч. Запятые допускаются, но игнорируются.
- Если для разделения целой и дробной части числа используется символ "," то точки "." слева от запятой интерпретируются как символы разделения тысяч. Эти точки допускаются, но игнорируются.
- Ведущие пробелы игнорируются.

Команда VAL_STRG



VAL_STRG (значение в строку) преобразует целое, целое без знака или число с плавающей точкой в соответствующую строку символов. Значение, представленное параметром IN, преобразуется в строку, на которую ссылается параметр OUT. Перед выполнением преобразования параметр OUT должен быть действительной строкой.

Конвертированная строка заменяет символы в строке OUT, начиная с указанного в параметре P смещения до числа символов, указанного в параметре SIZE. Число символов в SIZE должно укладываться в длину строки OUT, начиная с позиции P. Эта команда полезна для встраивания цифровых символов в текстовую строку. Например, вы можете поместить цифры "120" в строку "Давление насоса = 120 кг/кв.см".

Параметр	Тип параметра	Тип данных	Описание
IN	IN	SInt, Int, DInt, USInt, UInt, UDInt, Real	Значение, подлежащее преобразованию
SIZE	IN	USInt	Число символов, подлежащих записи в строку OUT
PREC	IN	USInt	Точность или размер дробной части. Он не включает десятичную точку.
FORMAT	IN	Word	Возможности форматирования выхода
P	IN_OUT	UInt	IN: Указатель на первый символ строки OUT, подлежащий замене (первый символ = 1) OUT: Указатель на следующий после замены символ строки OUT
OUT	OUT	String	Преобразованная строка

Параметр PREC определяет точность или число символов для дробной части в строке символов. Если значение параметра IN – целое число, то PREC определяет положение десятичной точки. Например, если значение данных равно 123 и PREC = 1, то результатом будет "12.3". Максимальная поддерживаемая точность для типа данных REAL составляет 7 цифр.

Если параметр P больше, чем текущий размер строки OUT, то до позиции P вставляются пробелы, а результат присоединяется к концу строки. Преобразование заканчивается, когда достигнута максимальная длина строки OUT.

Параметр FORMAT команды VAL_STRG

Параметр FORMAT команды VAL_STRG определен ниже. Неиспользуемые битовые позиции должны быть заменены нулями.

Бит 16							**Бит 8**	**Бит 7**							**Бит 0**
0	0	0	0	0	0	0	0	0	0	0	0	0	s	f	r

s = символ для знака числа — 1= использовать знаки "+" и "-"; 0 = использовать только знак "-"

f = представление формата — 1= экспоненциальное представление; 0 = представление с фиксированной точкой

r = формат для разделителя целой и дробной части — 1 = "," (запятая); 0 = "." (точка)

FORMAT (WORD)	**Символ для знака числа**	**Формат представления**	**Представление десятичной точки**
W#16#0000	Только "-"	Фиксированная точка	"."
W#16#0001			","
W#16#0002		Экспоненциальное	"."
W#16#0003			","
W#16#0004	"+" и "-"	Фиксированная точка	"."
W#16#0005			","
W#16#0006		Экспоненциальное	"."
W#16#0007			","
от W#16#0008 до W#16#FFFF	Недопустимые значения		

Правила форматирования для строки в параметре OUT:

- Если преобразованная строка короче указанного размера, то слева к ней добавляются ведущие пробелы.
- Если бит знака параметра FORMAT имеет значение ЛОЖЬ, то целые со знаком и без знака записываются в выходной буфер без ведущего знака "+". В случае необходимости используется знак "-".
  <ведущие пробелы><цифры без ведущих нулей>'.'<цифры PREC>
- Если бит знака имеет значение ИСТИНА, то целые со знаком и без знака всегда записываются в выходной буфер с ведущим знаком.
  <ведущие пробелы><знак><цифры без ведущих нулей>'.'<цифры PREC>
- Если в качестве параметра FORMAT устанавливается экспоненциальное представление, то числа, имеющие тип данных REAL, записываются в выходной буфер следующим образом:
  <ведущие пробелы><знак><цифра> '.' <цифры PREC>'E' ><знак><цифры без ведущего нуля>

- Если в качестве параметра FORMAT устанавливается представление с фиксированной точкой, то целые, целые без знака и вещественные значения записываются в выходной буфер следующим образом:
  <ведущие пробелы><знак><цифры без ведущих нулей>'.'<цифры PREC>
- Ведущие нули слева от десятичной точки (кроме цифры, стоящей непосредственно перед десятичной точкой) подавляются.
- Значения справа от десятичной точки округляются, чтобы уместиться в число цифр справа от десятичной точки, заданное параметром PREC.
- Размер выводимой строки должен, по крайней мере, на три байта превышать число цифр справа от десятичной точки.
- Значения в выводимой строке выравниваются вправо.

Условия, сообщаемые параметром ENO

Если при преобразовании возникает ошибка, то выводятся следующие результаты:

- ENO устанавливается в 0.
- OUT устанавливается в 0, или на значение, показанное в примерах на преобразование строк в значение.
- OUT остается неизменным или устанавливается на значение, показанное в примерах, когда OUT является строкой.

Состояние ENO	Описание
1	Нет ошибки
0	Недопустимый или недействительный параметр; например, обращение к несуществующему DB
0	Недопустимая строка с максимальной длиной 0 или 255
0	Недопустимая строка, в которой текущая длина больше, чем максимальная
0	Преобразованное числовое значение слишком велико для указанного типа данных OUT
0	Максимальный размер строки для параметра OUT должен быть достаточно велик, чтобы принять число символов, указанное в параметре SIZE, начиная с позиции символа, указанной в параметре P
0	Недопустимое значение P, где P=0 или P больше, чем текущая длина строки
0	Параметр SIZE должен быть больше параметра PREC

Примеры преобразования строк в числовые значения с помощью S_CONV

Строка IN	Тип данных OUT	Значение OUT	ENO
"123"	Int/DInt	123	ИСТИНА
"-00456"	Int/DInt	-456	ИСТИНА
"123.45"	Int/DInt	123	ИСТИНА
"+2345"	Int/DInt	2345	ИСТИНА
"00123AB"	Int/DInt	123	ИСТИНА
"123"	Real	123.0	ИСТИНА
"123.45"	Real	123.45	ИСТИНА
"1.23e-4"	Real	1.23	ИСТИНА
"1.23E-4"	Real	1.23	ИСТИНА
"12,345.67"	Real	12345.67	ИСТИНА
"3.4e39"	Real	3.4	ИСТИНА
"-3.4e39"	Real	-3.4	ИСТИНА
"1.17549e-38"	Real	1.17549	ИСТИНА
"12345"	SInt	0	ЛОЖЬ
"A123"	N/A	0	ЛОЖЬ
""	N/A	0	ЛОЖЬ
"++123"	N/A	0	ЛОЖЬ
"+-123"	N/A	0	ЛОЖЬ

Примеры преобразования числовых значений в строки с помощью S_CONV

Тип данных	Значение IN	Строка OUT	ENO
UInt	123	"123"	ИСТИНА
UInt	0	"0"	ИСТИНА
UDInt	12345678	"12345678"	ИСТИНА
Real	-INF	"INF"	ЛОЖЬ
Real	+INF	"INF"	ЛОЖЬ
Real	NaN	"NaN"	ЛОЖЬ

Примеры преобразования с помощью STRG_VAL

Строка IN	FORMAT (W#16#....)	Тип данных OUT	Значение OUT	ENO
"123"	0000	Int/DInt	123	ИСТИНА
"-00456"	0000	Int/DInt	-456	ИСТИНА
"123.45"	0000	Int/DInt	123	ИСТИНА
"+2345"	0000	Int/DInt	2345	ИСТИНА
"00123AB"	0000	Int/DInt	123	ИСТИНА
"123"	0000	Real	123.0	ИСТИНА
"-00456"	0001	Real	-456.0	ИСТИНА
"+00456"	0001	Real	456.0	ИСТИНА
"123.45"	0000	Real	123.45	ИСТИНА
"123.45"	0001	Real	12345.0	ИСТИНА
"123,45"	0000	Real	12345.0	ИСТИНА
"123,45"	0001	Real	123.45	ИСТИНА
".00123AB"	0001	Real	123.0	ИСТИНА
"1.23e-4"	0000	Real	1.23	ИСТИНА
"1.23E-4"	0000	Real	1.23	ИСТИНА
"1.23E-4"	0002	Real	1.23E-4	ИСТИНА
"12,345.67"	0000	Real	12345.67	ИСТИНА
"12,345.67"	0001	Real	12.345	ИСТИНА
"3.4e39"	0002	Real	+INF	ИСТИНА
"-3.4e39"	0002	Real	-INF	ИСТИНА
"1.1754943e-38" (и меньше)	0002	Real	0.0	ИСТИНА
"12345"	N/A	SInt	0	ЛОЖЬ
"A123"	N/A	N/A	0	ЛОЖЬ
""	N/A	N/A	0	ЛОЖЬ
"++123"	N/A	N/A	0	ЛОЖЬ
"+-123"	N/A	N/A	0	ЛОЖЬ

Примеры преобразования с помощью VAL_STRG

Эти примеры основаны на строке OUT, инициализированной следующим образом:

"Current Temp = xxxxxxxxxx C"
Символ "x"представляет пробелы, предназначенные для преобразуемого значения.

Тип данных	Значение IN	P	SIZE	FORMAT (W#16#....)	PREC	Строка OUT	ENO
UInt	123	16	10	0000	0	Current Temp = xxxxxxx123 C	ИСТИНА
UInt	0	16	10	0000	2	Current Temp = xxxxxx0.00 C	ИСТИНА
UDInt	12345678	16	10	0000	3	Current Temp = x12345.678 C	ИСТИНА
UDInt	12345678	16	10	0001	3	Current Temp = x12345,678 C	ИСТИНА
Int	123	16	10	0004	0	Current Temp = xxxxxx+123 C	ИСТИНА
Int	-123	16	10	0004	0	Current Temp = xxxxxx-123 C	ИСТИНА
Real	-0.00123	16	10	0004	4	Current Temp = xxx-0.0012 C	ИСТИНА
Real	-0.00123	16	10	0006	4	Current Temp = -1.2300E-3 C	ИСТИНА
Real	-INF	16	10	N/A	4	Current Temp = xxxxxx-INF C	ЛОЖЬ
Real	+INF	16	10	N/A	4	Current Temp = xxxxxx+INF C	ЛОЖЬ
Real	NaN	16	10	N/A	4	Current Temp = xxxxxxxNaN C	ЛОЖЬ
UDInt	12345678	16	6	N/A	3	Current Temp = xxxxxxxxxx C	ЛОЖЬ

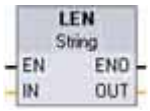
6.2.3.3 Операции со строками

Ваша управляющая программа может использовать следующие операции со строками и символами для формирования сообщений оператору и протоколирования процесса.

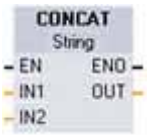
Общие ошибки для всех операций со строками

Операции со строками, при исполнении которых возникают показанные ниже недопустимые состояния, приводят к тому, что ENO = 0 и выводится пустая строка. Ошибочные состояния, которые возникают при определенных операциях, приведены под описанием соответствующей операции.

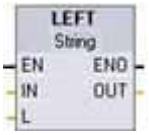
ENO	Условие	OUT
0	Текущая длина IN1 превышает максимальную длину IN1, или текущая длина IN2 превышает максимальную длину IN2 (неправильная строка)	Текущая длина устанавливается в 0
	Максимальная длина IN1, IN2 или OUT не помещается в выделенную область памяти	
	Максимальная длина IN1, IN2 или OUT равна 0 или 255 (недопустимая длина)	



LEN: Получить длину строки



CONCAT: Соединить две строки

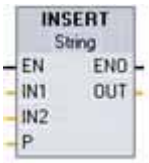


LEFT: Получить левую подстроку из строки

RIGHT: Получить правую подстроку из строки

MID: Получить среднюю подстроку из строки

FIND: Найти подстроку или символ в строке



INSERT: Вставить подстроку в строку

DELETE: Удалить подстроку из строки

REPLACE: Заменить подстроку в строке

Команда LEN

Параметр	Тип параметра	Тип данных	Описание
IN	IN	String	Вводимая строка
OUT	OUT	UInt	Число допустимых символов в строке IN

LEN (длина строки) дает текущую длину строки IN на выходе OUT. Пустая строка имеет длину ноль. В следующей таблице показаны коды условий для этой команды.

ENO	Условие	OUT
1	Недопустимые условия для строки отсутствуют	Действительная длина строки

Команда CONCAT

Параметр	Тип параметра	Тип данных	Описание
IN1	IN	String	Вводимая строка 1
IN2	IN	String	Вводимая строка 2
OUT	OUT	String	Комбинированная строка (строка 1 + строка 2)

CONCAT (объединить строки) соединяет параметры строк IN1 и IN2, формируя одну строку, представленную в параметре OUT. После объединения строка IN1 является левой частью, а строка IN2 правой частью объединенной строки. В следующей таблице показаны коды условий для этой команды.

ENO	Условие	OUT
1	Ошибки не обнаружены	Допустимые символы
0	Результирующая строка после объединения больше максимальной длины строки OUT	Символы результирующей строки копируются, пока не будет достигнута максимальная длина OUT

Команда LEFT

Параметр	Тип параметра	Тип данных	Описание
IN	IN	String	Вводимая строка
L	IN	Int	Длина подстроки, которая должна быть создана, используя самые левые L символов строки IN
OUT	OUT	String	Выводимая строка

Команда LEFT (левая подстрока) выдает подстроку, состоящую из первых L символов строки параметра IN.

- Если L больше, чем текущая длина строки IN, то вся строка IN выводится в параметре OUT.
- Если вводится пустая строка, то в OUT тоже выводится пустая строка.

В следующей таблице показаны коды условий для этой команды.

ENO	Условие	OUT
1	Ошибки не обнаружены	Допустимые символы
0	L меньше или равно 0	Текущая длина устанавливается в 0
	Длина подстроки (L), подлежащей копированию, больше максимальной длины строки OUT	Символы копируются, пока не будет достигнута максимальная длина строки OUT

Команда RIGHT

Параметр	Тип параметра	Тип данных	Описание
IN	IN	String	Вводимая строка
L	IN	Int	Длина подстроки, которая должна быть создана, используя самые правые L символов строки IN
OUT	OUT	String	Выводимая строка

Команда RIGHT (правая подстрока) выдает подстроку, состоящую из последних L символов строки параметра IN.

- Если L больше, чем текущая длина строки IN, то вся строка IN выводится в параметре OUT.
- Если вводится пустая строка, то в OUT тоже выводится пустая строка.

В следующей таблице показаны коды условий для этой команды.

ENO	Условие	OUT
1	Ошибки не обнаружены	Допустимые символы
0	L меньше или равно 0	Текущая длина устанавливается в 0
	Длина подстроки (L), подлежащей копированию, больше максимальной длины строки OUT	Символы копируются, пока не будет достигнута максимальная длина строки OUT

Команда MID

Параметр	Тип параметра	Тип данных	Описание
IN	IN	String	Вводимая строка
L	IN	Int	Длина подстроки, которая должна быть создана, используя L символов строки IN, начиная с позиции P
P	IN	Int	Положение первого символа подстроки, подлежащей копированию: P= 1, для позиции первого символа строки IN
OUT	OUT	String	Выводимая строка

Команда MID (средняя подстрока) выводит среднюю часть строки. Средняя подстрока имеет длину L символов и начинается с позиции P (включительно).

Если сумма L и P превышает текущую длину строки в параметре IN, то выводится подстрока, начинающаяся с позиции P и продолжающаяся до конца строки IN. В следующей таблице показаны коды условий для этой команды.

ENO	Условие	OUT
1	Ошибки не обнаружены	Допустимые символы
0	L или P меньше или равно 0	Текущая длина устанавливается в 0
	P больше, чем максимальная длина IN	
	Длина подстроки (L), подлежащей копированию, больше максимальной длины строки OUT	Символы копируются, начиная с позиции P, пока не будет достигнута максимальная длина OUT

Команда DELETE

Параметр	Тип параметра	Тип данных	Описание
IN	IN	String	Вводимая строка
L	IN	Int	Число символов, подлежащих удалению
P	IN	Int	Положение первого символа, подлежащего удалению: Первый символ строки IN находится в позиции 1
OUT	OUT	String	Выводимая строка

Команда DELETE (удалить подстроку) удаляет L символов из строки IN. Удаление символов начинается с позиции P (включительно), а оставшаяся подстрока выводится в параметре OUT.

- Если L равно нулю, то в OUT выводится введенная строка.
- Если сумма L и P больше, чем длина введенной строки, то строка удаляется до конца.

В следующей таблице показаны коды условий для этой команды.

ENO	Условие	OUT
1	Ошибки не обнаружены	Допустимые символы
0	P больше, чем текущая длина IN	IN копируется в OUT без удаления символов
	L меньше, чем 0, или P меньше или равно 0	Текущая длина устанавливается в 0
	Результирующая строка после удаления символов больше максимальной длины строки OUT	Символы результирующей строки копируются, пока не будет достигнута максимальная длина OUT

Команда INSERT

Параметр	Тип параметра	Тип данных	Описание
IN1	IN	String	Вводимая строка 1
IN2	IN	String	Вводимая строка 2
P	IN	Int	Положение последнего символа в строке IN1 перед точкой вставки для строки IN2. Первый символ строки IN1 находится в позиции 1.
OUT	OUT	String	Результирующая строка

Команда INSERT (вставить подстроку) вставляет строку IN2 в строку IN1. Вставка начинается после символа, находящегося в позиции P. В следующей таблице показаны коды условий для этой команды.

ENO	Условие	OUT
1	Ошибки не обнаружены	Допустимые символы
0	P больше, чем длина IN1	IN2 присоединяется к IN1 непосредственно после последнего символа IN1
	P меньше или равно 0	Текущая длина устанавливается в 0
	Результирующая строка после вставки больше максимальной длины строки OUT	Символы результирующей строки копируются, пока не будет достигнута максимальная длина OUT

Команда REPLACE

Параметр	Тип параметра	Тип данных	Описание
IN1	IN	String	Вводимая строка
IN2	IN	String	Строка с заменяющими символами
L	IN	Int	Число символов, подлежащих замене
P	IN	Int	Положение первого символа, подлежащего замене
OUT	OUT	String	Результирующая строка

Команда REPLACE (заменить подстроку) заменяет L символов в строке параметра IN1. Замена начинается с символа строки IN1, находящегося в положении P (включительно), заменяющими символами их строки параметра IN2.

- Если параметр L равен нулю, то строка IN2 вставляется в позицию P строки IN1 без удаления символов из строки IN1.
- Если P равно 1, то первые L символов строки IN1 заменяются символами строки IN2.

В следующей таблице показаны коды условий для этой команды.

ENO	Условие	OUT
1	Ошибки не обнаружены	Допустимые символы
0	P больше, чем длина IN1	IN2 присоединяется к IN1 непосредственно после последнего символа IN1
	Позиция P находится внутри IN1, но в IN1 остается менее чем L символов	IN2 заменяет конечные символы IN1, начиная с позиции P
	L меньше, чем 0, или P меньше или равно 0	Текущая длина устанавливается в 0
	Результирующая строка после замены больше максимальной длины строки OUT	Символы результирующей строки копируются, пока не будет достигнута максимальная длина OUT

Команда FIND

Параметр	Тип параметра	Тип данных	Описание
IN1	IN	String	Искать внутри этой строки
IN2	IN	String	Искать эту строку
OUT	OUT	Int	Положение в строке IN1 первого искомого совпадения

Команда FIND (искать подстроку) выдает положение внутри строки IN1 подстроки или символа, указанного в параметре IN2. Поиск начинается слева. Позиция первого появления строки IN2 выводится в OUT. Если строка IN2 не найдена в строке IN1, то выводится ноль. В следующей таблице показаны коды условий для этой команды.

ENO	Условие	OUT
1	Ошибки не обнаружены	Действительная позиция символа
0	IN2 больше, чем IN1	Позиция символа устанавливается в 0

6.2.4 Команды управления программой

6.2.4.1 Сброс контроля времени цикла

Команда RE_TRIGR (перезапустить контроль времени цикла) используется для увеличения максимально допустимого времени цикла, прежде чем таймер контроля времени цикла сгенерирует ошибку.

Используйте команду RE_TRIGR для перезапуска таймера контроля времени цикла во время исполнения цикла. Благодаря этому максимально допустимое время цикла увеличивается еще на один интервал максимального времени цикла с момента последнего исполнения функции RE_TRIGR.

CPU ограничивает использование команды RE_TRIGR программным циклом, например, OB1, и функциями, которые вызываются из программного цикла. Это значит, что таймер контроля времени цикла сбрасывается, и ENO = EN, если RE_TRIGR из любого OB, содержащегося в списке OB программного цикла.

ENO = ЛОЖЬ, и таймер контроля времени не сбрасывается, если RE_TRIGR выполняется из OB запуска, OB прерываний или OB ошибок.

Установка максимального времени цикла ПЛК

Вы можете установить значение для максимального времени цикла в конфигурации устройств ПЛК через "Cycle time [Время цикла]".

Контроль времен цикла	Минимальное значение	Максимальное значение	Значение по умолчанию
Максимальное время цикла	1 мс	6000 мс	150 мс

Контроль времени исполнения

Если максимальное время цикла истекает до того, как цикл завершен, то генерируется ошибка. Если кодовый блок обработки ошибок OB 80 включен в программу пользователя, то ПЛК исполняет OB 80, в который вы можете ввести программную логику для формирования специальной реакции на ошибку. Если OB 80 не включен в программу, то первое превышение лимита времени игнорируется.

Если в том же самом программном цикле происходит второе превышение максимального времени цикла (2-кратное значение максимального времени цикла), то генерируется ошибка, которая заставляет ПЛК перейти в состояние STOP.

В состоянии STOP исполнение вашей программы прекращается, однако системные коммуникации ПЛК и системная диагностика продолжаются.

6.2.4.2 Команда остановки цикла

Команда STP (остановить цикл сканирования ПЛК) переводит ПЛК в состояние STOP. Когда ПЛК находится в состоянии STOP, исполнение вашей программы и физическое обновление образа процесса прекращаются.

Дальнейшую информацию вы найдете под заголовком: Конфигурирование поведения выходных значений для состояния STOP CPU (стр. 56)

Если EN = ИСТИНА, то ПЛК переходит в состояние STOP, исполнение программы прекращается, и состояние ENO не имеет значения. В противном случае EN = ENO = 0.

6.2.4.3 Команды получения информации об ошибках

Команды получения информации об ошибках предоставляют информацию об ошибках при исполнении программных блоков. Если вы вставите команду GetError или GetErrorID в свой кодовый блок, то вы можете обрабатывать ошибки программы в своем программном блоке.

GET_ERROR

Команда GET_ERROR показывает, что произошла ошибка при исполнении программного блока, и заполняет заранее определенную структуру данных ошибки подробной информацией об ошибке.

Параметр	Тип данных	Описание
ERROR	ErrorStruct	Структура данных об ошибке: Вы можете переименовать эту структуру, но не элементы внутри нее.

Элемент данных ErrorStruct	Тип данных	Описание
ERROR_ID	Word	Идентификатор ошибки
FLAGS	Byte	Всегда 0.
REACTION	Byte	Реакция на ошибку: • 0 = игнорировать; ничего не записано (ошибка записи) • 1 = заменить: для входного значения был использован 0 (ошибка чтения) • 2 = пропустить команду
BLOCK_TYPE	Byte	Тип блока, где произошла ошибка: • 1 = OB • 2 = FC • 3 = FB
PAD_0	Byte	Внутренне заполняемый байт для целей выравнивания, равен 0

Элемент данных ErrorStruct	**Тип данных**	**Описание**
CODE_BLOCK_NUMBER	UInt	Номер блока, в котором произошла ошибка
ADDRESS	UDInt	Внутренний адрес в памяти для команды, в которой произошла ошибка
MODE	Byte	Внутреннее отображение того, как будут интерпретироваться оставшиеся поля, предназначенные для использования STEP 7 Basic
PAD_1	Byte	Внутренне заполняемый байт для целей выравнивания; не используется, равен 0
OPERAND_NUMBER	UInt	Число внутренних операндов команды
POINTER_NUMBER_ LOCATION	UInt	(A) Внутренний адрес указателя команды
SLOT_NUMBER_SCOPE	UInt	(B) Внутренний адрес в памяти
AREA	Byte	(C) Область памяти, на которую делается ссылка при возникновении ошибки: • L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE • I: 16#81 • Q: 16#82 • M: 16#83 • DB: 16#84, 85, 8A, 8B
PAD_2	Byte	Внутренне заполняемый байт для целей выравнивания; не используется, равен 0
DB_NUMBER	UInt	(D) DB, на который делается ссылка, когда происходит ошибка DB, иначе 0
OFFSET	UDInt	(E) Битовое смещение, на которое делается ссылка при возникновении ошибки (пример: 12 = байт 1, бит 4)

GET_ERR_ID

Команда GET_ERR_ID указывает, что произошла ошибка при исполнении программного блока, и сообщает ID (идентификационный код) ошибки.

Параметр	Тип данных	Описание
ID	Word	Значения идентификатора ошибки для элемента ErrorStruct ERROR_ID

ERROR_ID шестнадцатеричный	ERROR_ID десятичный	Ошибка исполнения программного блока
2503	9475	Ошибка – неинициализированный указатель
2522	9506	Ошибка чтения – операнд вне допустимого диапазона
2523	9507	Ошибка записи – операнд вне допустимого диапазона
2524	9508	Ошибка чтения – недействительная область
2525	9509	Ошибка записи – недействительная область
2528	9512	Ошибка чтения при выравнивании данных (неверное выравнивание битов)
2529	9513	Ошибка записи при выравнивании данных (неверное выравнивание битов)
2530	9520	DB защищен от записи
253A	9530	Глобальный DB не существует
253C	9532	Неправильная версия или FC не существует
253D	9533	Команда не существует
253E	9534	Неправильная версия или FB не существует
253F	9535	Команда не существует
2575	9589	Ошибка глубины вложения программ
2576	9590	Ошибка выделения локальных данных
2942	10562	Физический вход не существует
2943	10563	Физический выход не существует

Принцип действия

По умолчанию CPU реагирует на ошибку исполнения блока регистрацией ошибки в диагностическом буфере и переходом в состояние STOP. Однако, если вы поместите в кодовый блок одну или несколько команд GET_ERROR или ERR_ID, то этот блок в состоянии обрабатывать ошибки в самом блоке. В этом случае CPU не переходит в STOP и не регистрирует ошибку в диагностическом буфере. Вместо этого информация об ошибке сообщается на выходе команды GET_ERROR или GET_ERR_ID. Вы можете прочитать подробную информацию об ошибке с помощью команды GET_ERROR или только прочитать идентификатор ошибки с помощью команды GET_ERR_ID. Обычно первая ошибка является самой важной, а последующие ошибки являются только следствие первой ошибки.

Первое исполнение команды GET_ERROR или GET_ERR_ID в блоке возвращает первую ошибку, обнаруженную во время исполнения блока. Эта ошибка могла возникнуть в любом месте между началом блока и исполнением GET_ERROR или GET_ERR_ID. Следующие исполнения GET_ERROR или GET_ERR_ID возвращают первую ошибку после предыдущего исполнения GET_ERROR или GET_ERR_ID. История ошибок не сохраняется, и исполнение любой из этих команд побуждает ПЛК к регистрации следующей ошибки.

Тип данных ErrorStruct, используемый командой GET_ERROR, может быть вставлен в редакторе блока данных и в редакторах интерфейса блоков, так что логика вашей программы получает доступ к этим значениям. Для добавления этой структуры выберите ErrorStruct из ниспадающего списка типов данных. Вы можете создать несколько структур ErrorStruct, используя уникальные имена. Элементы ErrorStruct не могут быть переименованы.

Сбойное состояние, указываемое с помощью ENO

Если EN = ИСТИНА и исполняется GET_ERROR или GET_ERR_ID, то:

- ENO = ИСТИНА указывает, что во время исполнения кодового блока произошла ошибка и имеются данные об ошибке
- ENO = ЛОЖЬ указывает, что во время исполнения кодового блока не было ошибок

Вы можете подключить к ENO программу реагирования на ошибку, которая активизируется после возникновения ошибки. Если ошибка существует, то выходной параметр сохраняет данные об ошибке там, где ваша программа может к ним обратиться.

Команды GET_ERROR и GET_ERR_ID могут использоваться для передачи информации об ошибке из исполняющегося в данный момент блока (вызванного блока) в вызывающий блок. Поместите эту команду в последнюю сеть вызываемого программного блока, чтобы сообщить конечное состояние при исполнении вызванного блока.

6.2.5 Коммуникационные операции

6.2.5.1 Обмен данными с открытой сетью Open Ethernet

Обмен данными с открытой сетью Open Ethernet с использованием автоматического соединения и рассоединения (TSEND_C и TRCV_C)

Указание

Обработка команд TSEND_C и TRCV_C может занимать неопределенное количество времени. Чтобы обеспечить обработку этих команд в каждом цикле сканирования, вызывайте их из главного программного цикла, например, из OB программного цикла или из кодового блока, который вызывается из программного цикла. **Не** вызывайте эти команды из OB аппаратных прерываний, OB прерываний с задержкой, OB циклических прерываний, OB прерываний из-за ошибки или OB запуска.

За информацией о передаче данных с помощью этих команд обратитесь к разделу о согласованности данных (стр. 96).

Описание команды TSEND_C

Команда TSEND_C устанавливает связь с партнером через TCP или ISO on TCP, посылает данные и может завершить соединение. После установления и создания соединения оно автоматически поддерживается и контролируется с помощью CPU. Команда TSEND_C объединяет в себе функции команд TCON, TDISCON и TSEND.

Минимальный размер данных, которые вы можете передать с помощью команды TSEND_C, составляет один байт.

Указание

Настройка по умолчанию параметра LEN (LEN = 0) использует параметр DATA для определения длины передаваемых данных. Обеспечьте, чтобы параметр DATA, передаваемый командой TSEND_C, имел такой же размер, что и параметр DATA команды TRCV_C.

Следующие функции описывают действие команды TSEND_C:

- Для установления соединения команда TSEND_C должна исполняться с параметром CONT = 1.
- После успешного установления соединения TSEND_C устанавливает на один цикл параметр DONE.
- Для завершения соединения используется TSEND_C с параметром CONT = 0. Соединение прерывается немедленно. Это оказывает воздействие также на принимающую станцию. Соединение завершается и там, и данные внутри принимающего буфера могут быть потеряны.
- Для передачи данных через существующее соединение команда TSEND_C должна исполняться при нарастающем фронте на REQ. После успешной передачи TSEND_C устанавливает на один цикл в 1 параметр DONE.
- Для установления соединения и передачи данных команда TSEND_C должна исполняться с CONT =1 и REQ = 1. После успешной передачи TSEND_C устанавливает на один цикл в 1 параметр DONE.

Описание TRCV_C

Команда TRCV_C устанавливает связь с партнерским CPU через TCP или ISO on TCP, получает данные и может завершить соединение. После установления и создания соединения оно автоматически поддерживается и контролируется с помощью CPU. Команда TRCV_C объединяет в себе функции команд TCON, TDISCON, и TRCV.

Минимальный размер данных, которые вы можете принять с помощью команды TRCV_C, составляет один байт. Команда TRCV_C не поддерживает передачу булевых данных или булевых массивов.

Указание

Настройка по умолчанию параметра LEN (LEN = 0) использует параметр DATA для определения длины передаваемых данных. Обеспечьте, чтобы параметр DATA, передаваемый командой TSEND_C, имел такой же размер, что и параметр DATA команды TRCV_C.

Следующие функции описывают действие команды TRCV_C:

- Для установления соединения команда TRCV_C должна исполняться с параметром CONT = 1.
- Для получения данных исполняйте TRCV_C с параметром EN_R = 1. TRCV_C получает данные непрерывно, когда параметры EN_R = 1 и CONT = 1.
- Для завершения соединения используется TRCV_C с параметром CONT = 0. Соединение прерывается немедленно, и данные могут быть потеряны.

Режимы приема

Команда TRCV_C работает в таких же режимах, что и команда TRCV. В следующей таблице показано, как данные записываются в область приема.

Вариант протокола	Ввод данных в область приема	Параметр "connection_type [тип соединения]"
TCP	Прием данных заданной длины	B#16#11
ISO on TCP	Под управлением протокола	B#16#12

Указание

Из-за асинхронной обработки команды TSEND_C вы должны поддерживать согласованность данных в области передачи, пока параметр DONE или параметр ERROR не примет значение ИСТИНА.

Для команды TSEND_C состояние ИСТИНА параметра DONE означает, что данные были переданы успешно. Это не означает, что CPU партнера по соединению фактически прочитал принимающий буфер.

Из-за асинхронной обработки команды TRCV_C данные в области приема согласованы только в том случае, если параметр DONE = 1.

В следующей таблице показаны отношения между параметрами BUSY, DONE и ERROR.

BUSY	DONE	ERROR	Описание
ИСТИНА	Не имеет значения	Не имеет значения	Задание обрабатывается.
ЛОЖЬ	ИСТИНА	ЛОЖЬ	Задание успешно завершено.
ЛОЖЬ	ЛОЖЬ	ИСТИНА	Задание завершено с ошибкой. Причину ошибки можно найти в параметре STATUS.
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	Новое задание не назначено.

Параметры TSEND_C

Параметр	Тип параметра	Тип данных	Описание
REQ	INPUT	Bool	Управляющий параметр REQ запускает задание на передачу при нарастающем фронте через соединение, описанное в CONNECT.
CONT	INPUT	Bool	• 0: рассоединить • 1: установить и удерживать соединение
LEN	INPUT	Int	Максимальное число байтов, подлежащих передаче. (Значение по умолчанию = 0, что означает, что параметр DATA определяет длину данных, подлежащих передаче.).
CONNECT	IN_OUT	TCON-Param	Указатель на описание соединения
DATA	IN_OUT	Variant	Область передачи; содержит адрес и длину данных, подлежащих передаче.
COM_RST	IN_OUT	Bool	• 1: Полный перезапуск функционального блока, существующее соединение разрывается.
DONE	OUTPUT	Bool	• 0: Задание еще не запущено или еще исполняется. • 1: Задание исполнено с ошибкой.
BUSY	OUTPUT	Bool	• 0: Задание завершено. • 1: Задание еще не завершено. Новое задание не может быть запущено.
ERROR	OUTPUT	Bool	• 1: Во время обработки произошла ошибка. Параметр STATUS предоставляет подробную информацию о типе ошибки.
STATUS	OUTPUT	Word	Информация об ошибке

Параметры TRCV_C

Параметр	Тип параметра	Тип данных	Описание
EN_R	IN	Bool	Управляющий параметр, разблокированный для приема: Когда EN_R = 1, команда TRCV_C готова к приему. Задание на прием обрабатывается.
CONT	IN	Bool	Управляющий параметр CONT: • 0: рассоединить • 1: установить и удерживать соединение
LEN	IN	Int	Длина области приема в байтах. (Значение по умолчанию = 0, что означает, что параметр DATA определяет длину данных, подлежащих передаче.).
CONNECT	IN_OUT	TCON-Param	Указатель на описание соединения
DATA	IN_OUT	Variant	Область приема содержит начальный адрес и максимальную длину принимаемых данных.
COM_RST	IN_OUT	Bool	• 1: Полный перезапуск функционального блока; существующее соединение разрывается.
DONE	OUT	Bool	• 0: Задание еще не запущено или еще исполняется. • 1: Задание исполнено с ошибкой.
BUSY	OUT	Bool	• 0: Задание завершено. • 1: Задание еще не завершено. Новое задание не может быть запущено.
ERROR	OUT	Bool	• 1: Во время обработки произошла ошибка. Параметр STATUS предоставляет подробную информацию о типе ошибки.
STATUS	OUT	Word	Информация об ошибке
RCVD_LEN	OUT	Int	Количество фактически принятых данных, в байтах

Параметры Error и Status

ERROR	STATUS (W#16#...)	Описание
0	0000	Задание исполнено с ошибкой
0	7000	Обработки задания не происходит
0	7001	Запуск обработки задания, установление соединения, ожидание партнера по соединению
0	7002	Происходит прием или передача данных
0	7003	Соединение завершается
0	7004	Соединение установлено и контролируется, обработки задания не происходит
1	8085	Параметр LEN превышает максимально допустимое значение
1	8086	Параметр CONNECT выходит за пределы допустимого диапазона
1	8087	Достигнуто максимальное число соединений; дополнительные соединения невозможны
1	8088	Параметр LEN превышает область памяти, указанную в параметре DATA; принимающая область памяти слишком мала
1	8089	Параметр CONNECT не указывает на блок данных.
1	8091	Превышена максимальная глубина вложения
1	809A	Параметр CONNECT указывает на поле, которое не соответствует длине в описании соединения.
1	809B	local_device_id в описании соединения не согласовывается с CPU.
1	80A1	Коммуникационная ошибка: • Указанное соединение еще не установлено • Указанное соединение в настоящее время завершается; передача через это соединение невозможна • Интерфейс снова инициализируется
1	80A3	Делается попытка завершить несуществующее соединение
1	80A4	IP-адрес соединения с удаленным партнером неверен. Например, IP-адрес удаленного партнера совпадает с IP-адресом локального партнера.
1	80A7	Коммуникационная ошибка: вы вызвали TDISCON, прежде чем был завершен TCON (TDISCON должен сначала полностью завершить соединение, указанное в ID)
1	80B2	Параметр CONNECT указывает на блок данных, который был сгенерирован с ключевым словом UNLINKED
1	80B3	Несовместимые параметры: • Ошибка в описании соединения • Локальный порт (параметр local_tsap_id) уже присутствует в описании другого соединения • ID в описании соединения отличен от ID, указанного в качестве параметра

ERROR	STATUS (W#16#...)	Описание
1	80B4	При использовании ISO on TCP (connection_type = B#16#12) для установления пассивного соединения код ошибки 80B4 предупреждает вас, что введенный TSAP не соответствует одному из следующих требований к адресу: • Если локальный TSAP имеет длину 2 и значение ID для первого байта, равное E0 или E1 (шестнадцатеричное), то второй байт должен быть 00 или 01. • Если локальный TSAP имеет длину 3 или больше и значение ID для первого байта E0 или E1 (шестнадцатеричное), то второй байт должен быть 00 или 01, а все остальные байты должны быть действительными символами ASCII. • Если локальный TSAP имеет длину 3 или больше, и первый байт ID TSAP не имеет значения E0 или E1 (шестнадцатеричное), то все байты ID TSAP должны быть действительными символами ASCII. Действительными символами ASCII являются значения байтов от 20 до 7E (шестнадцатеричное).
1	80C3	Все ресурсы соединений используются.
1	80C4	Коммуникационная ошибка, связанная со временем: • Соединение не может быть установлено в настоящее время • Интерфейс получает новые параметры • Сконфигурированное соединение в настоящее время удаляется командой TDISCON
1	8722	Параметр CONNECT: Недействительная область источника: область не существует в DB
1	873A	Параметр CONNECT: Доступ к описанию соединения невозможен (напр., DB отсутствует)
1	877F	Параметр CONNECT: Внутренняя ошибка, например, недействительная ссылка на ANY

Обмен данными с открытой сетью Open Ethernet с использованием управления соединением и рассоединением

Указание

Обработка команд TSEND_C и TRCV_C может занимать неопределенное количество времени. Чтобы обеспечить обработку этих команд в каждом цикле сканирования, вызывайте их из главного программного цикла, например, из OB программного цикла или из кодового блока, который вызывается из программного цикла. **Не** вызывайте эти команды из OB аппаратных прерываний, OB прерываний с задержкой, OB циклических прерываний, OB прерываний из-за ошибки или OB запуска.

Обмен данными через Ethernet с помощью протоколов TCP и ISO on TCP

Обменом данными управляют в программе следующие команды:

- TCON устанавливает соединение.
- TSEND и TRCV передают и принимают данные.
- TDISCON разрывает соединение.

Минимальный размер данных, которые вы можете передать или принять с помощью команд TSEND и TRCV, составляет один байт. Команда TRCV не поддерживает передачу булевых данных или булевых массивов. Дополнительную информацию вы найдете в разделе о согласованности данных (стр. 96).

Указание

Настройка по умолчанию параметра LEN (LEN = 0) использует параметр DATA для определения длины передаваемых данных. Обеспечьте, чтобы параметр DATA, передаваемый командой TSEND, имел такой же размер, что и параметр DATA команды TRCV.

Оба партнера по обмену данными выполняют команду TCON, чтобы создать и установить коммуникационное соединение. С помощью параметров вы указываете активный и пассивный концевой пункт обмена данными. После создания и установления соединения оно автоматически поддерживается и контролируется посредством CPU.

Если соединение прекращается, например, из-за обрыва провода или удаленного партнера по обмену данными, то активный партнер пытается вновь установить сконфигурированное соединение. Вы не должны вновь выполнять команду TCON.

Если выполняется команда TDISCON или CPU перешел в состояние STOP, то существующее соединение завершается, и созданное соединение удаляется. Для создания и восстановления соединения вы должны снова выполнить команду TCON.

Описание функционирования

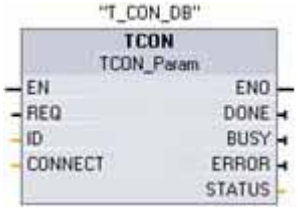
Команды TCON, TDISCON, TSEND и TRCV работают асинхронно, это значит, что обработка задания распространяется на несколько исполнений команд.

Например, вы запускаете задание для создания и установления соединения путем исполнения команды TCON с параметром REQ = 1. Затем вы используете дополнительные исполнения TCON для контроля выполнения задания и проверки его завершения с помощью параметра DONE.

В следующей таблице показаны отношения между BUSY, DONE и ERROR. Используйте эту таблицу для проверки текущего состояния задания.

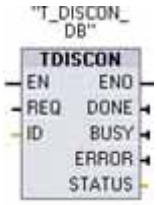
BUSY	DONE	ERROR	Описание
ИСТИНА	не имеет значения	не имеет значения	Задание обрабатывается.
ЛОЖЬ	ИСТИНА	ЛОЖЬ	Задание успешно завершено.
ЛОЖЬ	ЛОЖЬ	ИСТИНА	Задание завершено с ошибкой. Причину ошибки можно найти в параметре STATUS.
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	Новое задание не назначено.

Команда TCON



Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Управляющий параметр REQUEST запускает задание для установления соединения, указанного в ID. Задание запускается при нарастающем фронте.
ID	IN	CONN_OUC (Word)	Ссылка на соединение, которое должно быть установлено с удаленным партнером или между программой пользователя и коммуникационным уровнем операционной системы. ID должен быть идентичен соответствующему ID параметра в описании локального соединения. Диапазон значений: от W#16#0001 до W#16#0FFF
CONNECT	IN_OUT	TCON-Param	Указатель на описание соединения
DONE	OUT	Bool	Параметр состояния DONE: • 0: Задание еще не запущено или еще исполняется • 1: Задание исполнено с ошибкой
BUSY	OUT	Bool	BUSY = 1: Задание еще не завершено BUSY = 0: Задание завершено
ERROR	OUT	Bool	Параметр состояния ERROR: ERROR = 1: При обработке задания произошла ошибка. Параметр STATUS предоставляет подробную информацию о типе ошибки.
STATUS	OUT	Word	Параметр состояния STATUS: Информация об ошибке

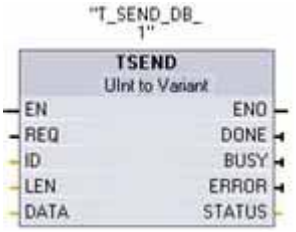
Команда TDISCON



TCP и ISO on TCP: Команда TDISCON завершает коммуникационное соединение от CPU к партнеру по обмену данными.

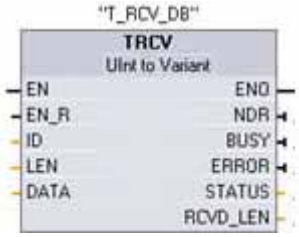
Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Управляющий параметр REQUEST запускает задание для установления соединения, указанного в ID. Задание запускается при нарастающем фронте.
ID	IN	CONN_OUC (Word)	Ссылка на соединение с удаленным партнером или между программой пользователя и коммуникационным уровнем операционной системы, которое должно быть завершено. ID должен быть идентичен соответствующему ID параметра в описании локального соединения. Диапазон значений: от W#16#0001 до W#16#0FFF
DONE	OUT	Bool	Параметр состояния DONE: • 0: Задание еще не запущено или еще исполняется • 1: Задание исполнено с ошибкой
BUSY	OUT	Bool	BUSY = 1: Задание еще не завершено BUSY = 0: Задание завершено
ERROR	OUT	Bool	ERROR = 1: Во время обработки произошла ошибка.
STATUS	OUT	Word	Код ошибки

Команда TSEND



Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Управляющий параметр REQUEST запускает задание на передачу при нарастающем фронте. Данные передаются из области, определяемой параметрами DATA и LEN.
ID	IN	CONN_OUC (Word)	Ссылка на соответствующее соединение. ID должен быть идентичен соответствующему ID параметра в описании локального соединения. Диапазон значений: от W#16#0001 до W#16#0FFF
LEN	IN	Int	Максимальное число байтов, подлежащих передаче этим заданием
DATA	IN_OUT	Variant	Указатель на область данных, подлежащую передаче: Область передатчика; содержит адрес и длину. Адрес относится: • к образу процесса на входах • к образу процесса на выходах • к битовой памяти • к блоку данных
DONE	OUT	Bool	Параметр состояния DONE: • 0: Задание еще не запущено или еще исполняется. • 1: Задание исполнено с ошибкой.
BUSY	OUT	Bool	• BUSY = 1: Задание еще не завершено. Новое задание не может быть запущено. • BUSY = 0: Задание завершено.
ERROR	OUT	Bool	Параметр состояния ERROR: ERROR = 1: Во время обработки произошла ошибка. Параметр STATUS предоставляет подробную информацию о типе ошибки
STATUS	OUT	Word	Параметр состояния STATUS: Информация об ошибке

Команда TRCV



Параметр	Тип параметра	Тип данных	Описание
EN_R	IN	Bool	Управляющий параметр, разблокированный для приема: При EN_R = 1 команда TRCV готова к приему. Задание на прием обрабатывается.
ID	IN	CONN_OUC (Word)	Ссылка на соответствующее соединение. ID должен быть идентичен соответствующему ID параметра в описании локального соединения. Диапазон значений: от W#16#0001 до W#16#0FFF
LEN	IN	Int	Длина области приема в байтах (Значение по умолчанию = 0, это значит, что параметр DATA определяет длину данных, подлежащих приему.).
DATA	IN_OUT	Variant	Указатель на принимаемые данные: Область приема, которая содержит адрес и длину. Адрес относится: • к образу процесса на входах • к образу процесса на выходах • к битовой памяти • к блоку данных
NDR	OUT	Bool	Параметр состояния NDR: • NDR = 0: Задание еще не запущено или еще исполняется. • NDR = 1: Задание успешно завершено.
BUSY	OUT	Bool	• BUSY = 1: Задание еще не завершено. Новое задание не может быть запущено. • BUSY = 0: Задание завершено.
ERROR	OUT	Bool	ERROR=1: Во время обработки произошла ошибка. Параметр STATUS предоставляет подробную информацию о типе ошибки.
STATUS	OUT	Word	Информация об ошибке
RCVD_LEN	OUT	Int	Количество фактически принятых данных, в байтах

Область приема

Команда TRCV записывает принимаемые данные в область приема, которая определяется следующими двумя переменными:

- Указатель на начало области
- Длина области

Указание

Настройка по умолчанию параметра LEN (LEN = 0) использует параметр DATA для определения длины передаваемых данных. Обеспечьте, чтобы параметр DATA, передаваемый командой TSEND, имел такой же размер, что и параметр DATA команды TRCV.

В следующей таблице показано, как команда TRCV вводит принимаемые данные в область приема.

Вариант протокола	Ввод данных в область приема	Параметр Тип соединения
TCP	Прием данных указанной длины	B#16#11
ISO on TCP	Под управлением протокола	B#16#12

Как только данные из задания приняты, TRCV передает их в область приема и устанавливает NDR в 1.

Коды условий для TCON

ERROR	STATUS (W#16#...)	Объяснение
0	0000	Соединение было успешно установлено
0	7000	Обработки задания не происходит
0	7001	Запуск обработки задания, установление соединения
0	7002	Последующий вызов (REQ не имеет значения), соединение устанавливается
1	8086	Параметр ID находится вне допустимого диапазона.
1	8087	Достигнуто максимальное число соединений; дополнительные соединения невозможны
1	809B	local_device_id в описании соединения не соответствует CPU.
1	80A1	Соединение или порт уже заняты пользователем
1	80A2	Локальный или удаленный порт занят системой
1	80A3	Делается попытка снова установить уже существующее соединение
1	80A4	IP-адрес удаленного конца соединения недействителен; возможно, он совпадает с локальным IP-адресом
1	80A7	Коммуникационная ошибка: вы выполнили TDISCON, прежде чем была завершена команда TCON. Команда TDISCON должна сначала полностью завершить соединение, указанное в параметре ID.
1	80B3	Противоречивая параметризация: Групповая ошибка для кодов ошибок от W#16#80A0 до W#16#80A2, W#16#80A4, от W#16#80B4 до W#16#80B9

ERROR	STATUS (W#16#...)	Объяснение
1	80B4	При использовании ISO on TCP (connection_type = B#16#12) для установления пассивного соединения код ошибки 80B4 предупреждает вас, что введенный TSAP не соответствует одному из следующих требований к адресу: • Если локальный TSAP имеет длину 2 и значение ID для первого байта, равное E0 или E1 (шестнадцатеричное), то второй байт должен быть 00 или 01. • Если локальный TSAP имеет длину 3 или больше и значение ID для первого байта E0 или E1 (шестнадцатеричное), то второй байт должен быть 00 или 01, а все остальные байты должны быть действительными символами ASCII. • Если локальный TSAP имеет длину 3 или больше, и первый байт ID TSAP не имеет значения E0 или E1 (шестнадцатеричное), то все байты ID TSAP должны быть действительными символами ASCII. Действительными символами ASCII являются значения байтов от 20 до 7E (шестнадцатеричное).
1	80B5	Ошибка в параметре active_est
1	80B6	Ошибка параметризации в параметре connection_type
1	80B7	Ошибка в одном из следующих параметров: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len
1	80B8	Параметр в описании локального соединения и ID параметра различны
1	80C3	Все ресурсы соединений используются.
1	80C4	Коммуникационная ошибка, связанная со временем: • Соединение не может быть установлено в настоящее время. • Интерфейс получает новые параметры. • Сконфигурированное соединение в настоящее время удаляется командой TDISCON.

Коды условий для TDISCON

ERROR	STATUS (W#16#...)	Объяснение
0	0000	Соединение было успешно прекращено
0	7000	Обработки задания не происходит
0	7001	Начало обработки задания, соединение завершается
0	7002	Последующий вызов (REQ не имеет значения), соединение завершается
1	8086	Параметр ID находится вне допустимого диапазона адресов.
1	80A3	Выполняется попытка завершить несуществующее соединение
1	80C4	Коммуникационная ошибка, связанная со временем: Интерфейс получает новые параметры или соединение в настоящее время устанавливается.

Коды условий для TSEND

ERROR	STATUS (W#16#...)	Объяснение
0	0000	Задание на передачу завершено без ошибок
0	7000	Обработки задания не происходит
0	7001	Начало обработки задания, данные отправляются: Во время этой обработки операционная система обращается к данным в области передачи DATA.
0	7002	Последующий вызов (REQ не имеет значения), задание обрабатывается: Во время этой обработки операционная система обращается к данным в области передачи DATA..
1	8085	Параметр LEN превышает максимально допустимое значение.
1	8086	Параметр ID находится вне допустимого диапазона адресов
1	8088	Параметр LEN больше, чем область памяти, указанная в DATA
1	80A1	Коммуникационная ошибка: • Указанное соединение еще не установлено • Указанное соединение в настоящее время завершается. Передача через это соединение невозможна. • Интерфейс снова инициализируется.
1	80C3	Внутренняя нехватка ресурсов: Блок с этим ID уже обрабатывается в другом классе приоритета.
1	80C4	Коммуникационная ошибка, связанная со временем: • Соединение с партнером по обмену данными в настоящее время не может быть установлено. • Интерфейс получает новые параметры или соединение в настоящее время устанавливается.

Коды условий для TRCV

ERROR	STATUS (W#16#...)	Объяснение
0	0000	Приняты новые данные: Текущая длина принятых данных отображается в RCVD_LEN.
0	7000	Блок не готов к приему
0	7001	Блок готов к приему, задание на прием было активизировано.
0	7002	Последующий вызов, задание на прием обрабатывается: Во время этой обработки данные записываются в область приема. Поэтому ошибка может привести к несогласованности данных в области приема.
1	8085	Параметр LEN превышает максимально допустимое значение, или вы изменили параметр LEN или DATA после первого вызова.
1	8086	Параметр ID находится вне допустимого диапазона адресов
1	8088	Область приема слишком мала: Значение LEN больше, чем область приема, указанная в параметре DATA.

ERROR	STATUS (W#16#...)	Объяснение
1	80A1	Коммуникационная ошибка: • Указанное соединение еще не установлено • Указанное соединение в настоящее время завершается. Задание на прием через это соединение невозможно • Интерфейс получает новые параметры.
1	80C3	Внутренняя нехватка ресурсов: Блок с этим ID уже обрабатывается в другом классе приоритета.
1	80C4	Коммуникационная ошибка, связанная со временем: • Соединение с партнером в настоящее время не может быть установлено. • Интерфейс получает новые параметры или соединение в настоящее время устанавливается.

6.2.5.2 Команды для двухточечного соединения

Глава, посвященная двухточечному соединению (Point-to-Point, PTP)(стр. 279) дает подробную информацию о командах PtP и коммуникационных модулях.

6.2.6 Команды прерывания

6.2.6.1 Команды Attach и Detach

С помощью команд ATTACH и DETACH вы можете активизировать и деактивизировать подпрограммы, инициализируемые событиями, вызывающими прерывания.

- Команда ATTACH активизирует исполнение подпрограммы OB прерываний для событий, вызывающих аппаратные прерывания.
- Команда DETACH деактивизирует исполнение подпрограммы OB прерываний для событий, вызывающих аппаратные прерывания.

Параметр	Тип параметра	Тип данных	Описание
OB_NR	IN	Int	Идентификатор организационного блока: Выберите его из имеющихся OB аппаратных прерываний, которые были созданы с помощью опции "Add new block [Добавить новый блок]". Дважды щелкните на поле этого параметра, затем щелкните на вспомогательной пиктограмме, чтобы увидеть имеющиеся OB.
EVENT	IN	DWord	Идентификатор события: Выберите его из имеющихся событий, вызывающих аппаратные прерывания, которые были разблокированы в конфигурации устройств ПЛК для цифровых входов или скоростных счетчиков. Дважды щелкните на поле этого параметра, затем щелкните на вспомогательной пиктограмме, чтобы увидеть имеющиеся события.
ADD (только ATTACH)	IN	Bool	ADD = 0 (по умолчанию): Это событие заменяет все предыдущие назначения событий для этого OB. ADD = 1: Это событие добавляется к предыдущим назначениями событий для этого OB.
RET_VAL	OUT	Int	Код условия выполнения

События, вызывающие аппаратные прерывания

CPU поддерживает следующие события, вызывающие аппаратные прерывания:

- События типа нарастающих фронтов (все встроенные цифровые входы CPU плюс цифровые входы сигнальной платы)
  - Нарастающий фронт возникает, когда цифровой вход переходит из состояния ВЫКЛ в состояние ВКЛ как реакция на изменение сигнала от полевого устройства, подключенного к этому входу.
- События типа падающих фронтов (все встроенные цифровые входы CPU плюс цифровые входы сигнальной платы)
  - Падающий фронт возникает, когда цифровой вход переходит из состояния ВКЛ в состояние ВЫКЛ.
- События типа Текущее значение скоростного счетчика (HSC) = эталонному значению (CV = RV) (HSC 1 … 6)
  - Прерывание CV = RV для HSC генерируется, когда текущее значение переходит от соседнего значения к значению, точно совпадающему с предварительно установленным эталонным значением.
- События типа Изменение направления счета HSC (HSC 1 … 6)
  - Событие типа Изменение направления счета происходит, когда обнаружено, что HSC перешел от прямого счета к обратному или от обратного к прямому.
- События типа Внешний сброс HSC (HSC 1 … 6)
  - Некоторые режимы HSC допускают назначение цифрового входа для внешнего сброса значения счетчика HSC в ноль. Событие типа Внешний сброс происходит для такого HSC, когда этот вход переходит из состояния ВЫКЛ в состояние ВКЛ.

Разблокирование событий, приводящих к аппаратным прерываниям, в конфигурации устройств

Аппаратные прерывания должны быть разблокированы при конфигурировании устройства. Вы должны пометить триггерную кнопку для разблокирования события в конфигурации устройств для канала цифрового ввода или HSC, если вы хотите назначить это событие во время конфигурирования или на этапе исполнения.

Опции триггерных кнопок в конфигурации устройств ПЛК:

- Цифровой вход
  - Разблокировать обнаружение нарастающего фронта
  - Разблокировать обнаружение падающего фронта
- Скоростной счетчик (HSC)
  - Разблокировать этот скоростной счетчик для использования
  - Генерировать прерывание при совпадении значения счетчика с эталонным значением
  - Генерировать прерывание при внешнем сбросе
  - Генерировать прерывание при изменении направления счета

Вставка новых OB аппаратных прерываний в вашу программу

По умолчанию при первом разблокировании события этому событию не ставится в соответствие никакой OB. На это указывает метка "<not connected [не присоединено]>" в конфигурации устройств для "HW interrupt: [Аппаратное прерывание:]". Событию, вызывающему аппаратные прерывания, может быть поставлен в соответствие только OB аппаратных прерываний. Все существующие OB аппаратных прерываний выводятся в ниспадающем списке "HW interrupt:". Если в этом списке OB отсутствуют, то вы должны создать OB типа "Hardware interrupt [Аппаратное прерывание]" следующим образом. В ветви "Program blocks [Программные блоки]" дерева проекта:

1. Дважды щелкните на "Add new block [Добавить новый блок]", выберите "Organization block [Организационный блок] (OB)", а затем "Hardware interrupt".
2. Вы имеете возможность переименовать OB, выбрать язык программирования (LAD или FBD) и задать номер блока (переключитесь в ручной режим и выберите другой номер блока вместо предложенного).
3. Отредактируйте OB и добавьте реакцию программы на возникновение события. Вы можете вызывать из этого OB вложенные FC и FB с глубиной вложенности до четырех.

Параметр OB_NR

Имена всех существующих OB аппаратных прерываний появляются в ниспадающем списке "HW interrupt: [Аппаратное прерывание]" в конфигурации устройств и в ниспадающем списке для параметра OB_NR команд ATTACH / DETACH.

Параметр EVENT

Когда разблокируется событие, вызывающее аппаратное прерывание, этому конкретному событию присваивается по умолчанию уникальное имя. Вы можете изменить имя этого события, редактируя поле ввода "Event name [Имя события]:", но это имя должно быть уникальным. Имена этих событий становятся именами переменных в таблице переменных "Constants [Константы]" и появляются в ниспадающем списке параметра EVENT для блоков команд ATTACH и DETACH. Значением этой переменной является внутренний номер, используемый для идентификации события.

Общий принцип действия

Каждое аппаратное прерывание может быть поставлено в соответствие OB аппаратных прерываний, который будет поставлен в очередь на исполнение, когда происходит событие, вызывающее это аппаратное прерывание. Установление соответствия между OB и событием может происходить во время конфигурирования или во время исполнения.

Вы можете назначать или отменять назначение OB разблокированному событию во время конфигурирования. Для назначения OB событию во время конфигурирования вы должны использовать ниспадающий список "HW interrupt [Аппаратное прерывание]:" (щелкните на направленной вниз стрелке справа) и выбрать OB из списка имеющихся OB аппаратных прерываний. Выберите подходящее имя OB из этого списка или выберите "<not connected [не присоединено]>" для отмены назначения.

Вы можете также назначать или отменять назначение разблокированного события, вызывающего аппаратные прерывания, во время исполнения. Для этого используйте в программе во время исполнения команды ATTACH или DETACH (при желании - несколько раз) для назначения или отмены назначения разблокированного события, вызывающего аппаратные прерывания, подходящему OB. Если никакой OB в настоящее время не назначен (из-за выбора "<not connected>" в конфигурации устройств или в результате выполнения команды DETACH), то разблокированное событие, вызывающее аппаратное прерывание, игнорируется.

Команда DETACH

Используйте команду DETACH для отмены назначения конкретного события или всех событий конкретному OB. Если параметр EVENT задан, то отменяется назначение только одного этого события из указанного OB_NR; все остальные события, назначенные в настоящее время этому OB_NR, не теряют своего назначения. Если параметр EVENT не задан, то назначение всех событий этому OB_NR будет отменено.

Коды условий

RET_VAL (W#16#....)	Состояние ENO	Описание
0000	1	Нет ошибки
0001	0	Назначения отсутствуют (только DETACH)
8090	0	OB не существует
8091	0	OB неверного типа
8093	0	Событие не существует

6.2.6.2 Команды запуска и отмены прерываний с задержкой

Вы можете запускать и отменять обработку прерываний с задержкой с помощью команд SRT_DINT и CAN_DINT. Каждое прерывание с задержкой является однократным событием, происходящим по истечении заданного интервала времени. Если событие, вызывающее задержку, отменяется до того, как время задержки истечет, то прерывание в программе не возникает.

Команда SRT_DINT запускает прерывание с задержкой, которое исполняет подпрограмму OB (организационный блок) по истечении времени задержки, указанного в параметре DTIME.

Команда CAN_DINT отменяет прерывание с задержкой, которое уже было запущено. В этом случае OB прерываний с задержкой не выполняется.

Параметры команды SRT_DINT

Параметр	Тип параметра	Тип данных	Описание
OB_NR	IN	Int	Организационный блок (OB), подлежащий запуску по истечении времени задержки: Выберите один из имеющихся OB прерываний с задержкой, которые были созданы с помощью опции дерева проектов "Add new block [Добавить новый блок]". Дважды щелкните на поле этого параметра, затем щелкните на вспомогательной пиктограмме, чтобы увидеть имеющиеся OB.
DTIME	IN	Time	Величина задержки (от 1 до 60000 мс) Вы можете сформировать более длительные времена задержки, например, используя счетчик внутри OB прерываний с задержкой.
SIGN	IN	Word	Не используется в S7-1200; принимается любое значение
RET_VAL	OUT	Int	Код условия выполнения

Параметры команды CAN_DINT

Параметр	Тип параметра	Тип данных	Описание
OB_NR	IN	Int	Идентификатор OB прерываний с задержкой. Вы можете использовать номер OB или символическое имя.
RET_VAL	OUT	Int	Код условия выполнения

Принцип действия

Команда SRT_DINT задает задержку времени, запускает внутренний таймер, отсчитывающий время задержки, и назначает событию, запускающему прерывание, подпрограмму OB прерываний с задержкой. По истечении заданного времени задержки генерируется программное прерывание, которое запускает на исполнение соответствующий OB прерываний с задержкой. Вы можете отменить запущенное прерывание с задержкой времени, прежде чем будет достигнуто заданное время задержки, с помощью команды CAN_DINT. Общее количество активных событий, вызывающих задержку времени и циклические прерывания, не должно превышать четырех.

Вставка подпрограмм OB прерываний с задержкой в ваш проект

Командам SRT_DINT и CAN_DINT могут быть поставлены в соответствие только OB прерываний с задержкой. В новом проекте нет OB прерываний с задержкой. Вы должны вставить OB прерываний с задержкой в свой проект. Для создания OB прерываний с задержкой, действуйте следующим образом:

1. Дважды щелкните на опции "Add new block [Добавить новый блок]" в ветви "Program blocks [Программные блоки]" дерева проектов, выберите "Organization block [Организационный блок] (OB)", а затем "Time delay interrupt [Прерывание с задержкой времени]".
2. Вы можете переименовать OB, выбрать язык программирования и номер блока. Если вы хотите назначить другой номер блока, чем тот, который был назначен автоматически, перейдите в режим ручной нумерации.
3. Отредактируйте подпрограмму OB прерываний с задержкой и сформируйте запрограммированную реакцию на событие, вызывающее прерывание с задержкой времени. Из OB прерываний с задержкой вы можете вызывать другие кодовые блоки FC и FB с глубиной вложения не боле четырех.
4. Имена вновь назначенных OB прерываний с задержкой будут находиться в вашем распоряжении при редактировании параметра OB_NR команд SRT_DINT и CAN_DINT.

Коды условий

RET_VAL (W#16#...)	**Описание**
0000	Нет ошибок
8090	Неправильный параметр OB_NR
8091	Неправильный параметр DTIME
80A0	Прерывание с задержкой еще не запущено

6.2.6.3

6.2.6.4 Команды активизации и деактивизации прерываний

Для разблокирования и блокирования обработки прерываний используются команды DIS_AIRT и EN_AIRT.

Команда DIS_AIRT задерживает обработку нового прерывающего события. Команду DIS_AIRT можно исполнять несколько раз в одном OB. Исполнения команды DIS_AIRT подсчитываются операционной системой. Каждая из этих команд действует, пока она не будет специально отменена командой EN_AIRT или пока текущий OB не будет полностью обработан.

Прерывания, которые произошли во время действия команды DIS_AIRT, обрабатываются, как только они будут снова разблокированы, или сразу после исполнения текущего OB.

Команда EN_AIRT разблокирует обработку прерывающего события, которое вы ранее заблокировали с помощью команды DIS_AIRT. Каждое исполнение DIS_AIRT должно быть отменено исполнением EN_AIRT. Если, например, вы заблокировали прерывания пять раз пятикратным исполнением команды DIS_AIRT, то вы должны их отменить пятикратным исполнением команды EN_AIRT.

Исполнения команды EN_AIRT должны происходить в том же самом OB или в любом FC или FB, вызываемом из этого OB, прежде чем для этого OB будут снова разблокированы прерывания.

Параметр RET_VAL указывает на то, сколько раз была заблокирована обработка прерываний, что определяется количеством поставленных в очередь исполнений команды DIS_AIRT. Обработка прерываний снова разблокируется только тогда, когда параметр RET_VAL = 0.

Параметр	Тип параметра	Тип данных	Описание
RET_VAL	OUT	Int	Количество задержек = количеству исполнений DIS_AIRT в очереди.

6.2.7 PID-регулирование

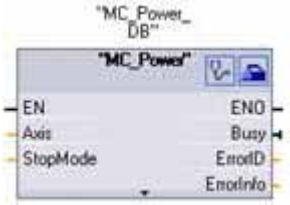
Оператор "PID_Compact" предоставляет в распоряжение PID-регулятор с оптимизацией самонастройки для автоматического и ручного режима.

Дальнейшую информацию о команде PID_Compact в системе онлайновой помощи портала TIA.

6.2.8 Команды управления перемещением

Команды управления перемещением используют для управления перемещением по оси соответствующий технологический блок данных и предназначенные для этого PTO (последовательности импульсов) CPU. Дальнейшую информацию о командах управления перемещением вы найдете в системе онлайновой помощи STEP 7 Basic.

ВНИМАНИЕ
Максимальная частота импульсных генераторов составляет 100 КГц для цифровых выходов CPU и 20 КГц для цифровых выходов сигнальной платы. Однако STEP 7 Basic не предупреждает вас, если вы сконфигурируете ось, максимальная скорость или частота по которой превышает аппаратные ограничения. Это может вызвать проблемы в вашем приложении, поэтому всегда обращайте внимание на то, чтобы не превысить максимальную частоту импульсов аппаратуры.



MC_Power разблокирует и блокирует ось для управления перемещением.

MC_Reset сбрасывает все ошибки управления перемещением. Все ошибки управления перемещением, которые могут быть квитированы, квитируются.

MC_Home устанавливает связь между программой управления осью и механической системой позиционирования оси.

MC_Halt отменяет все процессы перемещения и вызывает перемещение оси в стоп. Положение остановки не определено.

MC_MoveJog осуществляет толчковый режим работы для целей тестирования и ввода в действие.

MC_MoveAbsolute вызывает перемещение в абсолютное положение. Задание заканчивается, когда достигнута целевая позиция.

MC_MoveRelative вызывает перемещение для позиционирования относительно начального положения.

MC_MoveVelocity вызывает перемещение оси с заданной скоростью.

Указание

Последовательности импульсов не могут использоваться другими командами в программе пользователя

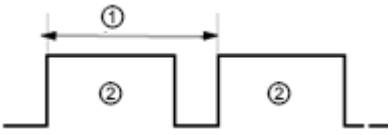
При конфигурировании выходов CPU или сигнальной платы в качестве генераторов импульсов (для PWM или основных команд управления перемещением) адреса соответствующих выходов (Q0.0, Q0.1, Q4.0 и Q4.1) удаляются из Q-памяти и не могут быть использованы для других целей в вашей пользовательской программе. Если ваша программа записывает значение в выход, используемый в качестве генератора импульсов, то CPU не записывает это значение в физический выход.

6.2.9 Команда формирования импульсов

6.2.9.1 Команда CTRL_PWM

Команда CTRL_PWM (Pulse Width Modulation [Широтно-импульсная модуляция] (PWM [ШИМ])) выдает последовательность импульсов с фиксированным временем цикла, но с переменным коэффициентом заполнения. Выход PWM работает непрерывно после запуска с заданной частотой (временем цикла).

Ширина импульсов меняется по потребности, чтобы достичь желаемого управления



① Время цикла
② Ширина импульса

Ширина импульса может быть задана в сотых долях времени цикла (0 – 100), в тысячных долях (0 – 1000), в десятитысячных долях (0 – 10000) или в аналоговом формате S7. Ширина импульса может меняться от 0 (отсутствие импульсов, всегда выключено) до полного заполнения (отсутствие импульсов, всегда включено).

Так как выход PWM может меняться от 0 до полного заполнения, то он представляет собой цифровой выход, во многом похожий на аналоговый выход. Например, выход PWM может использоваться для управления скоростью вращения двигателя от остановки до максимальной скорости или для управления положением клапана от закрытого до полностью открытого состояния.

Для управления быстрыми импульсными выходами имеются в распоряжении два импульсных генератора: ШИМ и последовательность импульсов (PTO). PTO используется командами управления перемещением. Вы можете назначить каждый импульсный генератор PWM или PTO, но не обоим в одно и то же время.

Эти два импульсных генератора поставлены в соответствие конкретным цифровым выходам, как это показано в следующей таблице. Вы можете использовать встроенные выходы CPU или, как вариант, выходы сигнальной платы. Адреса выходов показаны в следующей таблице (при этом предполагается конфигурация выходов по умолчанию). Если вы изменили адреса выходов, то эти адреса будут соответствовать адресам, назначенным вами. Независимо от этого PTO1/PWM1 использует первые два цифровых выхода, а PTO2/PWM2 – следующие два цифровых выхода, или на CPU, или на вставленной сигнальной плате. Обратите внимание, что для PWM нужен только один выход, тогда как PTO может, как вариант, использовать два выхода на канал. Если выход не нужен для импульсной функции, то он может быть использован для других целей.

Описание	**Назначение выходов по умолчанию**		
		Импульс	Направление
PTO 1	Встроенный в CPU	Q0.0	Q0.1
	Сигнальная плата	Q4.0	Q4.1
PWM 1	Встроенный в CPU	Q0.0	--
	Сигнальная плата	Q4.0	--
PTO 2	Встроенный в CPU	Q0.2	Q0.3
	Сигнальная плата	Q4.2	Q4.3
PWM 2	Встроенный в CPU	Q0.2	--
	Сигнальная плата	Q4.2	--

Конфигурирования импульсного канала для PWM

Чтобы подготовить функционирование PWM, сначала нужно сконфигурировать импульсный канал в конфигурации устройств, выбрав CPU, затем генератор импульсов (PTO/PWM), а затем PWM1 или PWM2. Разблокируйте генератор импульсов (триггерная кнопка). Если генератор импульсов разблокирован, то этому конкретному импульсному генератору назначается уникальное имя по умолчанию. Вы можете изменить это имя, редактируя поле "Name [Имя]:", но оно должно быть уникальным именем. Имена разблокированных генераторов импульсов становятся переменными в таблице переменных "constant", и будут предоставлены для использования в качестве параметра PWM команды CTRL_PWM.

ВНИМАНИЕ
Максимальная частота импульсных генераторов составляет 100 КГц для цифровых выходов CPU и 20 КГц для цифровых выходов сигнальной платы. Однако STEP 7 Basic не предупреждает вас, если вы сконфигурируете ось, максимальная скорость или частота по которой превышает аппаратные ограничения. Это может вызвать проблемы в вашем приложении, поэтому всегда обращайте внимание на то, чтобы не превысить максимальную частоту импульсов аппаратуры.

У вас есть возможность переименовать генератор импульсов, добавить комментарий и назначить параметры следующим образом:

- Используемый генератор импульсов: PWM или PTO (выберите PWM)
- Источник вывода: встроенный в CPU или сигнальная плата
- База времени: миллисекунды или микросекунды
- Формат ширины импульсов:
  - Сотые (от 0 до 100)
  - Тысячные (от 0 до 1000)
  - Десятитысячные (от 0 до 10000)
  - Аналоговый формат S7 (от 0 до 27648)
- Время цикла: Введите значение своего времени цикла. Это значение может быть изменено только в конфигурации устройств.
- Начальная ширина импульсов: Введите значение своей начальной ширины импульсов. Это значение может быть изменено во время исполнения.

Выходные адреса

Начальный адрес: Введите адрес выходного (Q) слова, где вы хотите сохранять значение ширины импульсов. Адресом по умолчанию является QW1000 для PWM1 и QW1002 для PWM2. Значение по этому адресу управляет шириной импульса и инициализируется на указанное выше значение для "Initial pulse width [Начальная ширина импульса]:" при каждом переходе CPU из STOP в RUN. Значение этого выходного (Q) слова можно изменять во время исполнения, чтобы изменить ширину импульса.

Параметр	Тип параметра	Тип данных	Начальное значение	Описание
PWM	IN	Word	0	Идентификатор PWM: Имена разблокированных генераторов импульсов становятся переменными в таблице переменных "constant" и предоставляются для использования в качестве параметра PWM.
ENABLE	IN	Bool		1=запустить генератор импульсов 0 = остановить генератор импульсов
BUSY	OUT	Bool	0	Функция занята
STATUS	OUT	Word	0	Код условия выполнения

Принцип действия

Для хранения информации о параметрах команда CTRL_PWM использует блок данных (DB). Когда вы вставляете команду CTRL_PWM в программный редактор, ей назначается DB. Параметры этого блока данных не изменяются отдельно пользователем, а управляются командой CTRL_PWM.

Задайте желаемый генератор импульсов, используя имя переменной для параметра PWM.

Когда вход EN принимает значение ИСТИНА, команда PWM_CTRL запускает или останавливает указанный PWM на основе значения на входе ENABLE. Ширина импульсов определяется значением в соответствующем адресе выходного (Q) слова.

Так как S7-1200 обрабатывает запрос, когда команда CTRL_PWM исполняется, то параметр BUSY у моделей CPU S7-1200 всегда принимает значение ЛОЖЬ.

Если обнаружена ошибка, то ENO устанавливается в ЛОЖЬ, а параметр STATUS содержит код ошибки.

Ширина импульса устанавливается на начальное значение, установленное в конфигурации устройств, когда ПЛК впервые переходит в режим RUN. Чтобы изменить ширину импульсов, вы записываете желаемые значения в адрес выходного (Q) слова, указанный в конфигурации устройств ("Output addresses [Выходные адреса]" / "Start address [Начальный адрес]:"). Чтобы записать желаемую ширину импульсов в соответствующее выходное (Q) слово, используйте команду, например, перемещения, преобразования, арифметических вычислений или PID. Вы должны использовать допустимый диапазон для значения Q-слова (сотые, тысячные, десятитысячные или аналоговый формат S7).

Коды условий

Значение STATUS	Описание
0	Нет ошибки
80A1	Идентификатор PWM не обращается к действительному PWM

Цифровым входам и выходам, назначенным PWM и PTO, нельзя принудительно присваивать значения

Цифровые входы и выходы, используемые для широтно-импульсной модуляции (PWM) и вывода последовательностей импульсов (PTO), назначаются при конфигурировании устройств. Когда адреса цифровых входов и выходов назначены этим функциям, то значения по этим адресам не могут быть изменены функцией принудительного присваивания значений в таблице наблюдения.

Выходы, предназначенные для вывода последовательностей импульсов, не могут использоваться другими командами в программе пользователя

Когда вы конфигурируете выходы CPU или сигнальной платы в качестве генераторов импульсов (для использования с PWM или основными командами управления перемещениями), соответствующие адреса выходов (Q0.0, Q0.1, Q4.0 и Q4.1) удаляются из памяти выходов (Q) и не могут быть использованы для других целей в вашей пользовательской программе. Если ваша пользовательская программа запишет какое-либо значение в выход, используемый в качестве генератора импульсов, то CPU не запишет это значение в физический выход.

6.3 Глобальные библиотечные команды

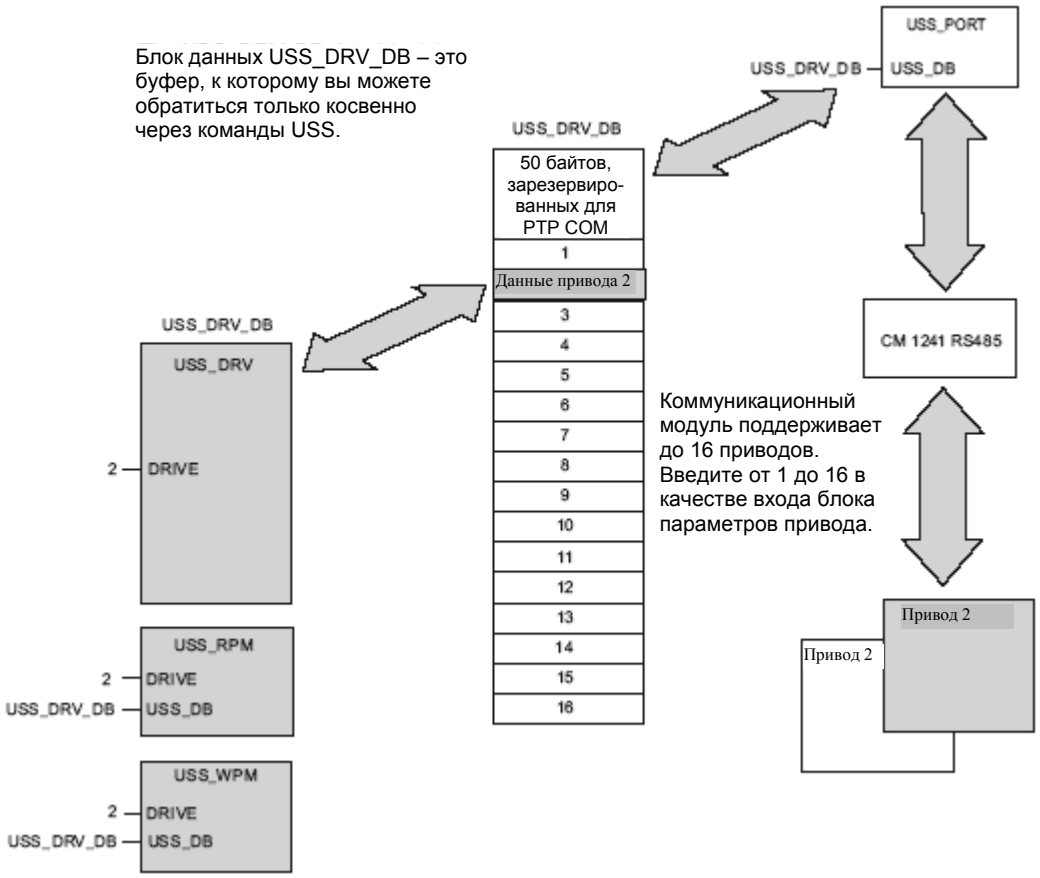
6.3.1 USS

Библиотека протокола USS служит для управления приводами фирмы Siemens, поддерживающими протокол USS. Эти команды включают в себя функции, специально спроектированные для использования протокола USS в обмене данными с приводом. Модуль CM 1241 RS485 обменивается данными с приводами через порты RS485. С помощью библиотеки USS вы можете управлять физическим приводом, а также считывать и записывать параметры привода.

6.3.1.1 Предпосылки для использования протокола USS

Эта библиотека предоставляет 1 FB и 3 FC для поддержки протокола USS. Каждый коммуникационный модуль CM 1241 RS485 поддерживает до 16 приводов.

Один экземплярный блок данных поддерживает функции хранения и буферизации для всех приводов в сети USS, подключенных к установленному вами коммуникационному модулю PtP. Функции USS для этих приводов совместно используют информацию, содержащуюся в этом блоке данных.



Все приводы (до 16), подключенные к одному CM 1241 RS485, являются частью одной и той же сети USS. Все приводы, подключенные к другому CM 1241 RS485, являются частью другой сети USS. Так как S7-1200 поддерживает до трех устройств CM 1241 RS485, то вы можете иметь до трех сетей USS, по 16 приводов, максимум, в каждой сети, так что в целом поддерживается до 48 приводов USS.

Каждая сеть USS управляется с помощью уникального блока данных (для трех сетей USS, использующих три устройства CM 1241 RS485, необходимы три блока данных). Все команды, связанные с одной сетью USS, должны совместно использовать этот блок данных. Сюда входят все команды USS_DRV, USS_PORT, USS_RPM и USS_WPM, используемые для управления всеми приводами в одной сети USS.

Команда USS_DRV является функциональным блоком (FB). Когда вы помещаете команду USS_DRV в редактор, диалоговое окно "Call options [Параметры вызова]" предложит вам назначить DB для этого FB. Если это первая команда USS_DRV в данной программе для этой сети USS, то вы можете принять назначение DB по умолчанию (или изменить имя, если желаете), и новый DB будет создан для вас. Если, однако, это не первая команда USS_DRV для этого канала, то вы должны использовать ниспадающий список в диалоговом окне "Call options", чтобы выбрать соответствующий DB, ранее назначенный этой сети USS.

Все команды USS_PORT, USS_RPM и USS_WPM являются функциями (FC). Когда вы помещаете эти функции в редактор, никакого DB не назначается. Вместо этого вы должны назначить входу "USS_DB" этих команд соответствующий DB (дважды щелкните на поле этого параметра, затем щелкните на вспомогательной пиктограмме, чтобы увидеть имеющиеся DB).

Функция USS_PORT управляет существующей связью между CPU и приводами через коммуникационный модуль PtP. При каждом вызове этой функции обрабатывается одна связь с одним приводом. Ваша программа должна вызывать эту функцию достаточно быстро, чтобы предотвратить истечение времени ожидания у приводов. Вы можете вызвать эту функцию в главной программе или в любом OB прерываний.

Функциональный блок USS_DRV предоставляет вашей программе доступ к указанному приводу в сети USS. Его входы и выходы соответствуют состояниям и управляющим элементам для привода. Если в сети имеется 16 приводов, то ваша программа должна иметь не менее 16 вызовов USS_DRV, по одному для каждого привода. Эти блоки должны вызываться с частотой, необходимой для управления функциями привода.

Функциональный блок USS_DRV можно вызывать только из OB, содержащего главную программу.

⚠ ОСТОРОЖНО

Вызывайте USS_DRV, USS_RPM, USS_WPM только из OB, содержащего главную программу. Функция USS_PORT может быть вызвана из любого OB, обычно из OB прерываний с задержкой времени.

Если не воспрепятствовать прерыванию USS_PORT, то это может привести к неожиданным ошибкам.

Функции USS_RPM и USS_WPM считывают и записывают рабочие параметры удаленных приводов. Эти параметры управляют внутренним функционированием привода. Для определения этих параметров обратитесь к руководству для соответствующего привода. Ваша программа может содержать этих функций столько, сколько необходимо, но в каждый данный момент времени может быть активен только один запрос на чтение или запись на каждый привод. Функции USS_RPM и USS_WPM вы можете вызывать только из OB, содержащего главную программу.

Расчет времени, необходимого для обмена данными с приводом

Обмен данными с приводом происходит асинхронно по отношению к циклу S7-1200. Обычно S7-1200 совершает несколько циклов, прежде чем будет завершена коммуникационная транзакция с приводом.

Интервал USS_PORT – это время, необходимое для одной транзакции с приводом. В следующей таблице показан минимальный интервал USS_PORT для каждой скорости передачи. Вызов функции USS_PORT чаще интервала USS_PORT не увеличивает количества транзакций. Интервал времени ожидания привода – это количество времени, которое предоставляется для транзакции, если из-за коммуникационных ошибок для завершения транзакции требуется 3 попытки. По умолчанию библиотека протокола USS автоматически выполняет до 2 попыток при каждой транзакции.

Скорость передачи	**Расчетный минимальный интервал вызова USS_PORT (миллисекунды)**	**Интервал для сообщения привода о превышении времени ожидания на каждый привод (миллисекунды)**
1200	790	2370
2400	405	1215
4800	212.5	638
9600	116.3	349
19200	68.2	205
38400	44.1	133
57600	36.1	109
115200	28.1	85

6.3.1.2 Команда USS_DRV

Команда USS_DRV обменивается данными с приводом, создавая сообщения с запросами и интерпретируя ответные сообщения привода. Для каждого привода должен использоваться отдельный функциональный блок, но все функции USS, относящиеся к одной сети USS и одному коммуникационному модулю PtP, должны использовать один и тот же экземплярный блок данных. Вы должны ввести имя DB, когда вы вставляете первую команду USS_DRV, а потом вновь использовать этот DB, который был создан при вставке первой команды.

При первом исполнении USS_DRV привод, указанный адресом USS (параметр DRIVE), инициализируется в экземплярном DB. После этой инициализации следующие исполнения USS_PORT могут начинать обмен данными с приводом по этому номеру.

Изменение номера привода требует перевода ПЛК в состояние STOP, а затем снова в RUN, чтобы инициализировать экземплярный DB. Входные параметры конфигурируются в передающем буфере USS TX, а выходы, если они имеются, считываются из "предыдущего" действительного ответного буфера. При исполнении USS_DRV передача данных не производится. Обмен данными с приводами осуществляется, при исполнении команды USS_PORT. Команда USS_DRV только конфигурирует сообщения, подлежащие передаче, и интерпретирует данные, которые могли быть приняты в предыдущем запросе.

Вы можете управлять направление вращения привода с помощью входа DIR (BOOL) или знака (положительного или отрицательного) на входе SPEED_SP (REAL). Следующая таблица показывает, как работают эти входы совместно для определения направления вращения в предположении, что двигатель включен для вращения вперед.

SPEED_SP	**DIR**	**Направление вращения привода**
Значение > 0	0	Назад
Значение > 0	1	Вперед
Значение < 0	0	Вперед
Значение < 0	1	Назад

LAD (стандартное представление)

LAD (расширенное представление)

Расширьте блок, чтобы отобразить все параметры, щелкнув в нижней части блока.

Контакты параметров, изображенные серым цветом, являются необязательными и не нуждаются в назначениях.

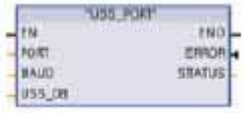
Параметр	Тип параметра	Тип данных	Описание
RUN	IN	Bool	Стартовый бит привода: Когда принимает значение ИСТИНА, этот вход разблокирует привод для работы с предустановленной скоростью.
OFF2	IN	Bool	Бит электрического останова: Когда принимает значение ЛОЖЬ, этот бит заставляет привод вращаться по инерции до остановки без торможения.
OFF3	IN	Bool	Бит быстрого останова – Когда принимает значение ЛОЖЬ, этот бит вызывает быстрый останов привода путем применения торможения.
F_ACK	IN	Bool	Бит квитирования неисправности – Этот бит сбрасывает бит неисправности привода. Этот бит устанавливается после устранения неисправности, показывая приводу, что ему больше не нужно сообщать о предыдущей неисправности.
DIR	IN	Bool	Управление направление вращения привода – Этот бит устанавливается, если привод должен вращаться в направлении вперед (для положительного SPEED_SP).
DRIVE	IN	USInt	Адрес привода: Этот вход является адресом привода USS. Допустимое значение находится в диапазоне от 1 до 16 .
PZD_LEN	IN	USInt	Длина в словах – Это количество слов данных PZD. Допустимыми значениями являются 2, 4, 6 или 8 слов. Значение по умолчанию 2.
SPEED_SP	IN	Real	Заданное значение скорости – Это скорость привода в процентах от сконфигурированной частоты. Положительное значение указывает направление вперед (если DIR имеет значение ИСТИНА).
CTRL3	IN	UInt	Управляющее слово 3 – Значение, записанное в конфигурируемый пользователем параметр на приводе. Пользователь должен сконфигурировать его на приводе. Необязательный параметр.
CTRL4	IN	UInt	Управляющее слово 4 – Значение, записанное в конфигурируемый пользователем параметр на приводе. Пользователь должен сконфигурировать его на приводе. Необязательный параметр.
CTRL5	IN	UInt	Управляющее слово 5 – Значение, записанное в конфигурируемый пользователем параметр на приводе. Пользователь должен сконфигурировать его на приводе. Необязательный параметр.
CTRL6	IN	UInt	Управляющее слово 6 – Значение, записанное в конфигурируемый пользователем параметр на приводе. Пользователь должен сконфигурировать его на приводе.
CTRL7	IN	UInt	Управляющее слово 7 – Значение, записанное в конфигурируемый пользователем параметр на приводе. Пользователь должен сконфигурировать его на приводе. Необязательный параметр.

Параметр	Тип параметра	Тип данных	Описание
CTRL8	IN	UInt	Управляющее слово 8 – Значение, записанное в конфигурируемый пользователем параметр на приводе. Пользователь должен сконфигурировать его на приводе. Необязательный параметр.
NDR	OUT	Bool	Готовы новые данные – Когда принимает значение ИСТИНА, этот бит указывает, что выходы содержат данные из нового коммуникационного запроса.
ERROR	OUT	Bool	Произошла ошибка – Когда принимает значение ИСТИНА, это указывает, что произошла ошибка и выход STATUS действителен. Все остальные выходы в случае ошибки устанавливаются в ноль. Коммуникационные ошибки сообщаются только на выходах команд USS_PORT ERROR и STATUS.
STATUS	OUT	UInt	Состояние запроса. Указывает результат цикла сканирования. Это не слово состояния, возвращаемое приводом.
RUN_EN	OUT	Bool	Рабочий режим разблокирован – Этот бит сообщает, работает ли привод.
D_DIR	OUT	Bool	Направление вращения привода – Этот бит сообщает, вращается ли привод вперед.
INHIBIT	OUT	Bool	Привод заблокирован – Этот бит сообщает о состоянии бита блокировки на приводе.
FAULT	OUT	Bool	Неисправность привода – Этот бит указывает, что привод зарегистрировал неисправность. Пользователь должен устранить проблему, а затем установить бит F_ACK, а затем сбросить данный бит.
SPEED	OUT	REAL	Текущая скорость привода (масштабированное значение слова состояния привода 2) – Значение скорости привода в процентах от сконфигурированной скорости.
STATUS1	OUT	UInt	Слово состояния привода 1 – Это значение содержит фиксированные биты состояния привода.
STATUS3	OUT	UInt	Слово состояния привода 3 – Это значение содержит конфигурируемое пользователем слово состояния привода.
STATUS4	OUT	UInt	Слово состояния привода 4 – Это значение содержит конфигурируемое пользователем слово состояния привода.
STATUS5	OUT	UInt	Слово состояния привода 5 – Это значение содержит конфигурируемое пользователем слово состояния привода.
STATUS6	OUT	UInt	Слово состояния привода 6 – Это значение содержит конфигурируемое пользователем слово состояния привода.
STATUS7	OUT	UInt	Слово состояния привода 7 – Это значение содержит конфигурируемое пользователем слово состояния привода.
STATUS8	OUT	UInt	Слово состояния привода 8 – Это значение содержит конфигурируемое пользователем слово состояния привода.

6.3.1.3 Команда USS_PORT

Команда USS_PORT управляет обменом данными через сеть USS. Обычно в программе имеется только по одной функции USS_PORT на коммуникационный модуль PtP, и каждый вызов этой функции обрабатывает передачу к приводу или от него. Ваша программа должна исполнять функцию USS_PORT достаточно часто, чтобы предотвратить простои привода. Все функции USS, относящиеся к одной сети USS и коммуникационному модулю PtP, должны использовать один и тот же экземплярный блок данных. USS_PORT обычно вызывается из OB прерываний с задержкой, чтобы предотвратить простои привода и сохранить для вызовов USS_DRV самые последние обновления данных USS.

LAD

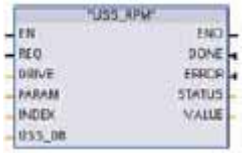


FBD

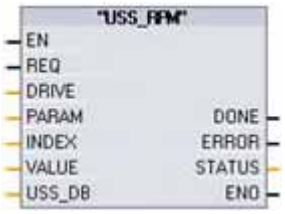
Параметр	Тип параметра	Тип данных	Описание
PORT	IN	Port	Коммуникационный модуль PtP. Идентификатор: Это константа, на которую можно ссылаться во вкладке "Constants [Константы]" стандартной таблицы переменных.
BAUD	IN	DInt	Скорость передачи, подлежащая использованию при обмене данными через USS.
USS_DB	IN	DInt	Это ссылка на экземплярный DB, который был создан и инициализирован при вставке команды USS_DRV в вашу программу.
ERROR	OUT	Bool	Когда принимает значение ИСТИНА, этот контакт указывает, что произошла ошибка и выход STATUS действителен.
STATUS	OUT	UInt	Состояние запроса. Указывает результат цикла сканирования или инициализации. Дополнительная информация для некоторых кодов состояния находится в переменной "USS_Extended_Error".

6.3.1.4 Команда USS_RPM

LAD



FBD



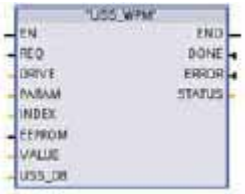
Команда USS_RPM считывает параметр из привода. Все функции USS, относящиеся к одной сети USS и коммуникационному модулю PtP, должны использовать один и тот же блок данных. Команда USS_RPM должна вызываться из OB, содержащего главную программу.

Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Передать запрос: Когда принимает значение ИСТИНА, он указывает, что нужен новый запрос на чтение. Он игнорируется, если запрос для этого параметра уже стоит в очереди.
DRIVE	IN	USInt	Адрес привода: Этот вход является адресом привода USS. Допустимое значение находится в диапазоне от 1 до 16 .
PARAM	IN	UInt	Номер параметра: Это вход указывает, какой параметр привода записывается. Диапазон значений этого параметра составляет от 0 до 2047. За подробностями о том, как получить доступ к параметрам за пределами этого диапазона, обратитесь к руководству для этого привода.
INDEX	IN	UInt	Индекс параметра: Этот вход указывает, в какой индекс параметра привода должна производиться запись. 16-битовое значение, в котором младший байт является текущим значением индекса с диапазоном (от 0 до 255). Старший байт может также использоваться приводом и зависит от конкретного привода. Подробности см. в руководстве для своего привода.
USS_DB	IN	Variant	Это ссылка на экземплярный DB, который был создан и инициализирован при вставке команды USS_DRV в вашу программу.
VALUE	IN	Word, Int, UInt, DWord, DInt, UDInt, Real	Это значение параметра, которое было считано и действительно только тогда, когда бит DONE имеет значение ИСТИНА.

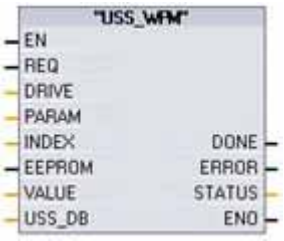
Параметр	Тип параметра	Тип данных	Описание
DONE	OUT	Bool	Готово: Значение ИСТИНА указывает, что выход VALUE содержит ранее запрошенное значение параметра чтения. Этот бит устанавливается, когда USS_DRV обнаруживает ответные данные о чтении из привода. Этот бит сбрасывается, когда: • вы запрашиваете ответные данные через опрос другого USS_RPM или • после второго из следующих двух вызовов USS_DRV
ERROR	OUT	Bool	Произошла ошибка – Когда принимает значение ИСТИНА, это указывает, что произошла ошибка и выход STATUS действителен. Все остальные выходы в случае ошибки устанавливаются в ноль. Коммуникационные ошибки сообщаются только на выходах команд USS_PORT ERROR и STATUS.
STATUS	OUT	UInt	Это значение состояния запроса. Он указывает результат запроса на чтение. Дополнительная информация для некоторых кодов состояния находится в переменной "USS_Extended_Error".

6.3.1.5 Команда USS_WPM

LAD



FBD



Команда USS_WPM изменяет параметр в приводе. Все функции USS, относящиеся к одной сети USS и коммуникационному модулю PtP, должны использовать один и тот же блок данных. USS_WPM должна вызываться из OB, содержащего главную программу.

Указание

Операции записи в ЭСППЗУ

Остерегайтесь чрезмерного использования операций записи в ЭСППЗУ. Минимизируйте количество операций записи в ЭСППЗУ, чтобы продлить срок его службы.

Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Передать запрос: Когда принимает значение ИСТИНА, он указывает, что необходим новый запрос на запись. Он игнорируется, если запрос для этого параметра уже стоит в очереди.
DRIVE	IN	USInt	Адрес привода: Этот вход является адресом привода USS. Допустимое значение находится в диапазоне от 1 до 16 .
PARAM	IN	UInt	Номер параметра: Это вход указывает, какой параметр привода записывается. Диапазон значений этого параметра составляет от 0 до 2047. За подробностями о том, как получить доступ к параметрам за пределами этого диапазона, обратитесь к руководству для этого привода.
INDEX	IN	UInt	Индекс параметра: Этот вход указывает, в какой индекс параметра привода должна производиться запись. 16-битовое значение, в котором младший байт является текущим значением индекса с диапазоном (от 0 до 255). Старший байт может также использоваться приводом и зависит от конкретного привода. Подробности см. в руководстве для своего привода.
EEPROM	IN	Bool	Сохранить в ЭСППЗУ привода: Когда принимает значение ИСТИНА, то записи в параметр привода будут сохранены в ЭСППЗУ привода. Если ЛОЖЬ, то запись является временной и не будет сохранена, если после выключения и последующего включения привода.

Параметр	Тип параметра	Тип данных	Описание
VALUE	IN	Word, Int, UInt, DWord, DInt, UDInt, Real	Значение параметра, которое должно быть записано. Оно должно оставаться действительным при изменении состояния REQ.
USS_DB	IN	Variant	Это ссылка на экземплярный DB, который был создан и инициализирован при вставке команды USS_DRV в вашу программу.
DONE	OUT	Bool	Готово: Значение ИСТИНА указывает, что вход VALUE был записан в привод. Этот бит устанавливается, когда USS_DRV обнаруживает данные реакции о записи из привода. Этот бит сбрасывается, когда вы запрашиваете подтверждение привода о том, что запись завершена, через опрос другого USS_WPM или после второго из следующих двух вызовов USS_DRV.
ERROR	OUT	Bool	Произошла ошибка: Когда принимает значение ИСТИНА, это указывает, что произошла ошибка и выход STATUS действителен. Все остальные выходы в случае ошибки устанавливаются в ноль. Коммуникационные ошибки сообщаются только на выходах команд USS_PORT ERROR и STATUS.
STATUS	OUT	UInt	Это значение состояния запроса. Он указывает результат запроса на запись. Дополнительная информация для некоторых кодов состояния находится в переменной "USS_Extended_Error".

6.3.1.6 Коды состояния USS

Коды состояния команд USS возвращаются на выходе STATUS функций USS.

Значение STATUS (W#16#....)	**Описание**
0000	Нет ошибки
8180	Длина ответа привода не соответствует символам, полученным от привода. Номер привода, в котором произошла ошибка, возвращается в переменной "USS_Extended_Error". Описание расширенного набора ошибок см. под этой таблицей.
8181	Параметр VALUE не принадлежал к типам данных Word, Real или DWord
8182	Пользователь ввел для параметра тип Word, а получил в ответе от привода DWord или Real
8183	Пользователь ввел для параметра тип DWord или Real, а получил в ответе от привода Word
8184	Ответная посылка от привода имела неправильную контрольную сумму. Номер привода, в котором произошла ошибка, возвращается в переменной "USS_Extended_Error". Описание расширенного набора ошибок см. под этой таблицей.
8185	Недопустимый адрес для привода (допустимый диапазон адресов для привода: 1-16)
8186	Заданное значение скорости вне допустимого диапазона (допустимый диапазон заданных значений для скорости: от -200% до 200%)
8187	Неверный номер привода получен в ответ на посланный запрос. Номер привода, в котором произошла ошибка, возвращается в переменной "USS_Extended_Error". Описание расширенного набора ошибок см. под этой таблицей.
8188	Указано недопустимое число слов для PZD (допустимый диапазон = 2, 4, 6 или 8 слов)
8189	Была задана недопустимая скорость передачи
818A	Канал запроса параметров используется другим запросом для этого привода
818B	Привод не отвечал на запросы и их повторения. Номер привода, в котором произошла ошибка, возвращается в переменной "USS_Extended_Error". Описание расширенного набора ошибок см. под этой таблицей.
818C	Привод вернул ошибку из расширенного набора в ответ на запрос параметров. Описание расширенного набора ошибок см. под этой таблицей.
818D	Привод вернул ошибку недопустимого доступа в ответ на запрос параметров. См. руководство к своему приводу, чтобы получить информацию о том, почему может быть ограничен доступ к параметру
818E	Привод не был инициализирован: Этот код ошибки возвращается в USS_RPM или USS_WPM, если команда USS_DRV для этого привода не была вызвана хотя бы один раз. Это удерживает инициализацию первого цикла USS_DRV от перезаписи стоящего в очереди запроса на чтение или запись параметров, так как при этом привод инициализируется как новый элемент. Для устранения этой ошибки вызовите USS_DRV для этого номера привода.
80Ax-80Fx	Конкретные ошибки, возвращаемые из FB двухточечной связи (PtP, Point-to-Point), вызванного библиотекой USS: Значения кодов этих ошибок не изменяются библиотекой USS, а определяются в описаниях команд PtP.

Коды расширенного набора ошибок для приводов USS

Приводы USS поддерживают доступ на чтение и запись к внутренним параметрам привода. Это свойство делает возможным дистанционное управление и конфигурирование привода. Операции доступа к параметрам привода могут потерпеть неудачу из-за таких ошибок, как выход значений за пределы допустимого диапазона или недопустимых запросов для текущего режима работы привода. Привод генерирует код ошибки, который возвращается в переменной "USS_Extended_Error" экземплярного DB команды USS_DRV. Значение этого кода ошибки действительно только для последнего исполнения команды USS_RPM или USS_WPM. Код ошибки привода помещается в переменную "USS_Extended_Error", если значением STATUS является шестнадцатеричное 818C. Значение кода ошибки "USS_Extended_Error" зависит от модели привода. Описание кодов расширенного набора ошибок для операций чтения и записи параметров вы найдете в руководстве для соответствующего привода.

6.3.2 MODBUS

6.3.2.1 MB_COMM_LOAD

LAD

FBD

Команда MB_COMM_LOAD конфигурирует порт на модуле двухточечной связи (Point-to-Point, PtP) CM 1241 RS485 или CM 1241 RS232 для обмена данными через протокол Modbus RTU.

Параметр	Тип параметра	Тип данных	Описание
PORT	IN	UInt	Идентификатор коммуникационного порта: После вставки модуля CM в конфигурацию устройств идентификатор порта появляется во вспомогательном ниспадающем списке на выводе PORT блока команды. На эту константу можно также ссылаться во вкладке "Constants [Константы]" стандартной таблицы переменных.
BAUD	IN	UDInt	Выбор скорости передачи: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 Все остальные значения недопустимы
PARITY	IN	UInt	Выбор контроля четности: • 0 – отсутствует • 1 – нечетные • 2 – четные

Параметр	Тип параметра	Тип данных	Описание
FLOW_CTRL	IN	UInt	Выбор управления потоком: • 0 – (по умолчанию) нет управления потоком • 1 – аппаратное управление потоком с всегда установленным RTS (недействительно для портов RS485) • 2 - аппаратное управление потоком с переключаемым RTS
RTS_ON_DLY	IN	UInt	Выбор задержки включения RTS: • 0 – (по умолчанию) нет задержки от активизации RTS до передачи первого символа сообщения • от 1 до 65535 – задержка в миллисекундах от активизации RTS до передачи первого символа сообщения (недействительно для портов RS485). Задержки RTS должны применяться независимо от выбора FLOW_CTRL.
RTS_OFF_DLY	IN	UInt	Выбор задержки выключения RTS: • 0 – (по умолчанию) нет задержки от последнего переданного символа до деактивизации RTS • от 1 до 65535 – задержка в миллисекундах от последнего переданного символа до деактивизации RTS (недействительно для портов RS485). Задержки RTS должны применяться независимо от выбора FLOW_CTRL.
RESP_TO	IN	UInt	Время ожидания ответа: Время в миллисекундах, в течение которого команда MB_MASTER ожидает ответа от slave-устройства. Если slave-устройство не отвечает в течение этого интервала времени, то MB_MASTER повторит запрос или завершит запрос с ошибкой, если сделано заданное количество повторных попыток. От 5 мс до 65535 мс (значение по умолчанию = 1000 мс).
MB_DB	IN	Variant	Ссылка на экземплярный блок данных, используемый командами MB_MASTER и MB_SLAVE. После вставки команды MB_SLAVE или MB_MASTER в вашу программу идентификатор DB появляется во вспомогательном ниспадающем списке на входе MB_DB блока команды.
ERROR	OUT	Bool	Ошибка: • 0 – Ошибка не обнаружена • 1 – Указывает, что ошибка была обнаружена и код ошибки в параметре STATUS действителен
STATUS	OUT	Word	Код ошибки конфигурирования порта

Команда MB_COMM_LOAD исполняется для конфигурирования порта для протокола Modbus RTU. После конфигурирования порта вы можете вести обмен данными через Modbus, исполняя команду MB_SLAVE или MB_MASTER.

Команда MB_COMM_LOAD должна вызываться один раз для инициализации порта. Команду MB_COMM_LOAD необходимо вызвать снова только в том случае, если должен быть изменен один из коммуникационных параметров. Вы можете вызвать MB_COMM_LOAD из OB запуска и выполнить его один раз, или вы можете инициировать вызов для однократного исполнения с помощью системного флага первого цикла.

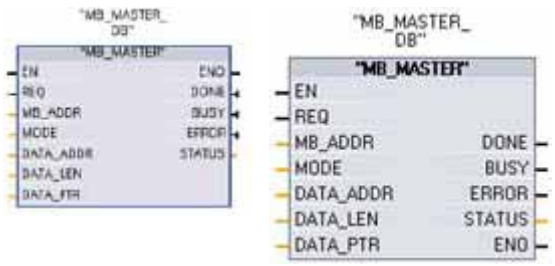
Для конфигурирования каждого порта каждого коммуникационного модуля, который используется для обмена данными через Modbus, должен быть использован один экземпляр команды MB_COMM_LOAD. Вы должны назначить уникальный экземплярный блок данных MB_COMM_LOAD для каждого порта, который вы используете. CPU S7-1200 ограничен 3 коммуникационными модулями.

Экземплярный блок данных назначается, когда вы вставляете команду MB_MASTER или MB_SLAVE. Ссылка на этот экземплярный блок данных производится, когда вы задаете параметр MB_DB в команде MB_COMM_LOAD.

Значение STATUS (W#16#....)	**Описание**
0000	Нет ошибки
8180	Недопустимое значение ID порта
8181	Недопустимое значение скорости передачи
8182	Недопустимое значение контроля четности
8183	Недопустимое значение управления потоком
8184	Недопустимое значение времени ожидания ответа
8185	Неправильный указатель MB_DB на экземплярный DB для команды MB_MASTER или MB_SLAVE

6.3.2.2 MB_MASTER

LAD **FBD**



Команда MB_MASTER позволяет вашей программе осуществлять обмен данными в качестве master-устройства Modbus, используя порт на модуле двухточечной связи (Point-to-Point, PtP) CM 1241 RS485 или CM 1241 RS232. Вы можете получить доступ к данным в одном или нескольких slave-устройствах Modbus.

Экземплярный блок данных назначается, когда вы вставляете команду MB_MASTER в свою программу. Имя этого экземплярного блока данных MB_MASTER используется, когда вы задаете параметр MB_DB в команде MB_COMM_LOAD.

Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Ввод запроса: • 0 – нет запроса • 1 – запрос на передачу данных в slave-устройство(-а) Modbus
MB_ADR	IN	USInt	Адрес станции Modbus RTU: Допустимый диапазон адресов: от 0 до 247 Значение 0 зарезервировано для транслирования сообщения всем slave-устройствам Modbus. Функциональные коды Modbus 05, 06, 15 и 16 являются единственными функциональными кодами, поддерживаемыми для широкого транслирования.
MODE	IN	USInt	Выбор режима: Определяет вид запроса: чтение, запись или диагностика Подробности вы найдете в следующей таблице функций Modbus.
DATA_ADDR	IN	UDInt	Начальный адрес в Slave-устройстве: Определяет начальный адрес данных, к которым нужно получить доступ в slave-устройстве Modbus. Допустимые адреса вы найдете в следующей таблице функций Modbus.
DATA_LEN	IN	UInt	Длина данных: Определяет число битов или слов, к которым нужно получить доступ в этом запросе. Допустимые длины вы найдете в следующей таблице функций Modbus.
DATA_PTR	IN	Variant	Указатель на данные: Указывает на адрес DB в CPU для записываемых или считываемых данных. DB должен быть типа "NOT symbolic access only [Только НЕ символическая адресация]". См. ниже указание для DATA_PTR.
NDR	OUT	Bool	Готовы новые данные: • 0 – транзакция не завершена • 1 – указывает, что команда MB_MASTER завершила запрошенную транзакцию со slave-устройством(-ами) Modbus
BUSY	OUT	Bool	Занят: • 0 – транзакция командой MB_MASTER не производится • 1 – осуществляется транзакция командой MB_MASTER
ERROR	OUT	Bool	Ошибка: • 0 – ошибка не обнаружена • 1 – указывает, что обнаружена ошибка и код ошибки в параметре STATUS действителен
STATUS	OUT	Word	Код условия выполнения

Правила обмена данными для master-устройства Modbus

- Команда MB_COMM_LOAD должна быть исполнена для конфигурирования порта до того, как команда MB_MASTER сможет обмениваться данными с этим портом.
- Если порт должен использоваться для инициирования запросов master-устройства Modbus, то этот порт не может использоваться командой MB_SLAVE. С этим портом может использоваться один или несколько экземпляров исполнения команды MB_MASTER.
- Команды Modbus не используют события, прерывающие обмен данными, для управления процессом обмена данными. Ваша программа должна опрашивать команду MB_MASTER об условиях завершения передачи и приема.
- Если ваша программа работает как master-устройство Modbus и использует команду MB_MASTER для передачи запроса slave-устройству, то вы должны продолжать опрос (исполнять команду MB_MASTER) до тех пор, пока не будет получен ответ от slave-устройства.
- Осуществляйте все исполнения команды MB_MASTER для данного порта из одного и того же OB (или из OB одного и того же уровня приоритета).

Параметр REQ

Значение REQ ЛОЖЬ = Нет запроса

Значение REQ ИСТИНА = Запрос на передачу данных slave-устройству(-ам) Modbus.

Вы должны подать сигнал на этот вход через контакт, управляемый нарастающим фронтом, при первом вызове на исполнение команды MB_MASTER. Импульс, запускаемый фронтом, вызывает запрос на передачу один раз. Все входы фиксируются и удерживаются в неизменном состоянии на время одного запроса и ответа, инициируемого этим входом.

Внутри команда MB_MASTER запускает механизм состояний, который обеспечивает, что никакой другой команде MB_MASTER не будет позволено выдавать запрос, пока данный запрос не будет завершен.

Кроме того, если один и тот же экземпляр вызова FB команды MB_MASTER исполняется снова с входом REQ, имеющим значение ИСТИНА, до завершения запроса, то последующие передачи не будут выполняться. Однако, как только запрос будет завершен, то будет выдан новый запрос, если MB_MASTER исполняется с входом REQ, имеющим значение ИСТИНА.

Через параметры DATA_ADDR и MODE выбирается тип функции Modbus

DATA_ADDR (начальный адрес Modbus в Slave-устройстве): Указывает начальный адрес данных, к которым нужно получить доступ в slave-устройстве Modbus.

MB_MASTER использует вместо ввода кода функции вход MODE. Комбинация MODE и области адресов Modbus определяет код функции, который используется в фактическом сообщении Modbus. В следующей таблице показано соответствие между параметром MODE команды MBUS_MASTER, кодом функции Modbus и областью адресов Modbus.

Функции Modbus команды MB_MASTER				
	Параметр DATA_ADDR для адреса Modbus	Тип адреса	Параметр DATA_LEN для длины данных Modbus	Функция Modbus
Режим 0				
Чтение	от 00001 до 09999	Выходные биты	от 1 до 2000	01H
	от 10001 – 19999	Входные биты	от 1 до 2000	02H
	от 30001 - 39999	Входные регистры	от 1 до 125	04H
	от 40001 до 49999 от 400001 до 465536 (расширенные)	Регистры временного хранения информации	от 1 до 125	03H
Режим 1				
Запись	от 00001 до 09999	Выходные биты	1 (один бит)	05H
	от 40001 до 49999 от 400001 до 465536 (расширенные)	Регистры временного хранения информации	1 (одно слово)	06H
	от 00001 до 09999	Выходные биты	от 2 до 1968	15H
	от 40001 до 49999 от 400001 до 465536 (расширенные)	Регистры временного хранения информации	от 2 до 123	16H
Режим 2				
Некоторые slave-устройства Modbus не поддерживают записи в один бит или одно слово с помощью функций Modbus 05H и 06H. В этих случаях используется режим 2 для принудительной записи одного бита и слова с помощью функций Modbus 15H и 16H.				
Запись	от 00001 до 09999	Выходные биты	от 1 до 1968	15H
	от 40001 до 49999 от 400001 до 465536(расширенные)	Регистры временного хранения информации	от 1 до 123	16H
Режим 11 • Считывает слово счетчика событий из slave-устройства Modbus, которое указывается в качестве входа для MB_ADDR • У slave-устройства Modbus S7-1200 фирмы Siemens этот счетчик увеличивает свое значение на 1 каждый раз, когда slave-устройство получает действительный запрос на чтение или запись (не широковещательный) от master-устройства Modbus. • Возвращаемое значение сохраняется по адресу слова, указанному в качестве входа DATA_PTR. • Для этого режима не требуется действительного значения DATA_LEN.				

Функции Modbus команды MB_MASTER
Режим 80 • Проверяет коммуникационное состояние slave-устройства Modbus, которое указывается в качестве входа для MB_ADDR • Установка выходного битв NDR команды MB_MASTER указывает, что адресованное slave-устройство Modbus отреагировало соответствующими ответными данными. • В вашу программу никакие данные не возвращаются. • Для этого режима не требуется действительного значения DATA_LEN.
Режим 81 • Сбрасывает счетчик событий (как возвращенный режимом 11) на slave-устройстве Modbus, который указывается в качестве входа для MB_ADDR • Установка выходного битв NDR команды MB_MASTER указывает, что адресованное slave-устройство Modbus отреагировало соответствующими ответными данными. • В вашу программу никакие данные не возвращаются. • Для этого режима не требуется действительного значения DATA_LEN.

Параметр DATA_PTR

Параметр DATA_PTR указывает локальный исходный или целевой адрес (адрес в CPU S7-1200) данных, которые, соответственно, должны быть прочитаны или записаны. Когда вы используете команду MB_MASTER для создания master-устройства Modbus, вы должны создать глобальный блок данных, который предоставляет память для процессов чтения и записи на slave-устройствах Modbus.

Указание

Параметр DATA_PTR должен ссылаться на глобальный блок данных, который был создан с деактивированным атрибутом Symbolic Access Only [Только символический доступ].

Вы должны снять метку с триггерной кнопки "Symbolic address only", когда вы добавляете новый блок данных, чтобы создать глобальный DB классического типа.

Структуры блока данных для параметра DATA_PTR

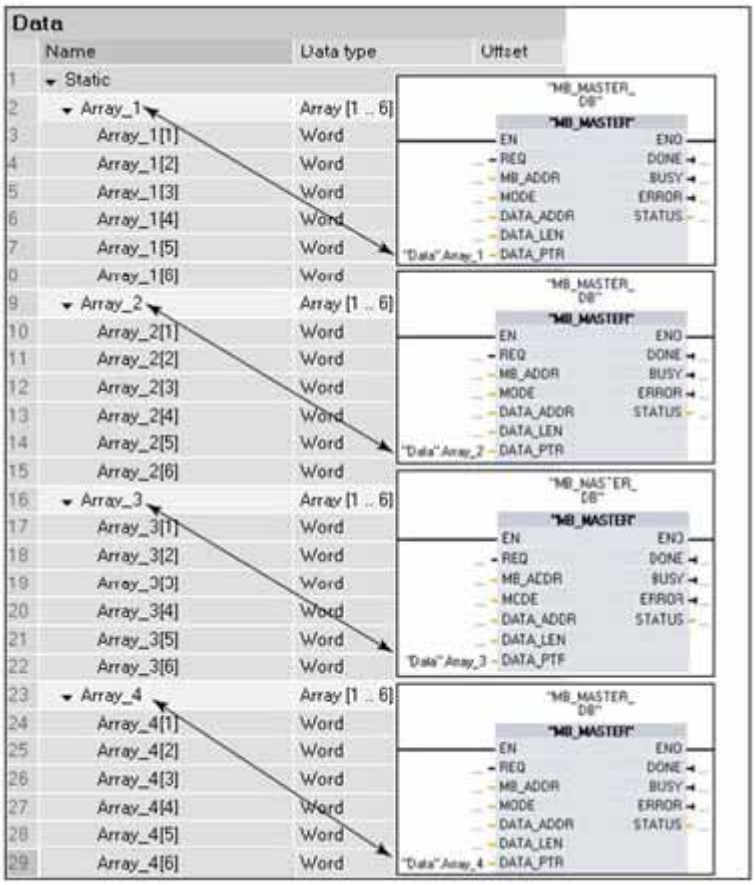
- Эти типы данных действительны для **чтения слов** адресов Modbus от 30001 до 39999, от 40001 до 49999 и от 400001 до 465536, а также для **записи слов** в адреса Modbus от 40001 до 49999 и от 400001 до 465536.
  - Стандартный массив данных типа WORD, UINT или INT, как показано ниже.
  - Именованная структура типа WORD, UINT или INT, в которой каждый элемент имеет уникальное имя и 16-битовый тип данных.
  - Именованная смешанная структура, в которой каждый элемент имеет уникальное имя и 16- или 32-битовый тип данных.
- Для чтения и записи битов для адресов Modbus от 00001 до 09999 и от 10001 до 19999.
  - Стандартный массив из данных типа Bool.
  - Именованная булева структура из уникально именованных переменных типа Bool.
- Хотя это и не является необходимым, но рекомендуется, чтобы каждая команда MB_MASTER имела свою собственную отдельную область в глобальном блоке данных. Причиной для этой рекомендации является то, что возможность нарушения данных возрастает, если несколько команд MB_MASTER производят чтение и запись в одной и той же области глобального блока данных.
- Не требуется, чтобы области данных DATA_PTR находились в одном и том же глобальном блоке данных. Вы можете создать один блок данных с несколькими областями для операций чтения Modbus, один блок данных для операций записи Modbus или один блок данных для каждой подчиненной станции.
- Все массивы в нижеприведенном примере созданы как массивы с базисом 1 [1 … ##]. Массивы могут создаваться также как массивы с базисом 0 [0 … ###] или смесью базиса 0 и базиса 1.

Примеры команд MB_MASTER, обращающихся к глобальным блокам данных DATA_PTR

Представленный ниже пример глобального блока данных использует 4 уникально именованных массивов с 6 словами для сохранения данных запросов Modbus. Хотя массивы данных в этом примере имеют одинаковый размер, но массивы могут быть разных размеров и показаны одного размера для упрощения примеров. Каждый массив может быть также заменен структурой данных, содержащей более информативные имена переменных и смешанные типы данных. Примеры альтернативных структур данных представлены в описании параметра HR_DB команды (стр. 237).

В представленных ниже примерах команды MB_MASTER показан только параметр DATA_PTR, но не показаны другие необходимые параметры. Цель этих примеров состоит в том, чтобы показать, как команда MB_MASTER использует блок данных DATA_PTR.

Стрелки на рисунке показывают, как каждый массив связан с различными командами MB_MASTER.



Первый элемент любого массива или структуры всегда является первым источником или первой целью любого процесса чтения или записи Modbus. Все описанные ниже сценарии основаны на вышеприведенной диаграмме.

Сценарий 1: Если первая команда MB_MASTER считывает 3 слова данных из адреса Modbus 40001 на любом действующем slave-устройстве Modbus, то происходит следующее.

Слово из адреса 40001 сохраняется в "Data".Array_1[1].

Слово из адреса 40002 сохраняется в "Data".Array_1[2].

Слово из адреса 40003 сохраняется в "Data".Array_1[3].

Сценарий 2: Если первая команда MB_MASTER считывает 4 слова данных из адреса Modbus 40015 на любом действующем slave-устройстве Modbus, то происходит следующее.

Слово из адреса 40015 сохраняется в "Data".Array_1[1].

Слово из адреса 40016 сохраняется в "Data".Array_1[2].

Слово из адреса 40017 сохраняется в "Data".Array_1[3].

Слово из адреса 40018 сохраняется в "Data".Array_1[4].

Сценарий 3: Если вторая команда MB_MASTER считывает 2 слова данных из адреса Modbus 30033 на любом действующем slave-устройстве Modbus, то происходит следующее.

Слово из адреса 30033 сохраняется в "Data".Array_2[1].

Слово из адреса 30034 сохраняется в "Data".Array_2[2].

Сценарий 4: Если третья команда MB_MASTER записывает 4 слова данных в адрес Modbus 40050 на любом действующем slave-устройстве Modbus, то происходит следующее.

Слово из "Data".Array_3[1] записывается в адрес Modbus 40050.

Слово из "Data".Array_3[2] записывается в адрес Modbus 40051.

Слово из "Data".Array_3[3] записывается в адрес Modbus 40052.

Слово из "Data".Array_3[4] записывается в адрес Modbus 40053.

Сценарий 5: Если третья команда MB_MASTER записывает 3 слова данных в адрес Modbus 40001 на любом действующем slave-устройстве Modbus, то происходит следующее.

Слово из "Data".Array_3[1] записывается в адрес Modbus 40001.

Слово из "Data".Array_3[2] записывается в адрес Modbus 40002.

Слово из "Data".Array_3[3] записывается в адрес Modbus 40003.

Сценарий 6: Если четвертая команда MB_MASTER использует режим 11 (извлекает значение счетчика действительных сообщений) из любого действующего slave-устройства Modbus, происходит следующее.

Слово со значением счетчика сохраняется в "Data".Array_4[1].

Пример считывания и записи битов, используя адреса слов в качестве входа DATA_PTR

Таблица 6-1. Сценарий 7: Чтение 4 выходных битов, начиная с адреса Modbus 00001

Значения входов команды MB_MASTER			**Значения slave-устройства Modbus**	
MB_ADDR	27 (пример slave-устройства)		**00001**	**ON**
MODE	0 (чтение)		**00002**	**ON**
DATA_ADDR	00001 (выходы)		**00003**	**OFF**
DATA_LEN	4		**00004**	**ON**
DATA_PTR	"Data".Array_4		00005	ON
			00006	OFF
			00007	ON
			00008	OFF

Значения "Data".Array_4[1] после запроса Modbus	
Старший байт	Младший байт
xxxx-1011	xxxx-xxxx
x указывает, что данные не изменяется	

Таблица 6-2. Сценарий 8: Чтение 12 выходных битов, начиная с адреса Modbus 00003

Значения входов команды MB_MASTER			**Значения slave-устройства Modbus**				
MB_ADDR	27 (пример slave-устройства)		00001	ON		**00010**	**ON**
MODE	0 (чтение)		00002	ON		**00011**	**OFF**
DATA_ADDR	00003 (выходы)		**00003**	**OFF**		**00012**	**OFF**
DATA_LEN	12		**00004**	**ON**		**00013**	**ON**
DATA_PTR	"Data".Array_4		**00005**	**ON**		**00014**	**OFF**
			00006	**OFF**		00015	ON
			00007	**ON**		00016	ON
			00008	**ON**		00017	OFF
			00009	**OFF**		00018	ON

Значения "Data".Array_4[1] после запроса Modbus	
Старший байт	Младший байт
1011-0110	xxxx-0100-
x указывает, что данные не изменяются	

Таблица 6-3. Сценарий 9: Запись 5 выходных битов, начиная с адреса Modbus 00001

Значения входов команды MB_MASTER			**Выходы slave-устройства до**			**Выходы slave-устройства после**
MB_ADDR	27 (пример slave-устройства)		**00001**	**ON**		**OFF**
MODE	1 (запись)		**00002**	**ON**		ON
DATA_ADDR	00001 (выходы)		**00003**	**OFF**		ON
DATA_LEN	5		**00004**	**ON**		**OFF**
DATA_PTR	"Data".Array_4		**00005**	**ON**		ON
			00006	OFF		не меняется
			00007	ON		не меняется
			00008	ON		не меняется
			00009	OFF		не меняется

Значения "Data".Array_4[1] после запроса Modbus на запись	
Старший байт	Младший байт
xxx1-0110	xxxxx-xxxx
x указывает, что данные не используются в запросе Modbus	

Таблица 6-4. Сценарий 10: Чтение 22 выходных битов, начиная с адреса Modbus 00003

Значения входов команды MB_MASTER			**Значения slave-устройства Modbus**				
MB_ADDR	27 (пример slave-устройства)		00001	ON		**00014**	**ON**
MODE	0 (чтение)		00002	ON		**00015**	**OFF**
DATA_ADDR	00003 (выходы)		**00003**	**OFF**		**00016**	**ON**
DATA_LEN	22		**00004**	**ON**		**00017**	**ON**
DATA_PTR	"Data".Array_4		**00005**	**ON**		**00018**	**OFF**
			00006	**OFF**		**00019**	**ON**
			00007	**ON**		**00020**	**ON**
			00008	**ON**		**00021**	**OFF**
			00009	**ON**		**00022**	**ON**
			00010	**OFF**		**00023**	**ON**
			00011	**OFF**		**00024**	**OFF**
			00012	**ON**		00025	OFF
			00013	**OFF**		00026	ON

Значения "Data".Array_4[1] после запроса Modbus	
Старший байт	Младший байт
0111-0110	0110-1010

Значения "Data".Array_4[2] после запроса Modbus	
Старший байт	Младший байт
xx01-1011	xxxx-xxxx
x указывает, что данные не изменяются	

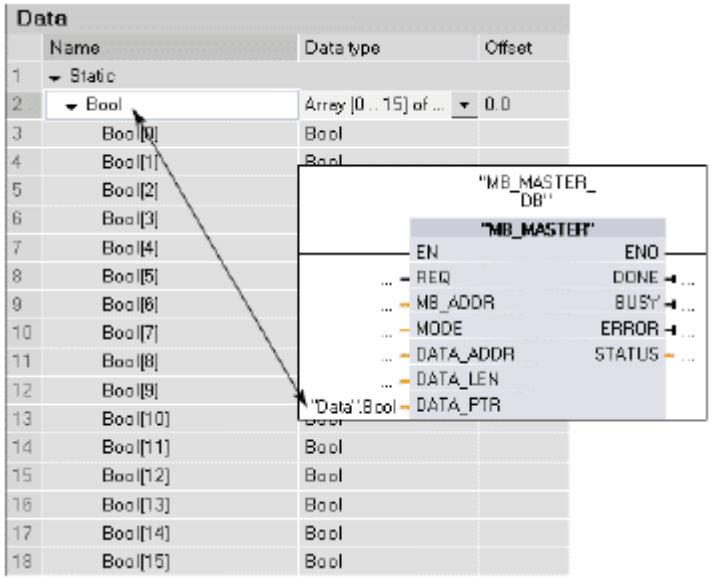
Пример считывания и записи битов, используя адреса битов в качестве входа DATA_PTR

Хотя процессы считывания и записи Modbus в адреса битов могут осуществляться путем использования адресов слов, области DATA_PTR могут быть также сконфигурированы как булевы типы данных, структуры или массивы, чтобы установить прямую, 1 к 1, взаимосвязь для первого считываемого или записываемого бита с помощью команды MB_MASTER.

Если вы используете булевы массивы или структуры, то рекомендуется, чтобы вы делали размер данных кратным 8 битам (по границам байта). Например, если вы создаете булев массив из 10 битов, то программное обеспечение STEP 7 Basic выделит 16 битов (2 байта) в глобальном блоке данных для этих 10 битов. Внутри блока данных они будут храниться как байт1 [xxxx xxxx] байт2 [---- --xx], где x указывает доступные адреса данных, а – указывает недоступные адреса. Длина запросов Modbus может составлять до 16 битов, но старшие 6 битов будут помещены в те адреса байта 2, которые не будут доступны для вашей программы.

Булевы области могут быть созданы как массив булевых значений или как структура из булевых переменных. Оба метода работают одинаковым образом и отличаются только способом создания и способом обращения к ним в вашей программе.

Представленный внизу вид редактора глобального блока данных показывает один массив из 16 булевых значений, созданный с базисом 0. Этот массив может быть также создан как массив с базисом 1. Стрелка показывает, как это массив связан с командой MB_MASTER.



Сценарии 11 и 12 показывают соответствие адресов Modbus адресам булева массива.

Таблица 6- 5 Сценарий 11: Запись 5 выходных битов, начиная с адреса Modbus 00001

Значения входов команды MB_MASTER			**Выходы slave-устройства до**		**Данные DATA_PTR**	**Выходы slave-устройства после**
MB_ADDR	27 (пример slave-устройства)		00001	ON	"Data".Bool[0]=ЛОЖЬ	OFF
MODE	1 (запись)		00002	ON	"Data".Bool[1]=ИСТИНА	ON
DATA_ADDR	00001 (выходы)		00003	OFF	"Data".Bool[2]=ИСТИНА	ON
DATA_LEN	5		00004	ON	"Data".Bool[3]-ЛОЖЬ	OFF
DATA_PTR	"Data".Bool		00005	ON	"Data".Bool[4]=ЛОЖЬ	OFF
			00006	OFF		не меняется
			00007	ON		не меняется
			00008	OFF		не меняется

Таблица 6- 6 Сценарий 12: Чтение 15 выходных битов, начиная с адреса Modbus 00004

Значения входов команды MB_MASTER			**Значение slave-устройства Modbus**		**Данные DATA_PTR после**
MB_ADDR	27 (пример slave-устройства)		00001	ON	
MODE	0 (чтение)		00002	ON	
DATA_ADDR	00003 (выходы)		**00003**	**OFF**	"Data".Bool[0]=ЛОЖЬ
DATA_LEN	15		**00004**	**ON**	"Data".Bool[1]=ИСТИНА
DATA_PTR	"Data".Bool		**00005**	**ON**	"Data".Bool[2]=ИСТИНА
			00006	**OFF**	"Data".Bool[3]-ЛОЖЬ
			00007	**ON**	Data".Bool[4]=ИСТИНА
			00008	**ON**	Data".Bool[5]=ИСТИНА
			00009	**ON**	Data".Bool[6]=ИСТИНА
			00010	**OFF**	Data".Bool[7]=ЛОЖЬ
			00011	**OFF**	Data".Bool[8]=ЛОЖЬ
			00012	**ON**	Data".Bool[9]=ИСТИНА
			00013	**OFF**	Data".Bool[10]=ЛОЖЬ
			00014	**ON**	Data".Bool[11]=ИСТИНА
			00015	**OFF**	Data".Bool[12]=ЛОЖЬ
			00016	**ON**	Data".Bool[13]=ИСТИНА
			00017	**ON**	Data".Bool[14]=ИСТИНА
			00018	OFF	
			00019	ON	

Коды условий

Значение STATUS (W#16#....)	**Описание**
0000	Нет ошибки
80C8	Заданное время ожидания ответа (см. RCVTIME или MSGTIME) равно 0.
80D1	Приемник издал запрос на управление потоком, чтобы остановить активную передачу и не возобновлять ее в течение указанного времени ожидания. Эта ошибка генерируется также при аппаратном управлении потоком, если приемник не объявляет о готовности к приему (CTS) в течение указанного времени ожидания.
80D2	Запрос на передачу был отменен, так как не был получен сигнал о готовности (DSR) от аппаратуры передачи данных (DCE).
80E0	Сообщение было завершено, так как приемный буфер полон.
80E1	Сообщение было завершено в результате ошибки контроля четности.
80E2	Сообщение было завершено в результате ошибки кадрирования.
80E3	Сообщение было завершено в результате ошибки переполнения.
80E4	Сообщение было завершено в результате того, что указанная длина превышает общий размер буфера.
8180	Неверное значение идентификатора порта
8186	Неверный адрес станции Modbus
8188	Неверное значение MODE или режим записи для области адресов slave-устройства, доступных только для чтения
8189	Неверное значение адреса данных
818A	Неверное значение длины данных
818B	• Неверный указатель на источник или цель локальных данных: Неверный размер
818C	Указатель на DB безопасного типа для DATA_PTR (должен быть DB классического типа)
8200	Порт занят обработкой запроса на передачу

6.3.2.3 MB_SLAVE

Команда MB_SLAVE позволяет вашей программе осуществлять обмен данными в качестве slave-устройства Modbus, используя порт на модуле двухточечной связи (Point-to-Point, PtP) CM 1241 RS485 или CM 1241 RS232. Master-устройство Modbus RTU может послать запрос, а затем ваша программа отвечает, исполняя команду MB_SLAVE.

Помещая команду MB_SLAVE в свою программу, вы должны назначить ей уникальный экземплярный блок данных. Имя этого экземплярного блока данных используется, когда вы задаете параметр MB_DB в команде MB_COMM_LOAD.

Коды функций связи Modbus (1, 2, 4, 5 и 15) могут считывать и записывать байты и слова непосредственно в образах процесса ПЛК на входах и выходах. В следующей таблице показано соответствие адресов Modbus образу процесса в CPU.

Функции Modbus команды MB_SLAVE						**S7-1200**	
Коды	Функция	Область данных	Диапазон адресов			Область данных	Адрес в CPU
01	Чтение битов	Выход	1	до	8192	Образ процесса на выходах	от Q0.0 до Q1023.7
02	Чтение битов	Вход	10001	до	18192	Образ процесса на входах	от I0.0 до I1023.7
04	Чтение слов	Вход	30001	до	30512	Образ процесса на входах	от IW0 до IW1022
05	Запись бита	Выход	1	до	8192	Образ процесса на выходах	от Q0.0 до Q1023.7
15	Запись битов	Выход	1	до	8192	Образ процесса на выходах	от Q0.0 до Q1023.7

Коды функций связи Modbus (3, 6, 16) используют отдельный и уникальный блок данных Modbus с регистром временного хранения информации. Этот DB вы должны создать, прежде чем вы сможете задать параметр MB_HOLD_REG команды MB_SLAVE. В следующей таблице показано соответствие регистра временного хранения информации Modbus адресу DB параметра MB_HOLD_REG в ПЛК.

Функции Modbus команды MB_SLAVE				**S7-1200**	
Коды	Функция	Область данных	Диапазон адресов	Область данных DB в CPU	Адрес DB в CPU
03	Чтение слов	Регистр временного хранения информации	от 40001 до 49999	MB_HOLD_REG	Слова от 1 до 9999
			от 400001 до 465535		Слова от 1 до 65535
06	Запись слова	Регистр временного хранения информации.	от 40001 до 49999	MB_HOLD_REG	Слова от 1 до 9999
			от 400001 до 465535		Слова от 1 до 65535
16	Запись слов	Регистр временного хранения информации	от 40001 до 49999	MB_HOLD_REG	Слова от 1 до 9999
			от 400001 до 465535		Слова от 1 до 65535

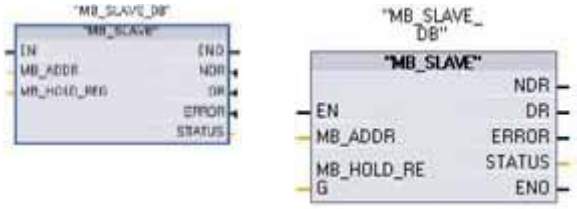
В следующей таблице показаны диагностические функции, поддерживаемые Modbus.

Диагностические функции Modbus команды MB_SLAVE S7-1200		
Коды	Подфункция	Описание
08	0000H	Эхотест – возврат данных опроса: Команда MB_SLAVE возвращает обратно в master-устройство Modbus полученное слово данных.
08	000AH	Очистка счетчика коммуникационных событий: Команда MB_SLAVE очищает счетчик коммуникационных событий, используемый для функции 11 Modbus.
11		Получение значения счетчика коммуникационных событий: Команда MB_SLAVE использует внутренний счетчик коммуникационных событий для записи количества успешных запросов Modbus на чтение и запись, которые были посланы slave-устройством Modbus. Счетчик не реагирует на функции 8, 11 и широковещательные запросы. Он не реагирует также на запросы, приводящие к коммуникационным ошибкам (например, ошибки контроля четности или контроля с помощью циклического избыточного кода CRC).

Команда MB_SLAVE поддерживает трансляцию запросов на запись из master-устройств Modbus, пока запросы относятся к действительным адресам.

Независимо от того, действителен запрос или нет, команда MB_SLAVE не отвечает master-устройству Modbus на широковещательный запрос.

LAD **FBD**



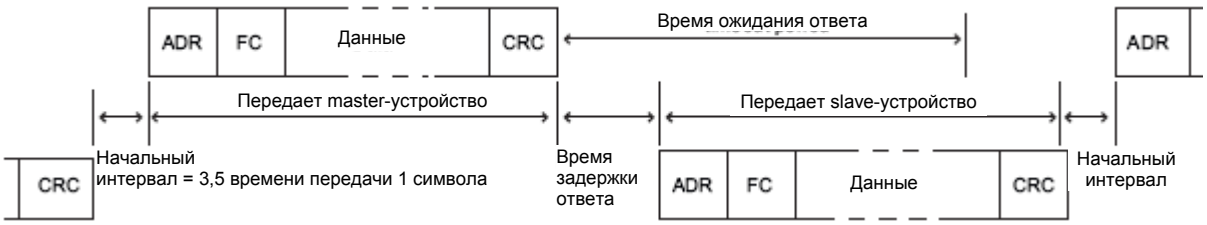
Параметр	**Тип параметра**	**Тип данных**	**Описание**
MB_ADDR	IN	USINT	Адрес Modbus RTU (от 1 до 247): Адрес станции slave-устройства Modbus.
MB_HOLD_REG	IN	VARIANT	Указатель на DB регистра временного хранения информации Modbus. DB регистра временного хранения информации должен быть классическим глобальным DB. См. ниже указание к MB_HOLD_REG.
NDR	OUT	BOOL	Готовы новые данные: • 0 – нет новых данных • 1 – указывает, что master-устройством Modbus были записаны новые данные
DR	OUT	BOOL	Чтение данных: • 0 – данные не считывались • 1 – указывает, что данные считывались master-устройством Modbus
ERROR	OUT	BOOL	Ошибка: • 0 – ошибка не обнаружена • 1 – указывает, что обнаружена ошибка и код ошибки в параметре STATUS действителен.
STATUS	OUT	WORD	Код ошибки

Правила обмена данными для slave-устройства Modbus

- Команда MB_COMM_LOAD должна быть исполнена для конфигурирования порта, прежде чем команда MB_SLAVE сможет обмениваться данными с этим портом.
- Если порт должен отвечать как slave-устройство master-устройству Modbus, то этот порт не может быть использован командой MB_MASTER. Для каждого данного порта может использоваться только один экземпляр MB_SLAVE.
- Команды Modbus не используют события, прерывающие обмен данными, для управления коммуникационным процессом. Ваша программа должна управлять коммуникационным процессом путем опроса команды MB_SLAVE о завершенных процессах передачи и приема.
- Команда MB_SLAVE должна исполняться периодически с частотой, позволяющей ей своевременно отвечать на поступающие запросы от master-устройства Modbus.
- Вы должны вызывать MB_SLAVE в каждом цикле из OB программного цикла.

Принцип действия

Команда MB_SLAVE должна исполняться периодически, чтобы получать каждый запрос от master-устройства Modbus, а затем соответствующим образом отвечать. Частота исполнения команды MB_SLAVE зависит от интервала времени ожидания ответа, задаваемого master-устройством Modbus. Это иллюстрируется следующим рисунком.



Время ожидания ответа - это интервал времени, в течение которого master-устройство Modbus ожидает начала ответа от slave-устройства Modbus. Этот интервал времени не определяется протоколом Modbus, а является параметром соответствующего master-устройства Modbus. Частота исполнения (время от одного исполнения до другого) команды MB_SLAVE должна основываться на конкретных параметрах вашего master-устройства Modbus. Как минимум, вы должны исполнять MB_SLAVE дважды в течение времени ожидания master-устройства Modbus.

Примеры для параметра MB_HOLD_REG

Параметр MB_HOLD_REG – это указатель на блок данных регистра временного хранения информации Modbus. Этот DB используется для хранения значений данных, к которым разрешен доступ (на чтение или запись) master-устройству Modbus. Вы должны создать этот блок данных и назначить структуру типов данных, которые оттуда будут считываться или туда записываться, прежде чем он сможет быть использован командой MB_SLAVE.

Указание

Блок данных регистра временного хранения информации Modbus должен ссылаться на глобальный блок данных, который должен быть создан с деактивированной триггерной кнопкой атрибута Symbolic Access Only [Только символический доступ].

При добавлении нового блока данных вы должны снять пометку с триггерной кнопки "Symbolic address only", чтобы создать блок данных классического типа

Регистры временного хранения информации могут использовать следующие структуры данных этого DB:

- Стандартный массив из слов
- Именованная структура из слов
- Именованная составная структура

Следующие примеры программ показывают, как можно использовать параметр MB_HOLD_REG параметр для обработки структур данных этого DB.

Пример 1 - Стандартный массив из слов

Этот пример для регистра временного хранения информации представляет собой массив из слов. Назначения типов данных могут быть изменены на другие типы размеров слова (INT и UINT).

Преимущества:

- Этот тип структуры регистра временного хранения информации может быть быстро и легко создан.
- Упрощена логика программы для доступа к элементу данных.

Недостатки:

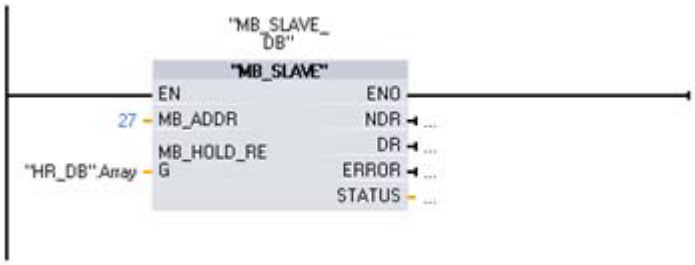
- Хотя вы можете программно ссылаться на каждый элемент массива по символическим именам (от "HR_DB"."Array"[1] до "HR_DB"."Array"[10]), эти имена не описывают внутреннюю функцию данных.
- Массив может состоять только из одного типа данных. В пользовательской программе со строгим контролем типов может потребоваться преобразование типов.

Вот как выглядит структура массива из слов в редакторе блоков данных.

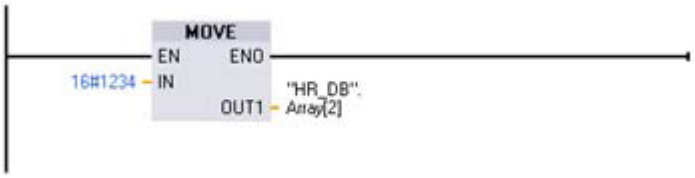
HR_DB

	Name	Data type	Offset	Default value	Initial value	Retain	Comment
1	Static						
2	Array	Array [1..10] of W...	0.0			☐	40001 to 40010
3	Array[1]	Word		W#16#0000	W#16#0000		
4	Array[2]	Word		W#16#0000	W#16#0000		
5	Array[3]	Word		W#16#0000	W#16#0000		
6	Array[4]	Word		W#16#0000	W#16#0000		
7	Array[5]	Word		W#16#0000	W#16#0000		
8	Array[6]	Word		W#16#0000	W#16#0000		
9	Array[7]	Word		W#16#0000	W#16#0000		
10	Array[8]	Word		W#16#0000	W#16#0000		
11	Array[9]	Word		W#16#0000	W#16#0000		
12	Array[10]	Word		W#16#0000	W#16#0000		

На следующем рисунке показано, как этот массив назначается входу MB_HOLD_REG команды MB_SLAVE.



К каждому элементу этого массива можно получить доступ по символическому имени, как это показано ниже. В этом примере новое значение передается во второй элемент массива, который соответствует адресу Modbus 40002.



Каждое из слов в этом массиве, в соответствии с определением в блоке данных, снабжает команду MB_SLAVE адресами регистра временного хранения информации Modbus. В этом примере, так как в массиве имеется только 10 элементов, имеется в наличии только 10 адресов регистра временного хранения информации Modbus, которые могут быть использованы этой командой MB_SLAVE и доступны master-устройству Modbus.

Ниже показано соответствие имен элементов массива адресам Modbus.

"HR_DB".Array[1]	Адрес Modbus 40001
" HR_DB ". Array[2]	Адрес Modbus 40002
" HR_DB ". Array[3]	Адрес Modbus 40003
...	...
" HR_DB ". Array[9]	Адрес Modbus 40009
" HR_DB ".Array [10]	Адрес Modbus 40010

Пример 2 - Именованная структура из слов

Этот пример регистра временного хранения информации представляет собой ряд слов с описательными символическими именами.

Преимущества:

- Каждый элемент структуры имеет описательное имя с назначенным ему конкретным типом данных.

Недостатки:

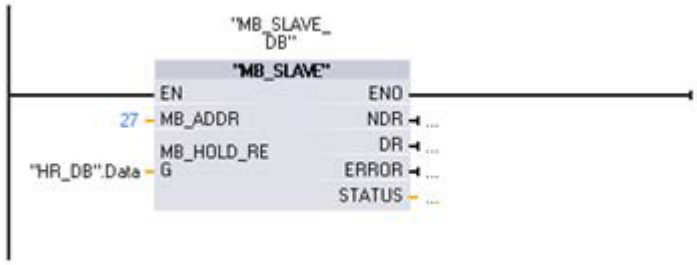
- Создание этого типа структуры требует больше времени, чем для стандартного массива из слов.
- Элементы требуют дополнительных символических ссылок при использовании в программе пользователя. В то время как ссылка на первый элемент простого массива имеет вид "HR_DB".Array[0], ссылка на элемент этого типа имеет вид "HR_DB".Data.Temp_1.

Вот как именованная структура из слов появляется в редакторе блоков данных. Каждый элемент имеет уникальное имя и может иметь тип WORD, UINT или INT.

HR_DB

	Name	Data type	Offset	Initial value	Retain	Comment
1	Static					
2	Data	Struct	0.0			
3	Temp_1	Int	0.0	0		40001
4	Temp_2	Int	2.0	0		40002
5	Temp_3	Int	4.0	0		40003
6	Good_Count	UInt	6.0	0		40004
7	Bad_Count	UInt	8.0	0		40005
8	Rework_Count	UInt	10.0	0		40006
9	Line_Stops	UInt	12.0	0		40007
10	Avg_Time	UInt	14.0	0		40008
11	Code_1	Word	16.0	0		40009
12	Code_2	Word	18.0	0		40010

На следующем рисунке показано, как вышеприведенная структура данных может быть назначена входу MB_HOLD_REG команды MB_SLAVE в вашей программе.



К каждому элементу этого массива можно получить доступ по символическому имени, как это показано ниже. В этом примере новое значение передается во второй элемент массива, который соответствует адресу Modbus 40002.

Ниже показано соответствие имен элементов массива адресам Modbus.

"HR_DB".Data.Temp_1	Адрес Modbus 40001
"HR_DB".Data.Temp_2	Адрес Modbus 40002
"HR_DB".Data.Temp_3	Адрес Modbus 40003
"HR_DB".Data.Good_Count	Адрес Modbus 40004
"HR_DB".Data.Bad_Count	Адрес Modbus 40005
"HR_DB".Data.Rework_Count	Адрес Modbus 40006
"HR_DB".Data.Line_Stops	Адрес Modbus 40007
"HR_DB".Data.Avg_Time	Адрес Modbus 40008
"HR_DB".Data.Code_1	Адрес Modbus 40009
"HR_DB".Data.Code_2	Адрес Modbus 40010

Пример 3 - Именованная составная структура

Этот пример регистра временного хранения информации представляет собой ряд смешанных типов данных с описательными символическими именами.

Преимущества:

- Каждый элемент структуры имеет описательное имя с назначенным ему конкретным типом данных.
- Возможна непосредственная передача типов данных, не основанных на словах.

Недостатки:

- Создание этого типа структуры требует больше времени, чем для стандартного массива из слов.
- Master-устройство Modbus должно быть сконфигурировано для приема данных, которые оно будет принимать от slave-устройства Modbus. Как показано на следующем рисунке, Temp_1 является 4-байтовым вещественным значением. Принимающее master-устройство должно быть способно снова составить из двух принятых слов ожидаемое вещественное значение.
- Элементы требуют дополнительных символических ссылок при использовании в программе пользователя. В то время как ссылка на первый элемент простого массива имеет вид "HR_DB".Array[0], ссылка на элемент этого типа имеет вид "HR_DB".Data.Temp_1.

Вот как именованная составная структура появляется в редакторе блоков данных. Каждый элемент имеет уникальное имя с несколькими размерами и типами данных.

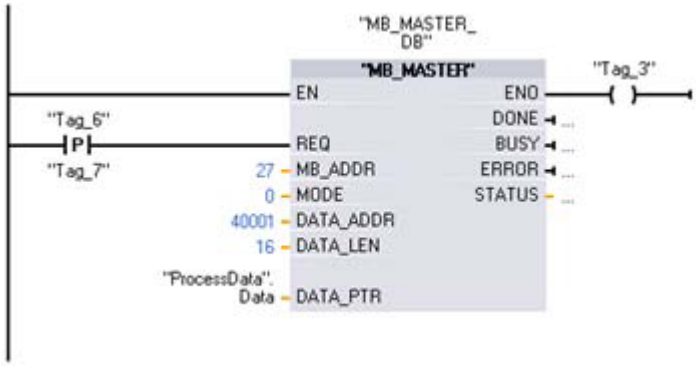
HR_DB

	Name	Data type	Offset	Initial value	Retain	Comment
1	Static					
2	Data	Struct	0.0			Modbus addresses 40001 to 400016
3	Temp_1	Real	0.0	0.0		40001 and 40002
4	Temp_2	Real	4.0	0.0		40003 and 40004
5	Good_Count	UDint	8.0	0		40005 and 40006
6	Bad_Count	UDint	12.0	0		40007 and 40008
7	Rework_Count	UDint	16.0	0		40009 and 40010
8	Line_Stops	Int	20.0	0		40011
9	Avg_Time	Int	22.0	0		40012
10	Long_Code	DWord	24.0	0		40013 and 40014
11	Code_1	Word	28.0	0		40015
12	Code_2	Word	30.0	0		40016

Ниже показано соответствие имен элементов массива адресам Modbus.

"HR_DB".Data.Temp_1	Адреса Modbus 40001 и 40002
"HR_DB".Data.Temp_2	Адреса Modbus 40003 и 40004
"HR_DB".Data.Good_Count	Адреса Modbus 40005 и 40006
"HR_DB".Data.Bad_Count	Адреса Modbus 40007 и 40008
"HR_DB".Data.Rework_Count	Адреса Modbus 40009 и 40010
"HR_DB".Data.Line_Stops	Адрес Modbus 400011
"HR_DB".Data.Avg_Time	Адрес Modbus 400012
"HR_DB".Data.Long_Code	Адрес Modbus 40013 и 40014
"HR_DB".Data.Code_1	Адрес Modbus 40015
"HR_DB".Data.Code_2	Адрес Modbus 40016

Другой CPU S7-1200, работающий в качестве master-устройства Modbus, может использовать команду MB_MASTER и идентичную структуру данных для получения блока данных от CPU S7-1200, работающего как slave-устройство Modbus. Эта команда master-устройства Modbus скопирует все 16 слов данных непосредственно из блока данных HR_DB slave-устройства блок данных в блок данных ProcessData master-устройства, как это показано ниже.



ProcessData

	Name	Data type	Offset	Initial value	Retain	Comment
1	Static					
2	Data	Struct	0.0			Modbus addresses 40001 to 400016
3	Temp_1	Real	0.0	0.0		40001 and 40002
4	Temp_2	Real	4.0	0.0		40003 and 40004
5	Good_Count	UDint	8.0	0		40005 and 40006
6	Bad_Count	UDint	12.0	0		40007 and 40008
7	Rework_Count	UDint	16.0	0		40009 and 40010
8	Line_Stops	Int	20.0	0		40011
9	Avg_Time	Int	22.0	0		40012
10	Long_Code	DWord	24.0	0		40013 and 40014
11	Code_1	Word	28.0	0		40015
12	Code_2	Word	30.0	0		40016

Для передачи одинаковых или различных структур из нескольких slave-устройств Modbus может быть использован ряд адресов блока данных Data_PTR master-устройства Modbus.

Коды условий

Значение STATUS (W#16#....)	**Описание**	
80C8	Заданное время ожидания ответа (см. RCVTIME или MSGTIME) равно 0	
80D1	Приемник издал запрос на управление потоком, чтобы остановить активную передачу и не возобновлять ее в течение указанного времени ожидания. Эта ошибка генерируется также при аппаратном управлении потоком, если приемник не объявляет о готовности к приему (CTS) в течение указанного времени ожидания.	
80D2	Запрос на передачу был отменен, так как не был получен сигнал о готовности (DSR) от аппаратуры передачи данных (DCE)	
80E0	Сообщение было завершено, так как приемный буфер полон	
80E1	Сообщение было завершено в результате ошибки контроля четности	
80E2	Сообщение было завершено в результате ошибки кадрирования	
80E3	Сообщение было завершено в результате ошибки переполнения	
80E4	Сообщение было завершено в результате того, что указанная длина превышает общий размер буфера	
8180	Неверное значение идентификатора порта	
8186	Неверный адрес станции Modbus	
8187	Неверный указатель на DB MB_HOLD_REG	
818C	Указатель на DB безопасного типа для MB_HOLD_REG (должен быть DB классического типа)	
	Код ответа, отправленный master-устройству Modbus (B#16#..)	
8380	Нет ответа	Ошибка CRC
8381	01	Код функции не поддерживается
8382	Нет ответа	Ошибка длины данных
8383	02	Ошибка адреса данных
8384	03	Ошибка значения данных
8385	03	Значение кода диагностических данных не поддерживается (код функции 08)

PROFINET

7

CPU S7-1200 имеет встроенный порт PROFINET, который поддерживает как Ethernet, так и коммуникационные стандарты на основе TCP/IP. CPU S7-1200 поддерживает следующие прикладные протоколы:

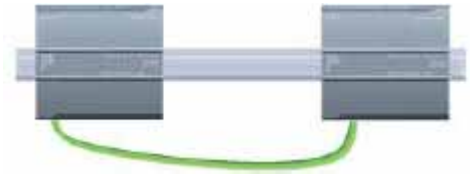
- Протокол управления передачей (Transport Control Protocol, TCP)
- ISO on TCP (RFC 1006)

CPU S7-1200 может обмениваться данными с другим CPU S7-1200, с устройством программирования STEP 7 Basic, с устройствами человеко-машинного интерфейса и с устройствами других производителей, использующими коммуникационные протоколы стандарта TCP. Имеются два способа обмена данными с помощью PROFINET:

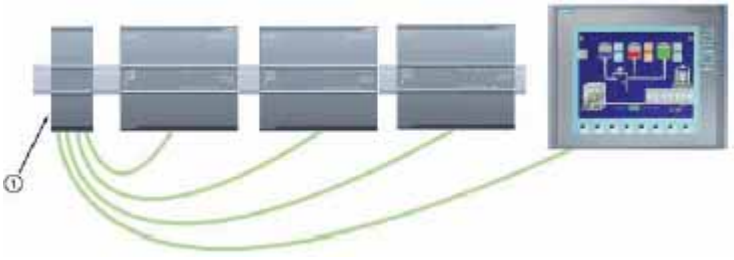
- Прямое соединение: Прямой обмен данными применяется, когда вы используете устройство программирования, устройство человеко-машинного интерфейса или другой CPU, соединенный с одним CPU.
- Соединение через сеть: Обмен данными через сеть используется, когда вы используете более двух устройств (например, CPU, устройства человеко-машинного интерфейса, устройства программирования и устройства других производителей).

Прямое соединение: Устройство программирования, соединенное с CPU S7-1200

Прямое соединение: Устройство человеко-машинного интерфейса, соединенное с CPU S7-1200



Прямое соединение: Соединение CPU S7-1200 с другим CPU S7-1200



Соединение через сеть: Более двух устройств, соединенных друг с другом с помощью коммутатора Ethernet CSM1277 ①

Коммутатор Ethernet не требуется для прямого соединения между устройством программирования или устройством человеко-машинного интерфейса и CPU. Коммутатор Ethernet необходим для сети с более чем двумя CPU или устройствами человеко-машинного интерфейса. Монтируемый на стойке и имеющий 4 порта коммутатор Ethernet CSM1277 фирмы Siemens может использоваться для соединения вашего CPU и устройств человеко-машинного интерфейса. Порт PROFINET на CPU S7-1200 не содержит коммутирующего устройства Ethernet.

Максимальное количество соединений для порта PROFINET

Порт PROFINET на CPU поддерживает следующие одновременные соединения для обмена данными.

- 3 соединения для обмена данными между устройством человеко-машинного интерфейса и CPU
- 1 соединение для обмена данными между устройством программирования (PG) и CPU
- 8 соединений для коммуникаций программы S7-1200 с помощью команд типа T-блоков (TSEND_C, TRCV_C, TCON, TDISCON, TSEN, TRCV)
- 3 соединения для пассивного CPU S7-1200, обменивающегося данными с активным CPU S7
  - Активный CPU S7 использует команды GET и PUT (S7-300 и S7-400) или команды ETHx_XFER (S7-200).
  - Активное коммуникационное соединение S7-1200 возможно только с помощью команд типа T-блоков.

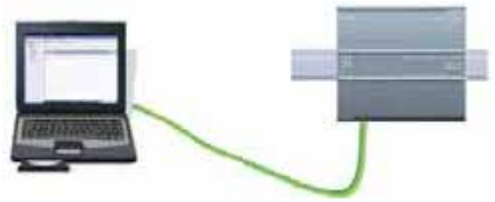
Ограниченные точки доступа к услугам транспортного уровня (TSAP) или номера портов для связи через ISO и TCP

Если вы используете команду "TCON" для создания и установления пассивной коммуникационной связи, то следующие адреса портов ограничены и не должны использоваться:

- ISO TSAP (пассивная): 01.00, 01.01, 02.00, 02.01, 03.00, 03.01
- Порт TCP (пассивный): 5001, 102, 123, 20, 21, 25, 34962, 34963, 34964, 80

7.1 Обмен данными с устройством программирования

CPU может обмениваться данными в сети с устройством программирования STEP 7 Basic.



При установлении связи между CPU и устройством программирования примите во внимание следующее:

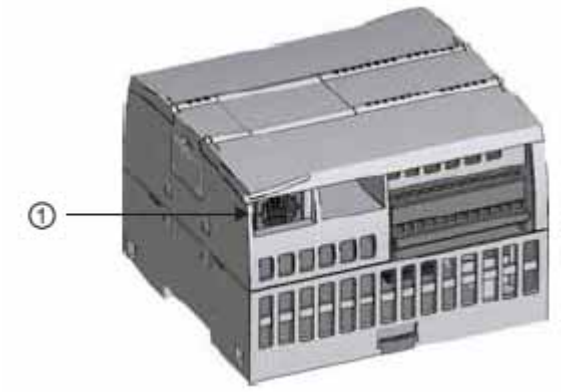
- Конфигурирование/настройка: Требуется конфигурирование аппаратуры.
- Для связи один к одному не нужен коммутатор Ethernet; коммутатор Ethernet необходим для более чем двух устройств в сети.

7.1.1 Создание аппаратного коммуникационного соединения

Интерфейсы PROFINET устанавливают физическое соединение между устройством программирования и CPU. Так как в CPU встроена функция автоматического распознавания приемного и передающего кабелей (Auto-Cross-Over), то для интерфейса может быть использован как стандартный, так и перекрёстный кабель Ethernet. Для непосредственного присоединения устройства программирования к CPU коммутатор Ethernet не требуется.

Для создания аппаратного соединения между устройством программирования и CPU действуйте следующим образом:

1. Установите CPU (стр. 26).
2. Вставьте кабель Ethernet в порт PROFINET, как показано ниже.
3. Подключите кабель Ethernet к устройству программирования.



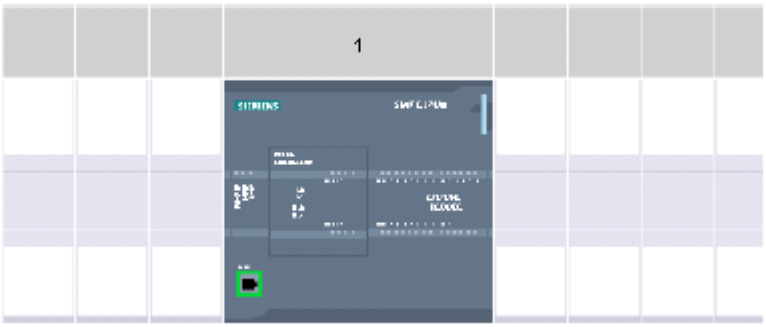
① Порт PROFINET

Для подключения PROFINET имеется необязательный компенсатор натяжения кабеля.

7.1.2 Конфигурирование устройств

Если вы уже создали проект с CPU, откройте сой проект в портале TIA.

Если нет, создайте проект и вставьте CPU (стр. 78) в стойку. В представленном ниже проекте CPU показан в отображении набора устройств портала TIA.



7.1.3 Назначение IP-адресов

7.1.3.1 Назначение IP-адресов устройству программирования и сетевым устройствам

Если ваше устройство программирования использует встроенную адаптерную плату, подключенную к ЛВС вашей установки (и, возможно, к всемирной сети), идентификатор сети IP-адреса и маска подсети вашего CPU и встроенной адаптерной платы устройства программирования должны быть в точности одинаковыми. Идентификатор (ID) сети является первой частью IP-адреса (первые три октета, например, **211.154.184**.16), которая определяет, какую IP-сеть вы используете. Маска подсети обычно имеет значение **255.255.255.0**; однако, так как ваш компьютер находится в ЛВС установки, то маска подсети может иметь другие значения (например, **255.255.254.0**), чтобы создавать уникальные подсети. Маска подсети, комбинируемая с помощью логической операции И с IP-адресом устройства, определяет границы IP-подсети.

Указание

В сценарии всемирной сети, где ваши устройства программирования, сетевые устройства и IP-маршрутизаторы будут обмениваться данными со всем миром, им должны быть назначены уникальные IP-адреса во избежание конфликта с другими пользователями сети. Для получения своего IP-адреса обратитесь в IT-отделение своей фирмы, персонал которой знаком с сетями вашей установки.

Если ваше устройство программирования использует адаптерную плату Ethernet/USB, подключенную к изолированной сети, то идентификатор сети IP-адреса и маска подсети вашего CPU и адаптерной платы Ethernet/USB устройства программирования должны быть в точности одинаковыми. Идентификатор сети является первой частью IP-адреса (первые три октета) (например, **211.154.184**.16), которая определяет, какую IP-сеть вы используете. Маска подсети обычно имеет значение **255.255.255.0**. Маска подсети, комбинируемая с помощью логической операции И с IP-адресом устройства, определяет границы IP-подсети.

Указание

Адаптерная плата Ethernet/USB полезна, если вы не хотите подключать свой CPU к ЛВС фирмы. Эта конструкция особенно полезна при начальном тестировании или приемосдаточных испытаниях.

Адаптерная плата устройства программирования	Тип сети	IP-адрес	Маска подсети
Встроенная адаптерная плата	Подключена к ЛВС вашей установки (и, возможно, к всемирной сети)	Идентификатор сети вашего CPU и встроенной адаптерной платы устройства программирования должны быть в точности одинаковыми. Идентификатор сети является первой частью IP-адреса (первыми тремя октетами) (например, 211.154.184.16), которая определяет, какую IP-сеть вы используете)	Маска подсети вашего CPU и встроенной адаптерной платы должны быть в точности одинаковыми. Маска подсети обычно имеет значение 255.255.255.0; однако, так как ваш компьютер находится в ЛВС установки, то маска подсети может иметь другие значения (например, 255.255.254.0), чтобы создавать уникальные подсети. Маска подсети, комбинируемая с помощью логической операции И с IP-адресом устройства, определяет границы IP-подсети.
Адаптерная плата Ethernet/USB	Подключена к изолированной сети	Идентификатор сети вашего CPU и адаптерной платы Ethernet/USB устройства программирования должны быть в точности одинаковыми. Идентификатор сети является первой частью IP-адреса (первыми тремя октетами) (например, 211.154.184.16), которая определяет, какую IP-сеть вы используете)	Маска подсети вашего CPU и адаптерной платы Ethernet/USB должны быть в точности одинаковыми. Маска подсети обычно имеет значение **255.255.255.0**. Маска подсети, комбинируемая с помощью логической операции И с IP-адресом устройства, определяет границы IP-подсети.

Назначение и проверка IP-адреса вашего устройства программирования с помощью "My Network Places [Сетевая среда]" (на вашем рабочем столе)

Вы можете назначить и проверить IP-адрес вашего устройства программирования с помощью следующих команд меню:

- (Щелкните правой клавишей мыши) "My Network Places [Сетевая среда]"
- "Properties [Свойства]"
- (Щелкните правой клавишей мыши) "Local Area Connection [Соединение с ЛВС]"
- "Properties [Свойства]"

В диалоговом окне "Local Area Connection Properties [Свойства соединения с ЛВС]" в поле "This connection uses the following items: [Это соединение использует следующие объекты]" прокрутите до "Internet Protocol (TCP/IP)". Щелкните на "Internet Protocol (TCP/IP)", а затем щелкните на кнопке "Properties [Свойства]". Выберите "Obtain an IP-address automatically (DHCP) [Получить IP-адрес автоматически (DHCP)]" или "Use the following IP-address [Использовать следующий IP-адрес]" (чтобы ввести статический IP-адрес).

Указание

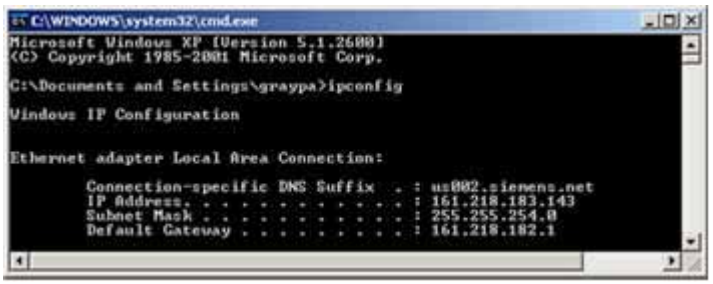
Протокол динамической конфигурации хоста (Dynamic Host Configuration Protocol, DHCP) автоматически назначает IP-адрес вашему устройству программирования при включении питания из сервера DHCP.

Проверка IP-адреса вашего устройства программирования с помощью команд "ipconfig" и "ipconfig /all"

Вы можете также проверить IP-адрес вашего устройства программирования, и, если необходимо, IP-адрес вашего IP-маршрутизатора (шлюза) с помощью следующих команд меню:

- Кнопка "Start [Пуск]" (на вашем рабочем столе)
- "Run [Выполнить]"

В диалоговом окне "Run" в поле "Open [Открыть]" введите "cmd" и щелкните на кнопке "OK". В отобразившемся диалоговом окне "C:\WINDOWS\system32\cmd.exe" введите команду "ipconfig". Пример результата показан ниже:



Дальнейшую информацию вы можете отобразить командой "ipconfig /all". Здесь можно найти тип адаптерной платы вашего устройства программирования и Ethernet-адрес (MAC):

Назначение IP-адреса CPU

Для назначения IP-адреса CPU можно использовать следующие два способа:

- Назначить IP-адрес в режиме online
- Сконфигурировать IP-адрес в своем проекте

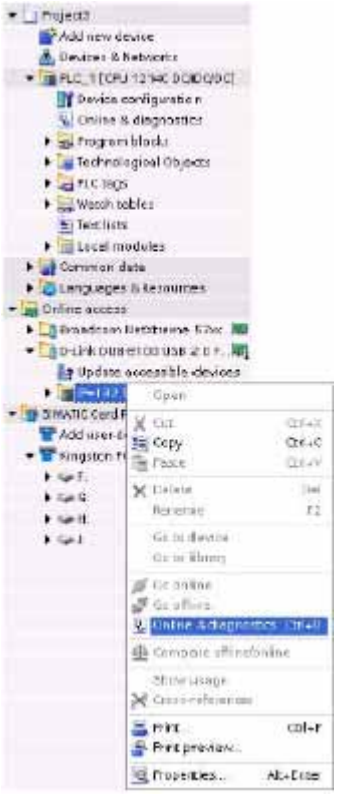
7.1.3.2 Назначение IP-адресов в режиме online

Вы можете назначить IP-адрес сетевому устройству в режиме online. Это особенно полезно при первом конфигурировании устройства.

Для назначения IP-адреса в режиме online действуйте следующим образом:

1. В дереве проекта (Project tree) проверьте, что CPU не назначен IP-адрес, с помощью следующих команд меню:
   - "Online access [Онлайновый доступ]"
   - <Адаптерная плата для сети, в которой находится устройство>
   - "Update accessible devices [Обновить доступные устройства]"

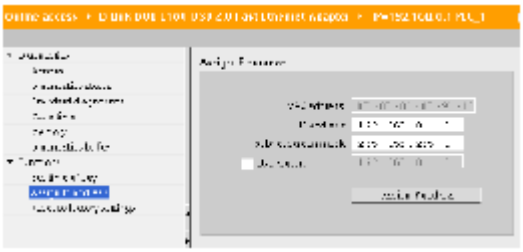
2. В дереве проекта выберите следующие команды меню:
   - "Online access [Онлайновый доступ]"
   - <Адаптерная плата для сети, в которой находится устройство>
   - "Update accessible devices [Обновить доступные устройства]"
   - <адрес устройства>
   - "Online & diagnostics [Онлайновый режим и диагностика]"



3. В диалоговом окне "Online & diagnostics" выберите следующие команды меню:

- "Functions [Функции]"
- "Assign IP address [Назначить IP-адрес]"

4. В поле "IP-address" введите свой новый IP-адрес.



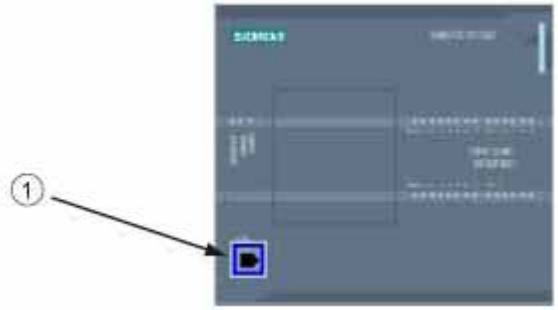
5. В дереве проекта (Project tree) проверьте, что ваш новый IP-адрес назначен CPU, с помощью следующих команд меню:

- "Online access [Онлайновый доступ]"
- <Адаптер для сети, в которой находится устройство>
- "Update accessible devices [Обновить доступные устройства]"

7.1.3.3 Конфигурирование IP-адреса в вашем проекте

Конфигурирование интерфейса PROFINET

После того как вы сконфигурировали стойку с CPU (стр. 252), вы можете сконфигурировать параметры интерфейса PROFINET. Для этого щелкните на зеленом поле PROFINET на CPU, чтобы выбрать порт PROFINET. Вкладка "Properties [Свойства]" в окне просмотра параметров отображает порт PROFINET.



① Порт PROFINET

Конфигурирование IP-адреса

Адрес Ethernet (MAC-адрес): В сети PROFINET каждому устройству производителем для идентификации назначается MAC-адрес (Media Access Control address [адрес протокола управления доступом к передающей среде]). MAC-адрес состоит из шести групп по две шестнадцатеричных цифры, разделенных дефисами (-) или двоеточиями (:) в порядке передачи (например, 01-23-45-67-89-AB или 01:23:45:67:89:AB).

IP-адрес: Каждое устройство должно также иметь адрес протокола Интернет (Internet Protocol, IP). Этот адрес позволяет устройству поставлять данные через более сложную сеть с маршрутизацией.

Каждый IP-адрес делится на четыре 8-битовых сегмента и представляется в десятичном формате с разделительными точками (например, 211.154.184.16). Первая часть IP-адреса используется для идентификатора сети (В какой сети вы находитесь?), и вторая часть адреса является идентификатором хоста (уникальным для каждого устройства в сети). IP-адрес 192.168.x.y является стандартным обозначением, которое распознается как часть ведомственной или частной сети, которая находится вне сети Интернет.

Маска подсети: Подсеть – это логическая группировка с соединенных друг с другом сетевых устройств. Узлы подсети находятся обычно недалеко друг от друга в локальной вычислительной сети (ЛВС). Маска (маска подсети или маска сети) определяет границы IP-подсети.

Маска подсети 255.255.255.0 обычно пригодна для небольших локальных сетей. Это значит, что все IP-адреса в этой сети должны иметь одинаковые первые 3 октета, и различные устройства в этой сети идентифицируются последним октетом (8-битовым полем). Примером этого является назначение маски подсети 255.255.255.0 и IP-адресов от 192.168.2.0 до 192.168.2.255 отдельным устройствам в небольшой локальной сети.

Единственное соединение между различными подсетями осуществляется через маршрутизатор. Если используются подсети, то должен применяться IP-маршрутизатор.

IP-маршрутизатор: Маршрутизаторы являются связующими звеньями между ЛВС. С помощью маршрутизатора компьютер, находящийся в ЛВС, может посылать сообщения в любые другие сети, за которыми, возможно, имеются другие ЛВС. Если цель данных находится за пределами ЛВС, то маршрутизатор передает данные дальше в другую сеть или группу сетей, где эти данные могут быть доставлены по назначению.
Маршрутизаторы для передачи и приема пакетов данных используют IP-адреса.

Свойства IP-адресов: В окне Properties [Свойства] выберите компонент конфигурации "Ethernet address [Адрес Ethernet]". Портал TIA отображает диалоговое окно для конфигурирования адреса Ethernet, в котором вы можете назначить проекту программного обеспечения IP-адрес CPU, в который загружается проект.

Указание

У CPU нет заранее сконфигурированного IP-адреса. Вы должны назначить IP-адрес для CPU вручную. Если ваш CPU соединен с маршрутизатором в сети, вы должны также ввести IP-адрес маршрутизатора. Все IP-адреса конфигурируются, когда вы загружаете проект.

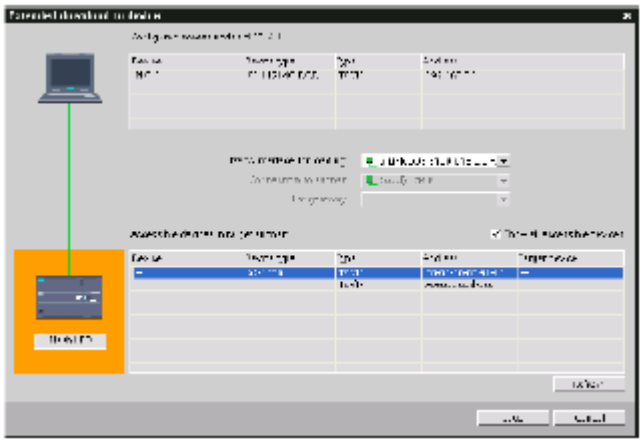
Дальнейшую информацию вы найдете под заголовком "Назначение IP-адресов устройству программирования и сетевым устройствам".

Следующая таблица определяет параметры IP-адреса:

Параметр		Описание
Subnet [Подсеть]	Имя подсети, к которой подключено устройство. Щелкните на кнопке "Add new subnet [Добавить новую подсеть]", чтобы создать новую подсеть. Установкой по умолчанию является "Not connected [Не соединено]". Возможны два типа соединений: • Установка по умолчанию "Not connected" предоставляет локальное соединение. • Подсеть необходима, если ваша сеть содержит два или более устройств.	
IP protocol	IP address [IP-адрес]	Назначенный IP-адрес для CPU
	Subnet mask [Маска подсети]	Назначенная маска подсети
	Use IP router [Использовать IP-маршрутизатор]	Щелкните на триггерной кнопке, если используется IP-маршрутизатор
	Router address [Адрес маршрутизатора]	Назначенный IP-адрес для маршрутизатора, если он имеется

7.1.4 Тестирование сети PROFINET

По окончании конфигурирования загрузите программу в CPU. Все IP-адреса конфигурируются при загрузке проекта.



Назначение IP-адреса устройству в режиме online

У CPU S7-1200 нет заранее сконфигурированного IP-адреса. IP-адрес CPU вы должны задать вручную.

Для назначения IP-адреса устройству в режиме online выполните шаги, описанные под заголовком "Назначение IP-адресов в режиме online".

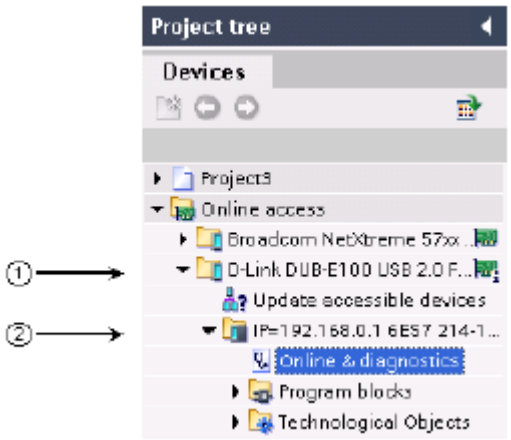
Для назначения IP-адреса в своем проекте вы должны сконфигурировать IP-адрес в конфигурации устройств, сохранить конфигурацию и загрузить ее в ПЛК. Дальнейшую информацию вы найдете под заголовком "Конфигурирование IP-адреса в вашем проекте".

Указание

Если вы назначили IP-адреса в режиме online, то вы можете изменять эти IP-адреса в конфигурации аппаратуры как online, так и offline.

Если вы назначили IP-адреса в конфигурации аппаратуры в режиме offline, то вы можете изменять IP-адреса, назначенные в проекте, в конфигурации аппаратуры только в режиме offline.

Используйте "Online access [Онлайновый доступ]" для отображения IP-адресов подключенных CPU, как показано ниже.



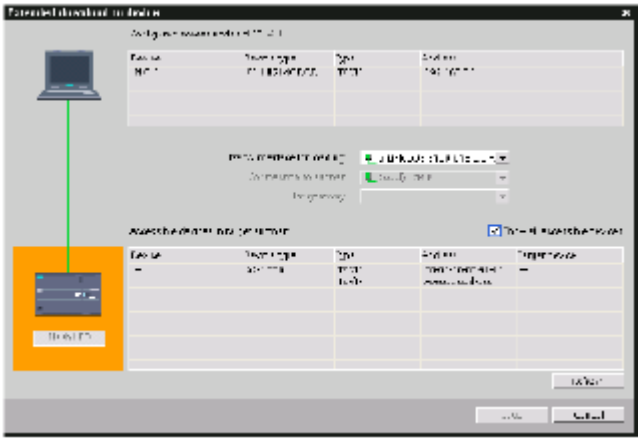
① Вторая из двух сетей Ethernet в этом устройстве программирования
② IP-адрес единственного CPU S7-1200 в этой сети Ethernet

Указание

Отображаются все сконфигурированные сети устройства программирования. Вы должны выбрать нужную сеть, чтобы отобразить необходимый IP-адрес CPU S7-1200.

Использование диалога "Extended download to device" для проверки подключения сетевых устройств

Функция CPU S7-1200 "Download to device [Загрузить в устройство]" и ее диалоговое окно "Extended download to device [Расширенная загрузка в устройство]" могут показать все имеющиеся сетевые устройства, а также всем ли устройствам назначены уникальные IP-адреса. Для отображения всех доступных и имеющихся устройств с назначенными им MAC- и IP-адресами активизируйте триггерную кнопку "Show all accessible devices [Показать все доступные устройства]".



Если желаемого сетевого устройства нет в этом списке, то связь с этим устройством была по какой-то причине прервана. Это устройство и сеть должны быть обследованы на наличие аппаратных и/или конфигурационных ошибок.

7.2 Обмен данными между устройствами человеко-машинного интерфейса и ПЛК

CPU поддерживает коммуникационные соединения PROFINET с устройствами человеко-машинного интерфейса. При установлении связи между CPU и устройствами человеко-машинного интерфейса должны быть выполнены следующие условия:

Конфигурирование/настройка:

- Порт PROFINET на CPU должен быть сконфигурирован для соединения с устройством человеко-машинного интерфейса.
- Это устройство человеко-машинного интерфейса должно быть установлено и сконфигурировано.
- Конфигурационные данные устройства человеко-машинного интерфейса являются частью проекта CPU и могут быть сконфигурированы и загружены из этого проекта.
- Для связи один к одному не нужен коммутатор Ethernet; коммутатор Ethernet необходим для более чем двух устройств в сети.

Указание

Монтируемый на стойке и имеющий 4 порта коммутатор Ethernet CSM1277 фирмы Siemens может использоваться для соединения вашего CPU и устройств человеко-машинного интерфейса. Порт PROFINET на CPU S7-1200 не содержит коммутирующего устройства Ethernet.

Поддерживаемые функции:

- Устройство человеко-машинного интерфейса может считывать и записывать данные в CPU.
- На основе информации, полученной из CPU, могут запускаться сообщения.
- Диагностика системы

Указание

WinCC Basic и STEP 7 Basic являются компонентами портала TIA. Дальнейшую информацию о конфигурировании устройств человеко-машинного интерфейса вы найдете в WinCC Basic.

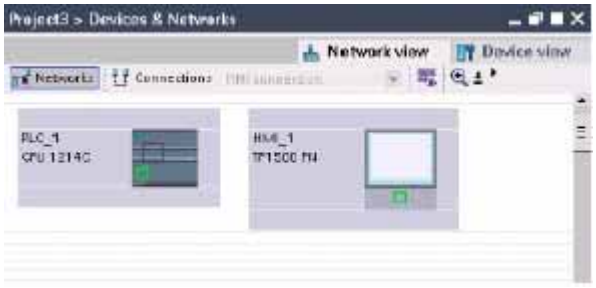
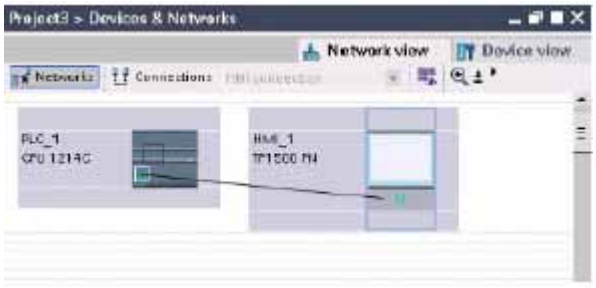
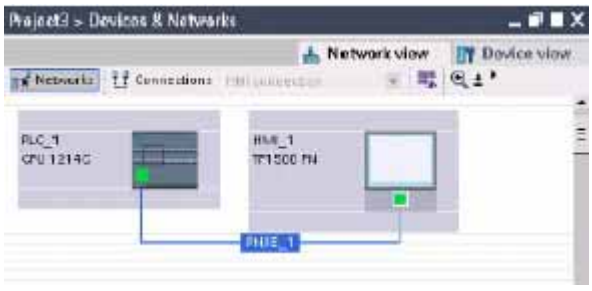
Шаги, необходимые для конфигурирования обмена данными между устройством человеко-машинного интерфейса и CPU

Шаг	Задача
1	Создание аппаратного коммуникационного соединения Интерфейс PROFINET устанавливает физическое соединение между устройством человеко-машинного интерфейса и CPU. Так как в CPU встроена функция автоматического распознавания приемного и передающего кабелей (Auto-Cross-Over), то для интерфейса может быть использован как стандартный, так и перекрёстный кабель Ethernet. Для соединения устройства человеко-машинного интерфейса и CPU коммутатор Ethernet не требуется. Дальнейшую информацию вы найдете под заголовком "Обмен данными с устройством программирования: Создание аппаратного коммуникационного соединения" (стр. 251).
2	Конфигурирование устройств Дальнейшую информацию вы найдете под заголовками "Обмен данными с устройством программирования. Конфигурирование устройств" (стр. 252).
3	Конфигурирование логических сетевых соединений между устройством человеко-машинного интерфейса и CPU Дальнейшую информацию вы найдете под заголовком "Конфигурирование логических сетевых соединений между устройством человеко-машинного интерфейса и CPU (стр. 264).
4	Конфигурирование IP-адреса в вашем проекте Используйте тот же самый процесс конфигурирования; Однако вы должны сконфигурировать IP-адреса для устройства человеко-машинного интерфейса и CPU. Дальнейшую информацию вы найдете под заголовком "Обмен данными с устройством программирования: Конфигурирование IP-адреса в вашем проекте" (стр. 257).
5	Тестирование сети PROFINET Вы должны загрузить конфигурацию для каждого CPU. Дальнейшую информацию вы найдете под заголовком "Обмен данными с устройством программирования: Тестирование сети PROFINET (стр. 259).

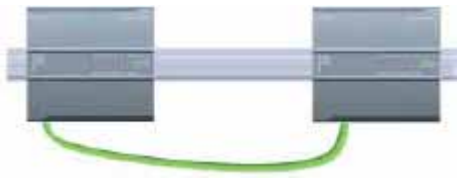
7.2.1 Конфигурирование логических сетевых соединений между устройством человеко-машинного интерфейса и CPU

После конфигурирования стойки с CPU вы можете приступить к конфигурированию своих сетевых соединений.

В портале "Devices & Networks [Устройства и сети]" вы можете использовать "Network view [Отображение сети]" для создания сетевых соединений между устройствами в вашем проекте. Для создания соединения в сети Ethernet выберите зеленое поле (Ethernet) на CPU. Проведите мышью линию к полю Ethernet на устройстве человеко-машинного интерфейса. Отпустите клавишу мыши, и ваше Ethernet-соединение создано.

Действие	Результат
Выберите "Network view [Отображение сети]" для отображения устройств, подлежащих соединению.	
Выберите порт на устройстве и протяните линию к порту на втором устройстве.	
Отпустите клавишу мыши, чтобы создать сетевое соединение.	

7.3 Обмен данными между ПЛК



CPU может обмениваться данными с другим CPU в сети, используя команды TSEND_C и TRCV_C.

При установлении связи между двумя CPU обратите внимание на следующее:

- Конфигурирование/настройка: Требуется конфигурирование аппаратуры.
- Поддерживаемые функции: Чтение и запись данных в равноправном CPU
- Для связи один к одному не нужен коммутатор Ethernet; коммутатор Ethernet необходим для более чем двух устройств в сети.

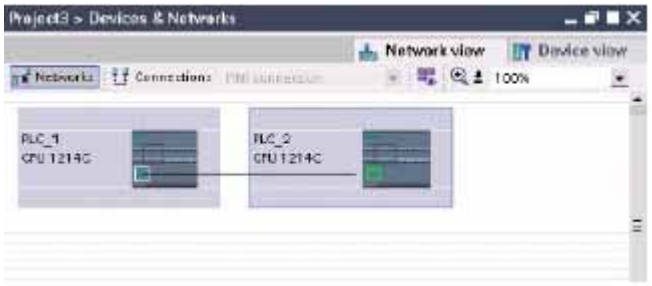
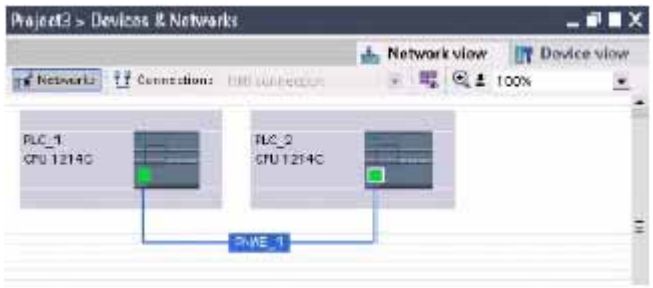
Шаги, необходимые для конфигурирования связи между двумя CPU

Шаг	Задача
1	Создание аппаратного коммуникационного соединения Интерфейс PROFINET устанавливает физическое соединение между двумя CPU. Так как в CPU встроена функция автоматического распознавания приемного и передающего кабелей (Auto-Cross-Over), то для интерфейса может быть использован как стандартный, так и перекрёстный кабель Ethernet. Для соединения двух CPU коммутатор Ethernet не требуется. Дальнейшую информацию вы найдете под заголовком "Обмен данными с устройством программирования: Создание аппаратного коммуникационного соединения".
2	Конфигурирование устройств Вы должны создать два проекта с CPU в каждом проекте. Дальнейшую информацию вы найдете под заголовком "Обмен данными с устройством программирования: Конфигурирование устройств".
3	Конфигурирование логических сетевых соединений между двумя CPU Дальнейшую информацию вы найдете под заголовком "Конфигурирование логических сетевых соединений между двумя CPU" (стр. 266).
4	Конфигурирование IP-адреса в вашем проекте Используйте тот же самый процесс конфигурирования; однако вы должны сконфигурировать IP-адреса для двух CPU (например, ПЛК _1 и ПЛК _2). Дальнейшую информацию вы найдете под заголовком "Обмен данными с устройством программирования: Конфигурирование IP-адреса в вашем проекте".
5	Конфигурирование параметров передачи и приема Вы должны сконфигурировать команды TSEND_C и TRCV_C в обоих CPU для установления связи между ними. Дальнейшую информацию вы найдете под заголовком " Конфигурирование параметров передачи и приема" (стр. 267).
6	Тестирование сети PROFINET Вы должны загрузить конфигурацию для каждого CPU. Дальнейшую информацию вы найдете под заголовком "Обмен данными с устройством программирования: Тестирование сети PROFINET".

7.3.1 Конфигурирование логических сетевых соединений между двумя CPU

После конфигурирования стойки с CPU вы можете приступить к конфигурированию своих сетевых соединений.

В портале "Devices & Networks [Устройства и сети]" вы можете использовать "Network view [Отображение сети]" для создания сетевых соединений между устройствами в вашем проекте. Для создания соединения в сети PROFINET выберите зеленое поле (PROFINET) на первом ПЛК. Проведите мышью линию к полю PROFINET на втором ПЛК. Отпустите клавишу мыши, и ваше PROFINET-соединение создано.

Действие	**Результат**
Выберите "Network view [Отображение сети]" для отображения устройств, подлежащих соединению.	
Выберите порт на одном устройстве и протяните линию к порту на втором устройстве.	
Отпустите клавишу мыши, чтобы создать сетевое соединение.	

7.3.2 Конфигурирование параметров передачи и приема

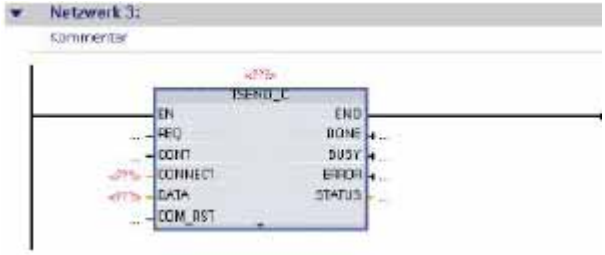
Для установления соединений между двумя CPU используются блоки передачи (T-блоки). Прежде чем CPU сможет включиться в обмен данными в сети PROFINET, вы должны сконфигурировать параметры для передачи и приема сообщений. Эти параметры определяют, как будет протекать обмен данными при передаче и приеме сообщений от целевого устройства.

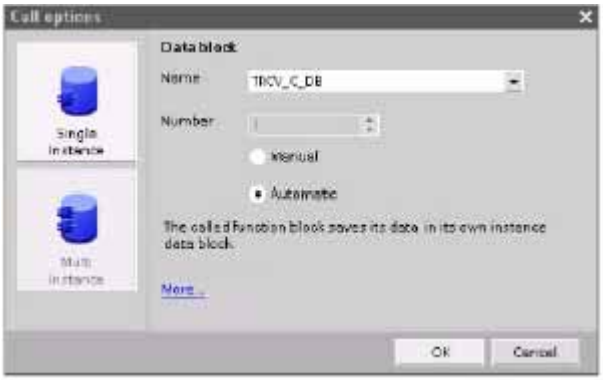
7.3.2.1 Конфигурирование параметров передачи для TSEND_C

Команда TSEND_C

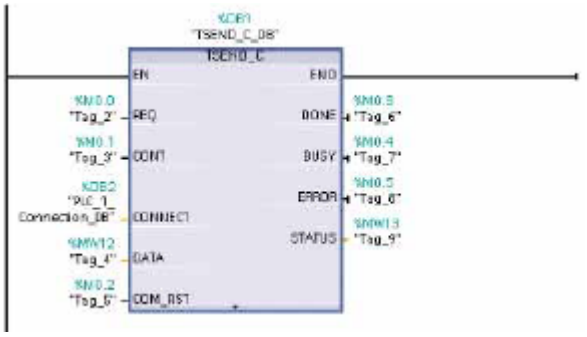
Команда TSEND_C (стр. 181) устанавливает связь с партнерской станцией. Эта связь создается, устанавливается и автоматически контролируется, пока не будет подана команда на разъединение. Команда TSEND_C объединяет в себе функции команд TCON, TDISCON и TSEND.

Из конфигурации устройств в STEP 7 вы можете установить, как команда TSEND_C должна передать данные. Сначала вы вставляете эту команду в программу из папки "Communications [Связь]" через "Extended Instructions [Расширенный набор команд]". Команда отображается вместе с диалоговым окном "Call options [Параметры вызова]", где вы назначаете DB для хранения параметров команды TSEND_C.





Вы можете назначить адреса в памяти переменных для входов и выходов, как показано на следующем рисунке.



Конфигурирование общих параметров

Вы можете задать коммуникационные параметры в диалоговом окне Properties [Свойства] команды TSEND_C. Это диалоговое окно появляется в нижней части страницы, когда вы выделяете любую часть команды TSEND_C.

Конфигурирование параметров соединения

Каждый CPU имеет встроенный порт PROFINET, который поддерживает стандартный обмен данными через PROFINET. Поддерживаемые протоколы Ethernet описаны в следующих двух типах соединений:

Протокол	Имя протокола	Использование
RFC 1006	ISO on TCP	Фрагментация и восстановление сообщений
TCP	Transport Control Protocol [Протокол управления передачей]	Транспортировка кадров

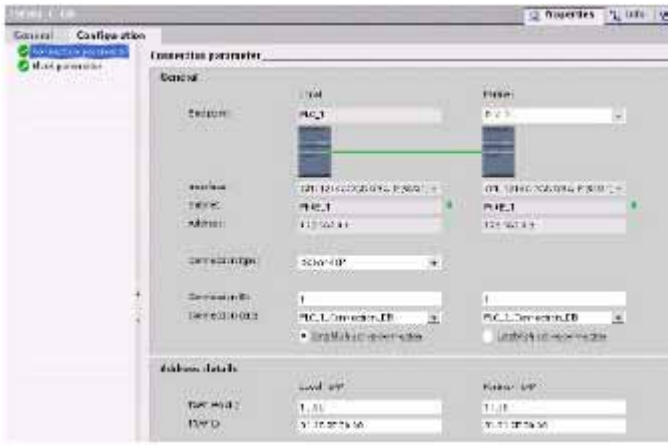
ISO on TCP (RFC 1006)

ISO on TCP - это механизм, который позволяет переносить ISO-приложения по сети TCP/IP. Этот протокол обладает следующими свойствами:

- Эффективный коммуникационный протокол, тесно связанный с аппаратным обеспечением
- Пригоден для объемов данных среднего и большого размера (до 8192 байтов)
- В отличие от TCP, сообщения характеризуются наличием идентификации конца данных и ориентированы на сообщения.
- Обладает способностью к маршрутизации; может использоваться в глобальной сети (WAN)
- Возможность динамического изменения длины данных.
- Из-за наличия программного интерфейса SEND / RECEIVE требует затрат времени на программирование.

С помощью точек доступа к услугам транспортного уровня (Transport Service Access Point, TSAP) протокол TCP допускает несколько соединений с одним IP-адресом (до 64K соединений). С помощью RFC 1006 TSAP однозначно идентифицируют эти соединения конечных коммуникационных пунктов с IP-адресом.

В разделе "Address Details [Подробности адреса]" диалогового окна Connection Parameters [Параметры соединения] вы определяете подлежащие использованию TSAP. TSAP в CPU вводится в поле "Local TSAP [Локальный TSAP]". TSAP, назначенный для соединения в партнерском CPU, вводится в поле "Partner TSAP [TSAP партнера]".



Параметр	Определение
General [Общие]	
End point: Partner [Конечный пункт: партнер]	Имя, назначенное CPU партнера (приемнику)
Interface [Интерфейс]	Имя, назначенное интерфейсам
Subnet [Подсеть]	Имя, назначенное подсетям
Address [Адрес]	Назначенные IP-адреса
Connection type [Тип соединения]	Тип протокола Ethernet
Connection ID [ID соединения]	Идентификационный номер
Connection data [Данные о соединении]	Адрес хранения данных локального и партнерского CPU
Active connection setup [Настройка активного соединения]	Селективная кнопка для выбора локального или партнерского CPU в качестве активного соединения
Address details [Подробности адреса]	
TSAP[1] (ASCII)	TSAP локального и партнерского CPU в формате ASCII
TSAP ID	TSAP локального и партнерского CPU в шестнадцатеричном формате

[1] При конфигурировании соединения с CPU S7-1200 через ISO on TCP используйте в расширении TSAP для пассивных коммуникационных партнеров только символы ASCII.

Протокол управления передачей (Transport Control Protocol, TCP)

TCP – это стандартный протокол, описанный в RFC 793: Transmission Control Protocol. Основной целью TCP является предоставление услуг, обеспечивающих надежное и безопасное соединение между парами процессов. Этот протокол обладает следующими свойствами:

- Эффективный коммуникационный протокол, так как он тесно связан с аппаратным обеспечением
- Пригоден для объемов данных среднего и большого размера (до 8192 байтов)
- Предоставляет значительно больше услуг для приложений, в особенности:
  - Восстановление в случае ошибки
  - Управление потоком
  - Надежность
- Протокол, ориентированный на соединения
- Может быть очень гибко использован с системами других производителей, которые поддерживают только TCP
- Возможность маршрутизации
- Применимы только статические длины данных.
- Сообщения квитируются.
- Приложения адресуются с помощью номеров портов.
- Большинство протоколов пользовательских приложений, например, TELNET и FTP, используют TCP.
- Из-за наличия программного интерфейса SEND / RECEIVE требует затрат времени на программирование.

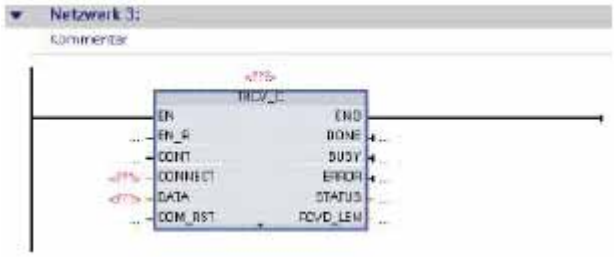
Параметр	Определение
General [Общие]	
End point: Partner [Конечный пункт: партнер]	Имя, назначенное CPU партнера (приемнику)
Interface [Интерфейс]	Имя, назначенное интерфейсам
Subnet [Подсеть]	Имя, назначенное подсетям
Address [Адрес]	Назначенные IP-адреса
Connection type [Тип соединения]	Тип протокола Ethernet
Connection ID [ID соединения]	Идентификационный номер
Connection data [Данные о соединении]	Адрес хранения данных локального и партнерского CPU
Active connection setup [Настройка активного соединения]	Селективная кнопка для выбора локального или партнерского CPU в качестве активного соединения
Address details [Подробности адреса]	
Port (decimal) [Порт (десятичный)]	Порт партнерского CPU в десятичном формате

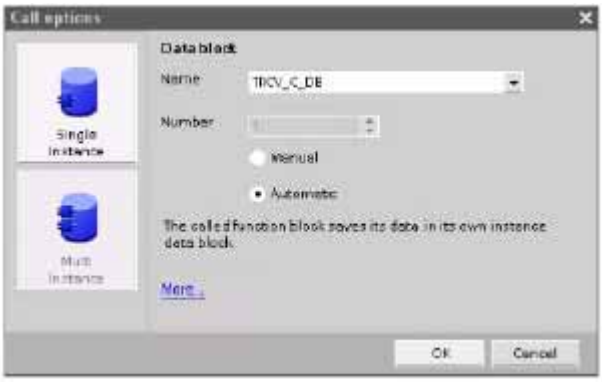
7.3.2.2 Конфигурирование параметров приема для TRCV_C

Команда TRCV_C

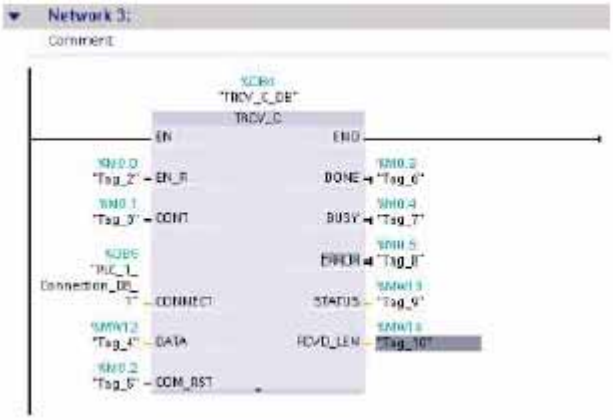
Команда TRCV_C (стр. 182) устанавливает связь с партнерской станцией. Связь создается, устанавливается и автоматически контролируется, пока она не будет разорвана этой командой. Команда TRCV_C объединяет в себе функции команд TCON, TDISCON, и TRCV.

Из конфигурации CPU в STEP 7 Basic вы можете установить, как команда TRCV_C должна получать данные. Сначала вы вставляете эту команду в программу из папки "Communications [Связь]" через "Extended Instructions [Расширенный набор команд]". Команда отображается вместе с диалоговым окном "Call options [Параметры вызова]", где вы назначаете DB для хранения параметров команды TRCV_C.





Вы можете назначить адреса в памяти переменных для входов и выходов, как показано на следующем рисунке.



Конфигурирование общих параметров

Вы можете задать коммуникационные параметры в диалоговом окне Properties [Свойства] команды TRCV_C. Это диалоговое окно появляется в нижней части страницы, когда вы выделяете любую часть команды TRCV_C.

Конфигурирование параметров соединения

Каждый CPU имеет встроенный порт PROFINET, который поддерживает стандартный обмен данными через PROFINET. Поддерживаемые протоколы Ethernet описаны в следующих двух типах соединений:

Протокол	Имя протокола	Использование
RFC 1006	ISO on TCP	Фрагментация и восстановление сообщений
TCP	Transport Control Protocol [Протокол управления передачей]	Транспортировка кадров

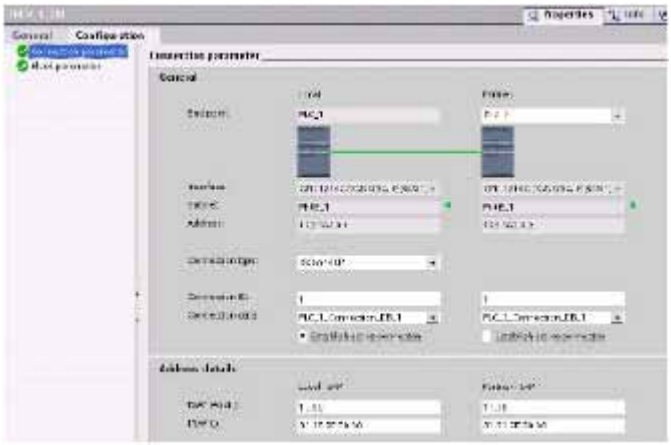
ISO on TCP (RFC 1006)

ISO on TCP - это механизм, который позволяет переносить ISO-приложения по сети TCP/IP. Этот протокол обладает следующими свойствами:

- Эффективный коммуникационный протокол, тесно связанный с аппаратным обеспечением
- Пригоден для объемов данных среднего и большого размера (до 8192 байтов)
- В отличие от TCP, сообщения характеризуются наличием идентификации конца данных и ориентированы на сообщения.
- Обладает способностью к маршрутизации; может использоваться в глобальной сети (WAN)
- Возможность динамического изменения длины данных.
- Из-за наличия программного интерфейса SEND / RECEIVE требует затрат времени на программирование.

С помощью точек доступа к услугам транспортного уровня (Transport Service Access Point, TSAP) протокол TCP допускает несколько соединений с одним IP-адресом (до 64K соединений). С помощью RFC 1006 TSAP однозначно идентифицируют эти соединения конечных коммуникационных пунктов с IP-адресом.

В разделе "Address Details [Подробности адреса]" диалогового окна Connection Parameters [Параметры соединения] вы определяете подлежащие использованию TSAP. TSAP в CPU вводится в поле "Local TSAP [Локальный TSAP]". TSAP, назначенный для соединения в партнерском CPU, вводится в поле "Partner TSAP [TSAP партнера]".



Параметр	Определение
General [Общие]	
End point: Partner [Конечный пункт: партнер]	Имя, назначенное CPU партнера (приемнику)
Interface [Интерфейс]	Имя, назначенное интерфейсам
Subnet [Подсеть]	Имя, назначенное подсетям
Адрес	Назначенные IP-адреса
Connection type [Тип соединения]	Тип протокола Ethernet
Connection ID [ID соединения]	Идентификационный номер
Connection data [Данные о соединении]	Адрес хранения данных локального и партнерского CPU
Active connection setup [Настройка активного соединения]	Селективная кнопка для выбора локального или партнерского CPU в качестве активного соединения
Подробности адреса	
TSAP[1] (ASCII)	TSAP локального и партнерского CPU в формате ASCII
TSAP ID	TSAP локального и партнерского CPU в шестнадцатеричном формате

[1] При конфигурировании соединения с CPU S7-1200 через ISO on TCP используйте в расширении TSAP для пассивных коммуникационных партнеров только символы ASCII.

Протокол управления передачей (Transport Control Protocol, TCP)

TCP – это стандартный протокол, описанный в RFC 793: Transmission Control Protocol. Основной целью TCP является предоставление услуг, обеспечивающих надежное и безопасное соединение между парами процессов. Этот протокол обладает следующими свойствами:

- Эффективный коммуникационный протокол, так как он тесно связан с аппаратным обеспечением
- Пригоден для объемов данных среднего и большого размера (до 8192 байтов)
- Предоставляет значительно больше услуг для приложений, в особенности:
  - Восстановление в случае ошибки
  - Управление потоком
  - Надежность
- Протокол, ориентированный на соединения
- Может быть очень гибко использован с системами других производителей, которые поддерживают только TCP
- Возможность маршрутизации
- Применимы только статические длины данных.
- Сообщения квитируются.
- Приложения адресуются с помощью номеров портов.
- Большинство протоколов пользовательских приложений, например, TELNET и FTP, используют TCP.
- Из-за наличия программного интерфейса SEND / RECEIVE требует затрат времени на программирование.

Параметр	Определение
General [Общие]	
End point: Partner [Конечный пункт: партнер]	Имя, назначенное CPU партнера (приемнику)
Interface [Интерфейс]	Имя, назначенное интерфейсам
Subnet [Подсеть]	Имя, назначенное подсетям
Address [Адрес]	Назначенные IP-адреса
Connection type [Тип соединения]	Тип протокола Ethernet
Connection ID [ID соединения]	Идентификационный номер
Connection data [Данные о соединении]	Адрес хранения данных локального и партнерского CPU
Active connection setup [Настройка активного соединения]	Селективная кнопка для выбора локального или партнерского CPU в качестве активного соединения
Address details [Подробности адреса]	
Порт (десятичный)	Локальный порт CPU в десятичном формате

7.4 Справочные данные

7.4.1 Получение адреса Ethernet (MAC-адреса) для CPU

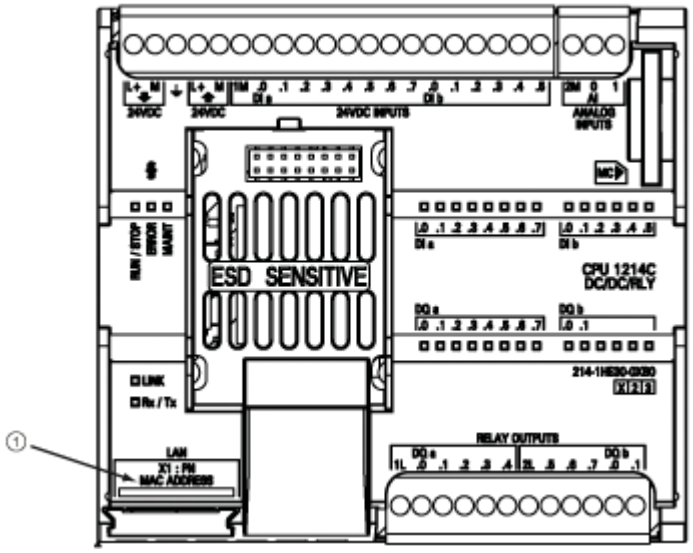
В сетях PROFINET MAC-адрес (Media Access Control address) является идентификатором, назначаемым производителем для идентификации. MAC-адрес обычно содержит зарегистрированный идентификационный номер производителя.

Стандартный (IEEE 802.3) формат для печати MAC-адресов в удобочитаемой форме – это шесть групп из двух шестнадцатеричных цифр, разделенных дефисами (-) или двоеточиями (:), в порядке передачи (например, 01-23-45-67-89-ab или 01:23:45:67:89:ab).

Указание

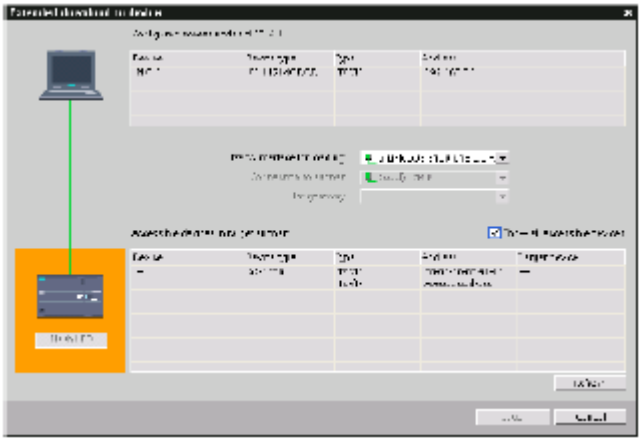
В каждый CPU на заводе загружается постоянный, уникальный MAC-адрес. Вы не можете изменить MAC-адрес CPU.

MAC-адрес напечатан спереди, в нижнем левом углу CPU. Чтобы иметь возможность прочитать MAC-адрес, нужно открыть нижние откидные крышки.



① MAC-адрес

Первоначально CPU не имели IP-адреса, а только установленный на заводе MAC-адрес. Обмен данными через PROFINET требует, чтобы всем устройствам был назначен уникальный IP-адрес.



Используйте функцию CPU "Download to device [Загрузить в устройство]" и диалоговое окно "Extended download to device [Расширенная загрузка в устройство]", чтобы показать все доступные сетевые устройства и обеспечить, чтобы всем им были назначены уникальные IP-адреса. Это диалоговое окно отображает все доступные и имеющиеся устройства с назначенными им MAC и IP-адресами. MAC-адреса особенно важны для идентификации устройств, не имеющих необходимого уникального IP-адреса.

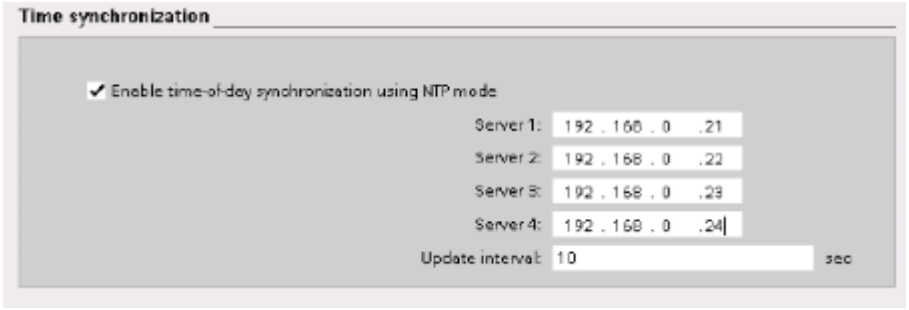
7.4.2 Конфигурирование синхронизирующего сетевого протокола (NTP)

Синхронизирующий сетевой протокол (Network Time Protocol, NTP) широко используется для синхронизации часов компьютерных систем с серверами точного текущего времени в сети Интернет. Обычно он обеспечивает точность менее одной миллисекунды в ЛВС и до нескольких миллисекунд в глобальных сетях. Обычные конфигурации NTP используют несколько резервных серверов и различные пути в сетях, чтобы обеспечить высокую точность и надежность.

Подсеть NTP работает с иерархией уровней, в которой каждому уровню присвоен номер, называемый слоем (стратой). Серверы слоя 1 (первичные) на самом нижнем уровне непосредственно синхронизируются с национальными службами времени. Серверы слоя 2 (вторичные) на следующем, боле высоком, уровне синхронизируются с серверами слоя 1 и так далее.

Параметры для синхронизации времени

В окне Properties [Свойства] выберите пункт "Time synchronization [Синхронизация времени]". Портал TIA отображает диалоговое окно Time synchronization:



Указание

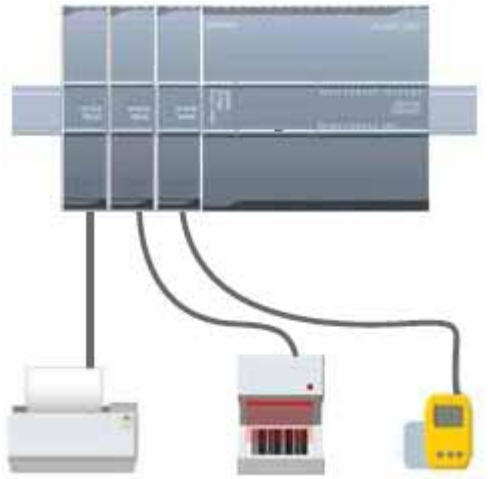
Все IP-адреса конфигурируются, когда вы загружаете проект.

Следующая таблица определяет параметры синхронизации времени:

Параметр	Определение
Enable time-of-day synchronization using NTP mode [Разблокировать синхронизацию времени, используя режим NTP]	Щелкните на триггерной кнопке, чтобы разблокировать синхронизацию времени с помощью серверов NTP.
Server 1	Назначенный IP-адрес для сервера сетевого времени 1
Server 2	Назначенный IP-адрес для сервера сетевого времени 2
Server 3	Назначенный IP-адрес для сервера сетевого времени 3
Server 4	Назначенный IP-адрес для сервера сетевого времени 4
Time synchronization interval [Интервал синхронизации времени]	Значение интервала (сек)

8 Двухточечная связь (PtP)

CPU поддерживает протокол двухточечной связи (PtP) для последовательного обмена данными на основе символов, в котором пользовательское приложение полностью определяет и реализует предпочтительный протокол. PtP предоставляет максимум свободы и гибкости, но требует больших затрат на реализацию в программе пользователя.



PtP предлагает большое разнообразие возможностей:

- Прямая передача данных на внешнее устройство, например, принтер
- Прием данных от других устройств, например, считывателей штрих-кода, считывателей устройств высокочастотной идентификации (RFID), видеокамер и систем технического зрения других фирм и многих других типов устройств
- Обмен информацией, передача и прием данных с помощью других устройств, например, устройств GPS, видеокамер или систем технического зрения других фирм, радиомодемов и многого другого

PtP-связь обеспечивает последовательный обмен данными с помощью стандартных универсальных асинхронных приемопередатчиков (UART), которые поддерживают различные скорости передачи и контроль четности. Коммуникационный модуль RS232 или RS485 (CM) предоставляет электрический интерфейс для осуществления PtP-связи.

STEP 7 Basic предоставляет библиотеки команд, которые вы можете использовать при программировании своего приложения. Эти библиотеки содержат функции PtP-связи для следующих протоколов:

- протокол USS для приводов
- протокол master-устройства Modbus RTU
- протокол slave-устройства Modbus RTU

8.1 Использование коммуникационных модулей RS232 и RS485

Два коммуникационных модуля (CM) обеспечивают интерфейс для PtP-связи: CM 1241 RS485 (стр. 369) и CM 1241 RS232 (стр. 370). Вы можете подключить до трех CM (любого типа). Устанавливайте CM слева от CPU или другого CM. Дальнейшую информацию об установке и снятии модулей вы найдете в главе "Монтаж" (стр. 30).

Коммуникационные модули RS232 и RS485 имеют следующие характеристики:

- Порт с потенциальной развязкой
- Поддержка протоколов двухточечной связи
- Конфигурирование и программирование с помощью расширенного набора команд и библиотечных функций
- Индикация действий по передаче и приему с помощью светодиодов
- Диагностический светодиод
- Питание через CPU. Нет необходимости во внешнем питании.

Дальнейшую информацию вы найдете под заголовком Технические данные коммуникационных модулей (стр. 369).

8.2 Конфигурирование коммуникационных портов

Коммуникационные модули могут быть сконфигурированы двумя способами:

- Сконфигурируйте в конфигурации устройств STEP 7 Basic параметры порта (скорость передачи и контроль четности), параметры передачи и параметры приема. Настройки конфигурации устройств постоянно хранятся в CPU. Эти настройки становятся действительными после выключения и последующего включения питания или после перехода из RUN в STOP.
- Для установки параметров используйте команды PORT_CFG, SEND_CFG и RCV_CFG. Настройки порта, установленные этими командами, действительны, пока CPU находится в режиме RUN. После перехода в состояние STOP или выключения и последующего включения питания настройки порта возвращаются к значениям, установленным в конфигурации устройств.

После конфигурирования аппаратуры (стр. 77) выполните параметризацию коммуникационных интерфейсов, выбрав один из CM в своей стойке.

Вкладка "Properties [Свойства]" окна просмотра параметров отображает параметры выбранного CM. Выберите "Port configuration [Конфигурация порта]" для редактирования следующих параметров:

- Скорость передачи
- Контроль четности
- Число стоповых битов
- Управление потоком (только RS232)
- Время ожидания

Кроме управления потоком, параметры конфигурации порта одинаковы, независимо от того, конфигурируете вы коммуникационный модуль RS232 или RS485. Значения параметров могут быть различными.

Порт может быть также сконфигурирован (или существующая конфигурация может быть изменена) из программы пользователя с помощью команды PORT_CFG (стр. 294).

Указание

Значения параметров, установленные командой PORT_CFG в программе пользователя, заменяют настройки конфигурации порта, установленные из STEP 7 Basic. Обратите внимание, что S7-1200 не сохраняет параметры, установленные командой PORT_CFG, в случае выключения питания.

Скорость передачи: По умолчанию скорость передачи составляет 9,6 Кбит в секунду. Допустимыми значениями являются:

300 Бод	2,4 Кбит/с	19,2 Кбит/с	76,8 Кбит/с

600 Бод	4,8 Кбит/с	28,4 Кбит/с	115,2 Кбит/с
1,2 Кбит/с	9,6 Кбит/с	57,6 Кбит/с	

Контроль четности: По умолчанию контроль четности отсутствует. Допустимыми значениями являются:

- No parity [Контроль четности отсутствует]
- Even [Проверка на четность]
- Odd [Проверка на нечетность]
- Mark (бит контроля четности всегда установлен в 1)
- Space (бит контроля четности всегда установлен в 0)

Число стоповых битов: Число стоповых битов может быть равно одному или двум. По умолчанию один.

Управление потоком: Для коммуникационного модуля RS232 вы можете выбрать аппаратное или программное управление потоком, как это описано в разделе "Управление потоками" (стр. 282). Если выбрано аппаратное управление потоком, то вы можете выбрать, будет ли сигнал RTS всегда включен или RTS включается. Если выбрано программное управление потоком, то вы можете определить символы ASCII для XON и XOFF.

Коммуникационный модуль RS485 не поддерживает управления потоком.

Время ожидания: Время ожидания определяет время, в течение которого коммуникационный модуль ожидает приема сигнала CTS после RTS, или приема XON после приема XOFF, в зависимости от типа управления потоком. Если время ожидания истекает до того, как коммуникационный модуль примет ожидаемый CTS или XON, то коммуникационный модуль прерывает операцию передачи и возвращает ошибку в программу пользователя. Время ожидания задается в миллисекундах в диапазоне от 0 до 65535 миллисекунд.

8.3 Управление потоками

Управление потоками – это механизм выравнивания потоков передаваемых и принимаемых данных, чтобы данные не терялись. Управление потоками гарантирует, что передающее устройство не передаст больше информации, чем сможет обработать принимающее устройство. Управление потоками может производиться аппаратно или программно. CM RS232 поддерживает как аппаратное, так и программное управление потоками. CM RS485 не поддерживает управления потоками. Вы определяете тип управления потоками при конфигурировании порта (стр. 281) или командой PORT_CFG.

Аппаратное управление потоками действует через сигналы готовности к передаче (Request-to-send, RTS) и готовности к приему (Clear-to-send, CTS). У CM RS232 сигнал RTS подается с контакта 7, а сигнал CTS принимается через контакт 8. CM 1241 является терминальным оборудованием (Data Terminal Equipment, DTE), которое обеспечивает RTS в качестве выхода и контролирует CTS на входе.

Аппаратное управление потоками: включаемый сигнал RTS

Если разблокировано аппаратное управление потоком с включаемым сигналом RTS для CM RS232, то модуль активизирует сигнал RTS для отправления данных. Он контролирует сигнал CTS, чтобы определить, может ли принимающее устройство принять данные. Если сигнал CTS активен, то модуль может передавать данные, пока сигнал CTS остается активным. Если сигнал CTS становится неактивным, то передача должна остановиться.

Передача возобновляется, когда сигнал CTS снова становится активным. Если сигнал CTS не активизируется в течение сконфигурированного времени ожидания, то модуль прерывает передачу и возвращает ошибку в программу пользователя. Вы задаете время ожидания при конфигурировании порта (стр. 281).

Управление потоком с включаемым сигналом RTS полезно для устройств, которые требуют сигнала о том, что передача активна. Примером может служить радиомодем, который использует сигнал RTS в качестве "ключа" для активизации радиопередатчика. Управление потоком с включаемым сигналом RTS не функционирует со стандартными телефонными модемами. Для телефонных модемов используйте опцию "RTS always on [RTS всегда включен]".

Аппаратное управление потоками: RTS всегда включен

В режиме "RTS always on" CM 1241 устанавливает RTS в активное состояние по умолчанию. Устройство, например телефонный модем, контролирует сигнал RTS из CM и использует этот сигнал как сигнал готовности к приему. Модем производит передачу в CM, когда сигнал RTS активен, т.е. когда телефонный модем распознает активный сигнал CTS. Если RTS не активен, то телефонный модем ничего не передает в CM.

Чтобы позволить модему передавать данные в CM в любой момент времени, сконфигурируйте аппаратное управление потоком с опцией "RTS always on [RTS всегда включен]". Таким образом, CM все время поддерживает сигнал RTS в активном состоянии. CM не деактивизирует сигнал даже в том случае, если модуль не может принимать символы. Передающее устройство должно гарантировать, что оно не переполнит принимающий буфер CM.

Использование сигналов готовности терминала к передаче данных (Data Terminal Block Ready, DTR) и готовности модема (Data Set Ready, DSR)

CM устанавливает DTR в активное состояние для любого типа аппаратного управления потоками. Модуль производит передачу только тогда, когда сигнал DSR становится активным. Состояние DSR анализируется только в начале операции передачи. Если DSR становится неактивным после начала передачи, то передача не останавливается

Программное управление потоком

Программное управление потоком использует для управления потоком специальные символы в сообщениях. Этими символами являются символы ASCII, которые представляют XON и XOFF.

XOFF указывает, что передача должна остановиться. XON указывает, что передачу можно возобновить.

Когда передающее устройство принимает символ XOFF от принимающего устройства, оно останавливает передачу. Передача возобновляется, когда передающее устройство принимает символ XON. Если оно не принимает символ XON в течение времени ожидания, которое определяется в конфигурации порта (стр. 281), CM прерывает передачу и возвращает ошибку в программу пользователя.

Программное управление потоком требует полнодуплексной связи, так как принимающий партнер должен быть в состоянии передать XOFF передающему партнеру, когда осуществляется передача. Программное управление потоком возможно только с сообщениями, содержащими только символы ASCII. Двоичные протоколы не могут использовать программное управление потоком.

8.4 Конфигурирование параметров приема и передачи

Прежде чем ПЛК сможет принять участие в PtP-связи, вы должны сконфигурировать параметры для передачи и приема сообщений. Эти параметры определяют, как должна действовать связь при передаче сообщений в целевое устройство и приеме сообщений из целевого устройства.

Конфигурирование параметров передачи

При конфигурировании CM вы определяете, как интерфейс связи передает данные, задавая свойство "Transmit message configuration [Конфигурирование передаваемых сообщений]" для выбранного CM.

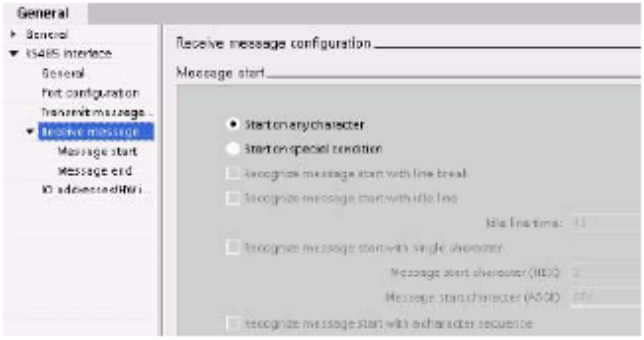
Вы можете также динамически конфигурировать или изменять параметры передачи сообщений из программы пользователя, используя команду SEND_CFG (стр. 296).

Указание

Значения параметров, установленные командой SEND_CFG в программе пользователя, заменяют настройки конфигурации порта. Обратите внимание, что CPU не сохраняет параметры, установленные командой SEND_CFG, в случае выключения питания.

Параметр	Определение
RTS On delay [Задержка включения RTS]	Определяет время ожидания после активизации RTS перед началом передачи. Диапазон времени составляет от 0 до 65535 мс, значение по умолчанию равно 0. Этот параметр действителен только тогда, когда в конфигурации порта (стр. 281) указано аппаратное управление потоком. CTS анализируется по истечении времени задержки включения RTS. Этот параметр действителен только для модулей RS232.
RTS Off delay [Задержка выключения RTS]	Определяет время ожидания перед деактивизацией RTS по окончании передачи. Диапазон времени составляет от 0 до 65535 мс, значение по умолчанию равно 0. Этот параметр действителен только тогда, когда в конфигурации порта (стр. 281) указано аппаратное управление потоком. Этот параметр действителен только для модулей RS232.
Send break at message start [Передать паузу в начале сообщения] Number of bit times in a break [Число тактов передачи в паузе]	Указывает, что в начале каждого сообщения передается пауза по истечении времени задержки включения RTS (если сконфигурировано) и при этом CTS активен. Вы определяете, сколько тактов содержит пауза, в течение которой линия удерживается в двоичном состоянии 0. По умолчанию 12, максимум 65535 до граничного значения 8 секунд.
Send idle line after a break [Передать информацию о простое линии после паузы] Idle line after a break [Простой линии после паузы]	Указывает, что после паузы в начале сообщения передается информация о простое линии. Параметр "Idle line after a break" определяет, сколько тактов длится простой линии, когда линия удерживается в состоянии 1. По умолчанию 12, максимум 65535 до граничного значения 8 секунд.

Конфигурирование параметров приема



В конфигурации устройств вы определяете, как интерфейс связи должен принимать данные и как он распознает начало и конец сообщения. Задайте эти параметры для выбранного CM в окне "Receive message configuration [Конфигурирование принимаемых сообщений]".

Вы можете также динамически конфигурировать или изменять параметры приема сообщений из программы пользователя, используя команду RCV_CFG (стр. 298).

Указание

Значения параметров, установленные командой RCV_CFG в программе пользователя, заменяют настройки конфигурации порта. Обратите внимание, что параметры, установленные командой RCV_CFG, не сохраняются в CPU в случае выключения питания.

Дальнейшую информацию вы найдете под заголовком Команда RCV_CFG.

Параметры начала сообщения

Вы можете определить, как коммуникационный модуль распознает начало сообщения. Начальные символы и символы, составляющие сообщение, входят в принимающий буфер до тех пор, пока не будет выполнено сконфигурированное условие конца сообщения.

Может быть задано несколько условий начала сообщения. Все эти условия должны быть выполнены, прежде чем будет распознано начало сообщения. Например, если вы сконфигурировали время простоя линии и определенный начальный символ, то СМ сначала будет ожидать выполнения требования о времени простоя линии, а затем заданного начального символа. Если будет принят какой-то другой символ (не заданный начальный символ), то СМ снова начнет поиск начала сообщения со времени простоя линии.

Порядок проверки стартовых условий:

- Простой линии
- Пауза на линии
- Символ или последовательности символов

При проверке нескольких стартовых условий, если одно из условий не выполнено, то СМ снова начнет проверку с первого необходимого условия.

Параметр	**Определение**
Start Character character [Символ для начала сообщения]	Это условие указывает, что сообщение начинается при успешном приеме определенного символа. Этот символ является первым символом в сообщении. Любой символ, принятый перед этим конкретным символом, будет проигнорирован.
Start on Any Character [Начало с любого символа]	Это условие указывает, что сообщение начинается при успешном приеме любого символа. Этот символ будет первым символом в сообщении.
Line Break [Пауза на линии]	Это условие указывает, что операция по приему сообщения должна начаться после приема символа паузы.
Idle Line [Простой линии]	Это условие указывает, что прием сообщения должен начаться после того, как принимающая линия в течение заданного числа тактов передачи находилась в простое. Как только это условие будет выполнено, начнется прием сообщения.
Special condition: Recognize message start with single [Особое условие: Распознавать начало сообщения по одному символу]	Указывает, что признаком начала сообщения является определенный символ. По умолчанию это символ начала текста STX.

Параметр	Определение
Special condition: Recognize message start with a character sequence [Особое условие: Распознавать начало сообщения по последовательности символов]	Указывает, что признаком начала сообщения является последовательность определенных символов. Для каждой последовательности можно определить до 5 символов. Для позиции каждого символа можно указать или конкретный шестнадцатеричный символ, или что этот символ игнорируется при сопоставлении последовательностей. Входящие последовательности сравниваются со сконфигурированными условиями начала сообщения, пока какое-либо условие не будет выполнено. Как только условие начала сообщения выполнено, начинается анализ условия конца сообщения. Вы можете сконфигурировать до 5 последовательностей символов, которые вы можете разблокировать и блокировать по мере необходимости. Условие начала сообщения выполнено, когда появляется одна из сконфигурированных последовательностей.
Пример параметризации	 В этой конфигурации условие начала сообщения выполнено, когда появляется одна из следующих комбинаций символов: • Когда принимается последовательность из пяти символов, в которой первым символом является 0x6A, а пятым 0x1C. В этой конфигурации в позициях 2, 3 и 4 могут находиться любые символы. После приема пятого символа начинается анализ условий конца сообщения. • Когда принимаются последовательно два символа 0x6A, которым предшествует любой символ. В этом случае анализ условий конца сообщения начинается после приема второго символа 0x6A (3 символов) Символ, предшествующий первому символу 0x6A, включается в условие начала сообщения. Примеры последовательностей, удовлетворяющих условиям начала сообщения: • <любой символ> 6A 6A • 6A 12 14 18 1C • 6A 44 A5 D2 1C

Параметры конца сообщения

Вы можете также определить, как интерфейс связи должен распознавать конец сообщения. Вы можете сконфигурировать несколько условий окончания сообщения. Если появляется одно из этих условий, сообщение заканчивается.

Несколько условий конца сообщения могут быть заданы одновременно. Сообщение оканчивается, когда любое из этих условий удовлетворяется. Например, вы можете задать условие конца сообщения временем ожидания конца сообщения в 300 миллисекунд, временем ожидания очередного символа в течение 40 тактов передачи и максимальной длиной 50 байт. Сообщение закончится, если его прием займет больше 300 миллисекунд, или если промежуток между двумя символами превышает 40 тактов передачи, или если принято 50 байтов.

Параметр	Определение
Recognize message end by message timeout [Распознавать конец сообщения по истечению времени сообщения]	Конец сообщения наступает, когда истекает сконфигурированный интервал времени ожидания конца сообщения. Отсчет времени ожидания начинается, когда принимается первый символ в соответствии с критерием начала сообщения. Значение по умолчанию составляет 200 мс, а диапазон от 0 до 65535 мс.
Recognize message end by response timeout [Распознавать конец сообщения по истечению времени ожидания ответа]	Конец сообщения наступает, когда сконфигурированный интервал времени ожидания ответа истекает, прежде чем будет получена действительная стартовая последовательность. Отсчет времени ожидания начинается, когда заканчивается передача. Значение по умолчанию составляет 200 мс, а диапазон от 0 до 65535 мс. Чтобы показать фактический конец сообщения, вы должны сконфигурировать другое условие окончания сообщения.
Recognize message end by inter-character gap [Распознавать конец сообщения по промежутку между символами]	Конец сообщения наступает, когда истекает максимальное сконфигурированное время ожидания следующего символа сообщения. Значение по умолчанию для интервала между символами составляет 12 тактов передачи, а максимальное значение равно 65535 тактам, но не более восьми секунд.
Recognize message end by max length [Распознавать конец сообщения по максимальной длине]	Конец сообщения наступает, когда принято сконфигурированное максимальное число символов. Значение по умолчанию составляет 0 байт, а максимальное значение 1024 байта.
Read message length from message [Считывать длину сообщения из самого сообщения]	Сообщение само указывает свою длину. Конец сообщения наступает, когда принято сообщение указанной длины. Способ задания и интерпретации длины сообщения описан ниже.
Recognize message end with a character [Распознавать конец сообщения по одному символу]	Конец сообщения наступает, когда принят определенный символ.

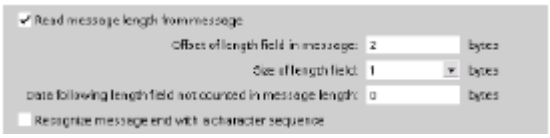
Параметр	Определение
Recognize message end with a character sequence [Распознавать конец сообщения по последовательности символов]	Конец сообщения наступает, когда принята определенная последовательность символов. Вы можете задать последовательность, содержащую до пяти символов. Для позиции каждого символа можно указать или конкретный шестнадцатеричный символ, или что этот символ игнорируется при сопоставлении последовательностей. Ведущие игнорируемые символы не являются частью условия конца сообщения. Завершающие игнорируемые символы являются частью условия конца сообщения.
Пример параметризации	 В этом случае условие конца сообщения удовлетворяется, когда последовательно принимаются два символа 0x7A, за которыми следуют два любых символа. Символ, предшествующий комбинации 0x7A 0x7A, не является частью последовательности конца сообщения. Два символа, следующие за комбинацией 0x7A 0x7A, необходимы для завершения последовательности конца сообщения. Символы в позициях 4 и 5 не имеют значения, но должны быть приняты, чтобы удовлетворить условие конца сообщения.

Указание длины сообщения в самом сообщении

При выборе особого условия, в котором длина сообщения включается в само сообщение, вы должны задать три параметра, которые определяют информацию о длине сообщения.

Фактическая структура сообщения зависит от используемого протокола. Этими тремя параметрами являются:

- n: позиция символа (база 1) в сообщении, с которого начинается указатель длины
- Length size [Размер указателя длины]: Число байтов (один, два или четыре) в указателе длины
- Length m [Длина m]: Число символов после указателя длины



Эти поля появляются в конфигурации приема сообщений в свойствах устройства.

Пример 1: Рассматривается сообщение, структурированное в соответствии со следующим протоколом:

STX	Len (n)	Символы с 3 по 14 учитываются при определении длины												
		ADR	PKE		INDEX		PWD		STW		HSW		BCC
1	2	3	4	5	6	7	8	9	10	11	12	13	14
STX	0x0C	xx	xxxx		xxxx		xxxx		xxxx		xxxx		xx

Параметры длины принимаемого сообщения конфигурируются следующим образом:

- n = 2 (Информация о длине сообщения начинается с байта 2)
- Размер указателя длины = 1 (Длина сообщения определена в одном байте)
- Длина m = 0 (Вслед за указателем длины нет дополнительных символов, которые не учитываются при определении длины. За указателем длины следуют 12 символов)

В этом примере символы с 3 по 14 включительно являются символами, которые учитываются при расчете длины Len (n).

Пример 2: Рассмотрим еще одно сообщение, структурированное в соответствии со следующим протоколом:

SD1	Len (n)	Len (n)	SD2	Символы с 5 по 10 учитываются при определении длины						FCS	ED
				DA	SA	FA	Блок данных=3 байта				
1	2	3	4	5	6	7	8	9	10	11	12
xx	0x06	0x06	xx	xx	xx	xx	xx	xx	xx	xx	xx

Параметры длины принимаемого сообщения конфигурируются следующим образом:

- n = 3 (Информация о длине сообщения начинается с байта 3)
- Значение длины = 1 (Длина сообщения определена в одном байте)
- Длина m = 3 (Вслед за указателем длины имеются три символа, которые не учитываются при определении длины. В протоколе этого примера символы SD2, FCS и ED не учитываются при расчете длины. Остальные шесть символов учитываются при расчете длины; таким образом, общее число символов после указателя длины равно девяти)

В этом примере символы с 5 по 10 включительно являются символами, которые учитываются при расчете длины Len (n).

8.5 Программирование обмена данными через PtP

STEP 7 Basic предоставляет расширенный набор команд, которые позволяют программе пользователя выполнять обмен данными через двухточечное соединение (Point-to-Point, PtP) с помощью протокола, сконструированного и заданного в программе пользователя. Эти команды могут быть разделены на две категории:

- Команды конфигурирования
- Коммуникационные команды

Команды конфигурирования

Прежде чем ваша пользовательская программа сможет начать обмен данными через PtP, вы должны сконфигурировать коммуникационный интерфейс и параметры для приема и передачи данных.

Вы можете выполнить конфигурирование интерфейса и сообщений для каждого коммуникационного модуля в конфигурации устройств или с помощью этих команд в вашей пользовательской программе:

- PORT_CFG
- SEND_CFG
- RCV_CFG

Коммуникационные команды

Коммуникационные команды PtP позволяют программе пользователя посылать сообщения в коммуникационные модули и принимать сообщения из этих модулей. Дальнейшую информацию о передаче данных с помощью этих команд вы найдете в разделе о согласованности данных (стр. 96).

Все функции PtP работают асинхронно. Программа пользователя может использовать архитектуру опроса для определения состояния операций передачи и приема. Команды SEND_PTP и RCV_PTP могут исполняться одновременно. Коммуникационные модули буферизуют передаваемые и принимаемые сообщения по мере необходимости до максимального размера буфера 1024 байта.

Коммуникационные модули передают сообщения устройствам, участвующим в обмене данными через PtP, и принимают сообщения от них. Протокол сообщений находится в буфере, который принимается из определенного коммуникационного порта или передается в этот порт.	• SEND_PTP • RCV_PTP
Дополнительные команды предоставляют возможность сброса приемного буфера, а также получения и установки определенных сигналов RS232.	• RCV_RST • SGN_GET • SGN_SET

8.5.1 Архитектура опроса

Команды двухточечного обмена данными в S7-1200 должны вызываться циклически /периодически для контроля принимаемых сообщений. Опрос процесса передачи сообщает программе пользователя, когда передача заканчивается.

Архитектура опроса: master-устройство

Типичной для master-устройства является следующая последовательность:

1. Команда SEND_PTP инициирует передачу коммуникационному модулю.
2. Команда SEND_PTP выполняется в следующих друг за другом циклах, опрашивая состояние процесса передачи.
3. Когда команда SEND_PTP показывает, что передача завершена, пользовательский код может готовиться к приему ответа.
4. Команда RCV_PTP выполняется многократно для контроля ответа. После получения в CM ответного сообщение команда RCV_PTP копирует ответ в CPU и сообщает, что получены новые данные.
5. Программа пользователя может обрабатывать ответ.
6. Перейдите у шагу 1 и повторите цикл.

Архитектура опроса: slave-устройство

Типичной для slave-устройства является следующая последовательность:

1. Программа пользователя должна выполнять команду RCV_PTP в каждом цикле.
2. После того как CM получил запрос, команда RCV_PTP показывает, что новые данные готовы, и запрос будет скопирован в CPU.
3. Программа пользователя должна обслужить запрос и сгенерировать ответ.
4. Для передачи ответ обратно master-устройству используется команда SEND_PTP.
5. Повторите команду SEND_PTP несколько раз, чтобы убедиться, что передача происходит.
6. Перейдите у шагу 1 и повторите цикл.

Slave-устройство должно обеспечить достаточно частый вызов команды RCV_PTP, чтобы получить передачу от master-устройства, прежде чем у него истечет время ожидания ответа. Для выполнения этой задачи программа пользователя может вызывать RCV_PTP из циклического OB, где время цикла достаточно для получения передачи от master-устройства, прежде чем истечет время ожидания. Если вы установите время цикла для OB так, чтобы на интервале времени ожидания master-устройства команда выполнилась дважды, то будет гарантировано, что программа пользователя примет все передачи без потерь.

8.6 Команды для двухточечного соединения

8.6.1 Общие параметры команд для двухточечного соединения

Поведение светодиодов на коммуникационном модуле

На коммуникационном модуле (CM) имеется три светодиодных индикатора:

- Диагностический светодиод: Этот светодиод мигает красным светом, пока к нему не обращается CPU. После запуска CPU проверяет модули и обращается к CM. Диагностический светодиод начинает мигать зеленым светом. Это значит, что CPU обратился к CM, но еще не обеспечил его конфигурацией. Конфигурация загружается в модуль, когда программа загружается в CPU. После загрузки в CPU диагностический светодиод на коммуникационном модуле должен постоянно гореть зеленым светом.
- Светодиод передачи: Этот светодиод находится над светодиодом приема. Светодиод передачи горит, когда данные передаются из коммуникационного порта.
- Светодиод приема: Это светодиод горит, когда данные принимаются коммуникационным портом.

Разрешающая способность тактов передачи

Для некоторых параметров задается количество тактов передачи при сконфигурированной скорости передачи. Задание параметра в тактах передачи делает этот параметр независимым от скорости передачи. Максимальное значение для всех параметров, задаваемых в тактах передачи, равно 65535. Однако максимальное время, которое может измерить S7-1200, равно 8 секундам.

Входной параметр REQ

Многие из команд PtP используют вход REQ, чтобы инициировать исполнение команды при нарастающем фронте. Вход REQ должен быть равен 1 (ИСТИНА) в течение одного исполнения команды, но он может продолжать принимать значение ИСТИНА так долго, как это необходимо. Команда не начнет другую операцию, пока она не будет вызвана с входом REQ, имеющим значение ЛОЖЬ, так что команда может сбросить предыдущее состояние входа REQ. Это необходимо, чтобы команда могла обнаружить нарастающий фронт для инициирования следующей операции.

Когда вы вставляете в программу команду PtP, вы получаете напоминание о необходимости задать экземплярный DB. Используйте уникальный DB для каждого типа команды PtP. Это значит, что все команды SEND_PTP для данного порта должны иметь один и тот же экземплярный DB, но SEND_PTP и RCV_PTP должны иметь разные экземплярные DB. Этот гарантирует, что входы, например, REQ, будут надлежащим образом обработаны каждой командой.

Входной параметр PORT

Выберите из ниспадающего меню (связанного с входом PORT) идентификатор порта CM для обработки этого экземпляра команды. Этот номер вы найдете также как "hardware identifier [идентификатор аппаратуры]" в информации о конфигурации для CM.

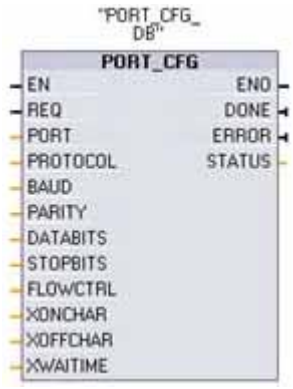
Выходные параметры NDR, DONE, ERROR, и STATUS

- Выход DONE указывает, что запрошенная операция выполнена без ошибок. Этот выход устанавливается на время одного цикла.
- Выход NDR (готовы новые данные) указывает, что запрошенное действие завершено без ошибок и получены новые данные. Этот выход устанавливается на время одного цикла.
- Выход ERROR указывает, что запрошенное действие завершено с ошибкой. Этот выход устанавливается на время одного цикла.
- Выход STATUS используется для сообщения об ошибках или промежуточных результатах.
  - Если устанавливается бит DONE или NDR, то STATUS устанавливается в 0 или получает значение информационного кода.
  - Если бит ERROR устанавливается, то STATUS получает значение кода ошибки.
  - Если ни один из упомянутых битов не устанавливается, то команда возвращает информацию, описывающую текущее состояние функции, например, состояние "занято".

Общие коды условий

STATUS (W#16#....)	Описание
0000	Нет ошибки
8x3A	Недопустимый указатель в параметре x
8070	Вся внутренняя память экземпляра используется
8080	Недопустимый номер порта
8081	Превышено время ожидания, ошибка модуля или другая внутренняя ошибка
8082	Параметризация не удалась, так как она происходит в фоновом режиме
8083	Переполнение буфера: СМ вернул принятое сообщение с длиной, превышающей допустимый параметр длины.
8090	Неправильная длина сообщения, неправильный субмодуль или недопустимое сообщение
8091	Неправильная версия в параметризующем сообщении
8092	Неправильная длина записи в параметризующем сообщении

8.6.2 Команда PORT_CFG



PORT_CFG (конфигурация порта) дает вам возможность изменять параметры порта, например, скорость передачи, из своей программы.

Вы можете установить начальную статическую конфигурацию порта в свойствах конфигурации устройств или просто использовать значения по умолчанию. Вы можете исполнить команду PORT_CFG в вашей программе для изменения конфигурации. Изменения конфигурации, выполненные с помощью команды PORT_CFG, не сохраняются постоянно в CPU. Параметры, установленные в конфигурации устройств, восстанавливаются, когда CPU переходит из RUN в STOP, а также после выключения и последующего включения питания.

Дальнейшую информацию вы найдете под заголовками Конфигурирование коммуникационных портов (стр. 281) и Управление потоками (стр. 282).

Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Активизировать изменение конфигурации при нарастающем фронте на этом входе.
PORT	IN	PORT	Идентификатор коммуникационного порта: Этот логический адрес является константой, на которую может быть сделана ссылка во вкладке "Constants" стандартной таблицы переменных.
PROTOCOL	IN	UInt	0 - протокол двухточечной связи 1..n - будущие определения конкретных протоколов

Параметр	Тип параметра	Тип данных	Описание
BAUD	IN	UInt	Скорость передачи порта: 1 - 300 Бод 2 - 600 Бод 3 - 1200 Бод 4 - 2400 Бод 5 - 4800 Бод 6 - 9600 Бод 7 - 19200 Бод 8 - 38400 Бод 9 - 57600 Бод 10 - 76800 Бод 11 - 115200 Бод
PARITY	IN	UInt	Контроль четности порта: 1 - Нет контроля четности 2 - контроль на четность 3 - контроль на нечетность 4 - контроль по единичному биту четности 5 - контроль по нулевому биту четности
DATABITS	IN	UInt	Число битов на символ: биты данных 1 - 8 биты данных 2 - 7
STOPBITS	IN	UInt	Стоповые биты: 1 - 1 стоповый бит 2 - 2 стоповых бита
FLOWCTRL	IN	UInt	Управление потоком: 1 - Нет управление потоком 2 - XON/XOFF 3 – Аппаратное, RTS всегда включен 4 - Аппаратное, RTS включается
XONCHAR	IN	Char	Укажите символ, который используется в качестве символа XON. Обычно это символ DC1 (11H). Этот параметр анализируется только в том случае, если разблокировано управление потоком.
XOFFCHAR	IN	Char	Укажите символ, который используется в качестве символа XOFF. Обычно это символ DC3 (13H). Этот параметр анализируется только в том случае, если разблокировано управление потоком.
XWAITIME	IN	UInt	Укажите, как долго следует ждать символа XON после получения символа XOFF, или как долго следует ждать сигнала CTS после активизации RTC (от 0 до 65535 мс). Этот параметр анализируется только в том случае, если разблокировано управление потоком.
DONE	OUT	Bool	ИСТИНА в течение одного цикла, после того как последний запрос был выполнен без ошибок
ERROR	OUT	Bool	ИСТИНА в течение одного цикла, после того как последний запрос был выполнен с ошибкой
STATUS	OUT	Word	Код условия выполнения

STATUS (W#16#....)	Описание
80A0	Указанный протокол не существует.
80A1	Указанная скорость передачи не существует.
80A2	Указанный вариант контроля четности не существует.
80A3	Указанное число битов данных не существует.
80A4	Указанное число стоповых битов не существует.
80A5	Указанный тип управления потоком не существует.
80A6	Время ожидания равно 0 и управление потоком разблокировано
80A7	XON и XOFF являются недопустимыми значениями

8.6.3 Команда SEND_CFG

Команда SEND_CFG (конфигурирование передачи) делает возможным динамическое конфигурирование параметров последовательной передачи для порта двухточечной связи. Все сообщения, стоящие в очереди в коммуникационном модуле (CM), будут отвергнуты, как только будет выполнена команда SEND_CFG.

Вы можете установить начальную статическую конфигурацию порта в свойствах конфигурации устройств или просто использовать значения по умолчанию. Вы можете исполнить команду SEND_CFG в вашей программе для изменения конфигурации. Изменения конфигурации, выполненные с помощью команды SEND_CFG, не сохраняются постоянно в ПЛК. Параметры, установленные в конфигурации устройств, восстанавливаются, когда ПЛК переходит из RUN в STOP, а также после выключения и последующего включения питания.

См. Конфигурирование параметров приема и передачи (стр. 284).

Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Активизировать изменение конфигурации при нарастающем фронте на этом входе
PORT	IN	PORT	Идентификатор коммуникационного порта: Этот логический адрес является константой, на которую может быть сделана ссылка во вкладке "Constants" стандартной таблицы переменных.
RTSONDLY	IN	UInt	Количество миллисекунд ожидания после активизации RTS, прежде чем произойдет передача Tx-данных. Этот параметр действителен только в том случае, если разблокировано аппаратное управление потоком. От 0 до 65535 мс. 0 блокирует это свойство.

Параметр	Тип параметра	Тип данных	Описание
RTSOFFDLY	IN	UInt	Количество миллисекунд ожидания после передачи Tx-данных, прежде чем RTS будет деактивизирован: Этот параметр действителен только в том случае, если разблокировано аппаратное управление потоком. От 0 до 65535 мс. 0 блокирует это свойство.
BREAK	IN	UInt	Этот параметр указывает, что в начале каждого сообщения будет передана пауза в течение указанного числа тактов передачи. Максимальное число тактов равно 65535. 0 блокирует это свойство. Максимум 8 секунд
IDLELINE	IN	UInt	Этот параметр указывает, что перед началом каждого сообщения линия будет находиться в состоянии простоя в течение указанного числа тактов передачи. Максимальное число тактов равно 65535. 0 блокирует это свойство. Максимум 8 секунд
DONE	OUT	Bool	ИСТИНА в течение одного цикла, после того как последний запрос был выполнен без ошибок
ERROR	OUT	Bool	ИСТИНА в течение одного цикла, после того как последний запрос был выполнен с ошибкой
STATUS	OUT	Word	Код условия выполнения

STATUS (W#16#....)	Описание
80B0	Конфигурация прерывания передачи недопустима
80B1	Время паузы превышает допустимое значение (2500 тактов передачи)
80B2	Время простоя превышает допустимое значение (2500 тактов передачи)

8.6.4 Команда RCV_CFG

Команда RCV_CFG (конфигурирование приема) выполняет динамическое конфигурирование параметров последовательного приема для порта двухточечной связи. Эта команда конфигурирует условия, которые указывают на начало и конец принимаемого сообщения. Все сообщения, стоящие в очереди в коммуникационном модуле (CM), отвергаются, когда выполняется команда RCV_CFG.

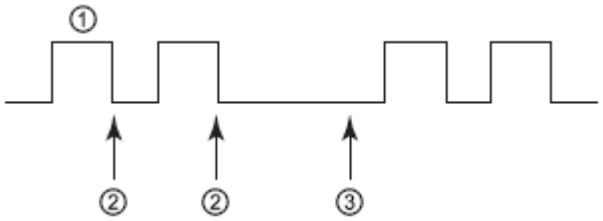
Вы можете установить начальную статическую конфигурацию порта в свойствах конфигурации устройств или просто использовать значения по умолчанию. Вы можете исполнить команду RCV_CFG в вашей программе для изменения конфигурации. Изменения конфигурации, выполненные с помощью команды RCV_CFG, не сохраняются постоянно в ПЛК. Параметры, установленные в конфигурации устройств, восстанавливаются, когда ПЛК переходит из RUN в STOP, а также после выключения и последующего включения питания. Дальнейшую информацию вы найдете под заголовком "Конфигурирование параметров приема" (стр. 285).

Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Активизировать изменение конфигурации при нарастающем фронте на этом входе
PORT	IN	PORT	Идентификатор коммуникационного порта: Этот логический адрес является константой, на которую может быть сделана ссылка во вкладке "Constants" стандартной таблицы переменных.
CONDITIONS	IN	CONDITIONS	Структура данных этого параметра определяет условия начала и конца сообщения. Они описаны ниже.
DONE	OUT	Bool	ИСТИНА в течение одного цикла, после того как последний запрос был выполнен без ошибок
ERROR	OUT	Bool	ИСТИНА в течение одного цикла, после того как последний запрос был выполнен с ошибкой
STATUS	OUT	Word	Код условия выполнения

Условия начала сообщения для команды RCV_PTP

Команда RCV_PTP использует конфигурацию, заданную командой RCV_CFG, для определения начала и конца сообщений при двухточечном соединении. Начало сообщения определяется стартовыми условиями. Начало сообщения может быть определено одним или комбинацией нескольких стартовых условий. Если задано больше одного стартового условия, то все эти условия должны быть выполнены перед началом сообщения. Возможные стартовые условия:

- "Начальный символ" указывает, что сообщение начинается при успешном приеме определенного символа. Этот символ будет первым символом в сообщении. Любой символ, принятый до этого конкретного символа, будет отвергнут.
- "Любой символ" указывает, что любой успешно принятый символ будет началом сообщения. Этот символ будет первым символом в сообщении.
- "Пауза на линии" указывает, что операция по приему сообщения должна начаться после приема символа паузы.
- "Простаивающая линия" указывает, что прием сообщения должен начаться после того, как принимающая линия пробудет в состоянии покоя в течение заданного количества тактов передачи. Как только это условие выполняется, начинается передача сообщения.



① Символы

② Новый запуск таймера простоя линии

③ Простой линии обнаружен, и начат прием сообщения

- "Переменные последовательности символов": Стартовые условия могут быть построены на переменном количестве последовательностей символов (максимум до 4), состоящих из переменного числа символов (максимум до 5). Позиция каждого символа в каждой последовательности может быть выбрана в качестве определенного символа, или в качестве безразличного символа ("джокера"), на месте которого может стоять любой символ. Эти стартовые условия могут использоваться, когда начало сообщения указывают несколько различных последовательностей символов.

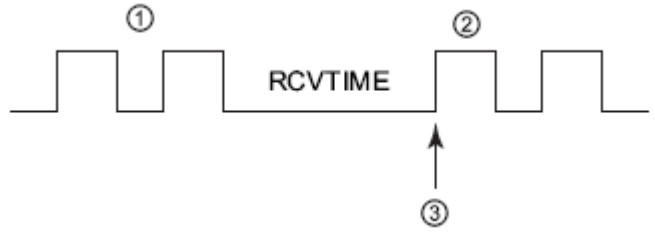
  Рассмотрим следующее, принятое в шестнадцатеричном коде сообщение: "**68** 10 aa **68** bb 10 aa 16" и сконфигурированные стартовые последовательности, показанные в следующей таблице. Стартовые последовательности начинают анализироваться, когда успешно принят первый символ 68H. При успешном приеме четвертого символа (второй символ 68H) выполняется стартовое условие 1. Как только стартовые условия выполнены, начинается анализ конечных условий. Обработка стартовой последовательности может быть завершена из-за различных ошибок при контроле четности, ошибок кадрирования или ошибок интервала времени между символами. Эти ошибки приводят к тому, что сообщение не принимается, так как стартовое условие не было выполнено.

Стартовое условие	Первый символ	Первый символ +1	Первый символ +2	Первый символ +3	Первый символ +4
1	**68**H	xx	xx	**68**H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

Условия окончания сообщения для команды RCV_PTP

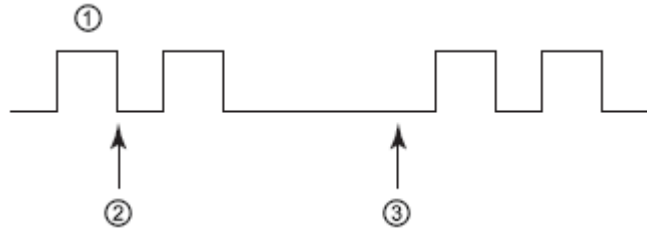
Конец сообщения определяется указанием конечных условий. Конец сообщения определяется первым появлением одного или нескольких сконфигурированных конечных условий. Возможные условия конца сообщения:

- "Истечение времени ожидания ответа" указывает, что символ ответа должен быть успешно принят в течение времени, заданного параметром RCVTIME. Таймер начинает работать, как только передача успешно завершается, и модуль начинает операцию приема. Если символ не принимается в течение интервала, задаваемого параметром RCVTIME, то в соответствующую команду RCV_PTP возвращается ошибка. Истечение времени ожидания ответа не определяет конкретного условия конца сообщения. Оно только указывает, что какой-нибудь символ должен быть успешно принят в течение заданного времени. Особое условие конца сообщения для определения конечного условия для ответных сообщений.



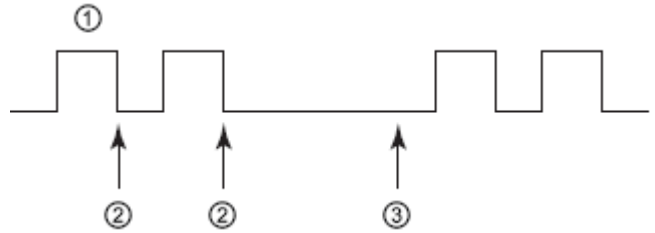
① Переданные символы
② Принятые символы
③ Первый символ должен быть успешно принят в течение этого времени

- "Истечение времени ожидания сообщения" указывает, что сообщение должно быть принято в течение времени, указанного в параметре MSGTIME. Таймер начинает работать, как только будет выполнено указанное стартовое условие.



① Принятые символы
② Условие начала сообщения выполнено: запускается таймер сообщения
③ Таймер сообщения завершает свою работу и завершает сообщение

- "Промежуток между символами" – это время, измеряемое от конца одного символа (последнего стопового бита) до конца следующего символа. Если время между любыми двумя символами превышает сконфигурированное количество тактов передачи, то сообщение завершается.



① Принятые символы
② Новый запуск таймера межсимвольного времени
③ Таймер межсимвольного времени завершает работу и завершает сообщение с ошибкой

- "Максимальная длина": Операция приема останавливается, как только принято заданное количество символов. Это условие может быть использовано для предотвращения ошибки переполнения буфера.

  Если это конечное условие комбинируется с условиями истечения времени ожидания и условие истечения времени ожидания выполняется, то все действительные принятые символы выводятся, даже если максимальная длина еще не достигнута. Это позволяет поддерживать протоколы различной длины, даже если известна только максимальная длина.

- Комбинированное условие "N + Размер указателя длины + Длина M". Это конечное условие может быть использовано для обработки сообщения переменной длины, содержащего поле длины.

  – "N" указывает позицию (число символов от начала сообщения), где начинается поле длины. (База 1)

  – " Размер указателя длины " указывает поле длины. Допустимыми значениями являются 1, 2 или 4 байта.

  – "Длина M" указывает количество завершающих символов (после поля длины), которые не включаются в длину сообщения. Это значение может быть использовано для указания длины поля контрольной суммы, размер которого не включается в поле длины

  – В качестве примера рассмотрим формат сообщения, состоящий из начального символа, адресного символа, однобайтного поля длины, данных сообщения, символов контрольной суммы и конечного символа. Записи под заголовком "Длина" соответствуют параметру N. Значение N равно 3 и указывает, что байтом длины является третий байт от начала сообщения. Размер указателя длины равен 1, указывая, что значение длины сообщения содержится в одном байте. Поля контрольной суммы и конечного символа соответствуют параметру "Длина M". Значение параметра "Длина M" равно 3, указывая число байтов в полях контрольной суммы и конечного символа.

Начальный символ	**Адрес**	**Длина (N)**	**Сообщение**		**Контрольная сумма и конечный символ Длина M**		
(1)	**(2)**	**(3)**	**... (x)**		**x+1 x+2 x+3**		
xx	xx	xx	xx	xx	xx	xx	xx

- Переменные символы: Это конечное условие может использоваться для завершения приема на основе различных последовательностей символов. Эти последовательности могут состоять из различного числа символов (не более 5). Позиция каждого символа в каждой последовательности может быть выбрана для записи определенного символа или безразличного символа ("джокера"), означающего, что условию удовлетворяет любой символ. Все ведущие символы, сконфигурированные так, чтобы они были проигнорированы, не являются необходимой частью сообщения. Все замыкающие символы, которые игнорируются, являются необходимой частью сообщения.

Структура типа данных параметра CONDITIONS, часть 1 (стартовые условия)

Параметр	Тип параметра	Тип данных	Описание
STARTCOND	IN	UInt	Задание стартовых условий: • 01H – Начальный символ • 02H – Любой символ • 04H - Пауза на линии • 08H – Простаивающая линия • 10H - Последовательность 1 • 20H - Последовательность 2 • 40H - Последовательность 3 • 80H - Последовательность 4
IDLETIME	IN	UInt	Число тактов передачи для времени ожидания простоя линии. Используется только вместе с условием "Простаивающая линия". От 0 до 65535
STARTCHAR	IN	Byte	Начальный символ для условия "Начальный символ".
STRSEQ1CTL	IN	Byte	Управление игнорированием/сравнением для каждого символа последовательности 1: Это разблокирующие биты для каждого символа в начальной последовательности • 01H - Символ 1 • 02H - Символ 2 • 04H - Символ 3 • 08H - Символ 4 • 10H - Символ 5 Блокирование бита для определенного символа означает, что в этой позиции последовательности подходит любой символ.
STRSEQ1	IN	Char[5]	Последовательность 1, начальные символы (5 символов)
STRSEQ2CTL	IN	Byte	Управление игнорированием/сравнением для каждого символа последовательности 2
STRSEQ2	IN	Char[5]	Последовательность 2, начальные символы (5 символов)
STRSEQ3CTL	IN	Byte	Управление игнорированием/сравнением для каждого символа последовательности 3
STRSEQ3	IN	Char[5]	Последовательность 3, начальные символы (5 символов)
STRSEQ4CTL	IN	Byte	Управление игнорированием/сравнением для каждого символа последовательности 4
STRSEQ4	IN	Char[5]	Последовательность 4, начальные символы (5 символов)

Структура типа данных параметра CONDITIONS, часть 2 (конечные условия)

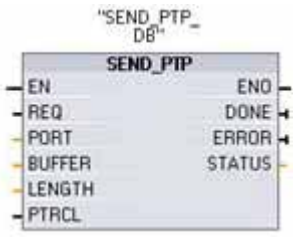
Параметр	Тип параметра	Тип данных	Описание
ENDCOND	IN	UInt	Этот параметр определяет условие окончания сообщения: • 01H – Время ответа • 02H – Время сообщения • 04H – Интервал между символами • 08H – Максимальная длина • 10H - N + Длина + M • 20H – Последовательность символов
MAXLEN	IN	UInt	Максимальная длина сообщения: Используется только в том случае, если в качестве конечного условия выбрана максимальная длина сообщения. От 0 до 1023 байт
N	IN	UInt	Позиция байта поля длины в сообщении. Используется только с конечным условием N + Длина + M. От 1 до 1023 байт
LENGTHSIZE	IN	UInt	Размер поля (1, 2 или 4 байта). Используется только с конечным условием N + Длина + M.
LENGTHM	IN	UInt	Определяет число символов после поля длины, которые не включены в значение поля длины. Используется только с конечным условием N + Длина + M. От 0 до 255 байт
RCVTIME	IN	UInt	Определяет, сколько времени необходимо ждать первого символа, подлежащего приему. Операция приема будет завершена с ошибкой, если символ не будет успешно принят в течение указанного времени. Этот параметр используется только с условием "Время ответа". От 0 до 65535 тактов передачи, не более 8 секунд Этот параметр в действительности не оценивается как условие конца сообщения, так как он анализирует только стартовые условия. Должно быть выбрано отдельное условие конца сообщения.
MSGTIME	IN	UInt	Определяет, сколько времени необходимо ждать полного приема всего сообщения после приема первого символа. Этот параметр используется только с условием "Время сообщения". От 0 до 65535 миллисекунд.
CHARGAP	IN	UInt	Определяет число тактов передачи между символами. Если число тактов передачи между символами превышает указанное значение, то конечное условие выполняется. Этот параметр используется только с условием "Интервал между символами". От 0 до 65535 миллисекунд.

Параметр	Тип параметра	Тип данных	Описание
ENDSEQ1CTL	IN	Byte	Управление игнорированием/сравнением для каждого символа последовательности 1: Это разблокирующие биты для каждого символа в конечной последовательности. Символ 1 – это бит 0, символ 2 – это бит 1, …, символ 5 – это бит 4. Блокирование бита для определенного символа означает, что в этой позиции последовательности подходит любой символ.
ENDSEQ1	IN	Char[5]	Последовательность 1, начальные символы (5 символов)

Коды условий

STATUS (W#16#....)	Описание
80C0	Выбрано недопустимое стартовое условие
80C1	Выбрано недопустимое конечное условие, конечное условие не выбрано
80C2	Разблокировано прерывание приема, и это невозможно
80C3	Разблокировано конечное условие "Максимальная длина", и максимальная длина равна 0 или > 1024
80C4	Разблокирована рассчитываемая длина, и N >= 1023
80C5	Разблокирована рассчитываемая длина, и длина не равна 1, 2 или 4
80C6	Разблокирована рассчитываемая длина, и значение M > 255
80C7	Разблокирована рассчитываемая длина, и рассчитываемая длина > 1024
80C8	Разблокировано время ожидания ответа, и время ожидания ответа равно нулю
80C9	Разблокировано время ожидания для интервала между символами, и оно равно нулю или > 2500
80CA	Разблокировано время ожидания простоя линии, и оно равно нулю или > 2500
80CB	Разблокирована конечная последовательность, но все ее символы "безразличны"
80CC	Разблокирована начальная последовательность (любая из 4), но все ее символы "безразличны"

8.6.5 Команда SEND_PTP



Команда SEND_PTP (передать данные через двухточечное соединение) инициирует передачу данных. SEND_PTP передает указанный буфер в CM. Программа CPU продолжает исполняться, пока CM передает данные с заданной скоростью передачи. В каждый данный момент времени в очереди может находиться только одна операция передачи. CM возвращает ошибку, если исполняется вторая команда SEND_PTP, в то время как CM уже передает сообщение.

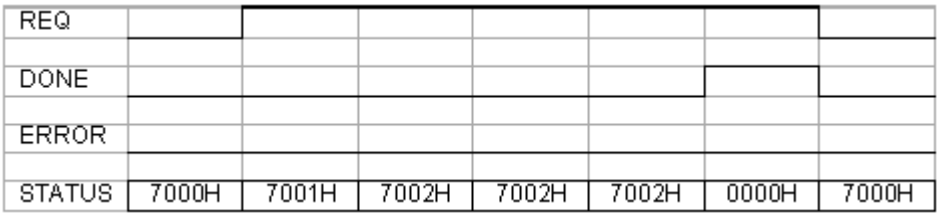
Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Активизирует запрошенную передачу при нарастающем фронте на этом входе, разблокирующем передачу. Это инициирует передачу содержимого буфера в коммуникационный модуль (CM) двухточечного соединения.
PORT	IN	PORT	Идентификатор коммуникационного порта: Этот логический адрес является константой, на которую может быть сделана ссылка во вкладке "Constants" стандартной таблицы переменных.
BUFFER	IN	Variant	Этот параметр указывает на начальный адрес буфера передачи. Булевы данные или булевы массивы не поддерживаются.
LENGTH	IN	UInt	Длина передаваемого кадра в байтах При передаче составной структуры всегда используйте длину 0.
PTRCL	IN	Bool	Этот параметр выбирает буфер для стандартной двухточечной связи или для специальных протоколов, поставляемых фирмой Siemens, которые реализованы в присоединенном CM. ЛОЖЬ = операции двухточечной связи, управляемые программой пользователя (единственная применимая опция)
DONE	OUT	Bool	ИСТИНА в течение одного цикла, после того как последний запрос был выполнен без ошибок
ERROR	OUT	Bool	ИСТИНА в течение одного цикла, после того как последний запрос был выполнен с ошибкой
STATUS	OUT	Word	Код условия выполнения

Пока операция передачи выполняется, выходы DONE и ERROR принимают значение ЛОЖЬ. Когда операция передачи завершена, выход DONE или выход ERROR принимает значение ИСТИНА в течение одного цикла, чтобы показать состояние операции передачи. Пока DONE или ERROR принимает значение ИСТИНА, выход STATUS действителен.

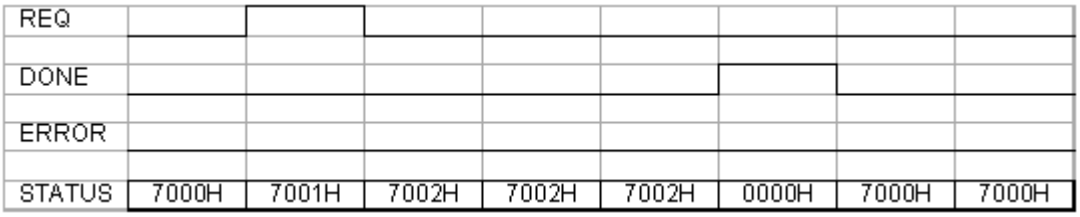
Команда возвращает состояние 16#7001, если коммуникационный модуль (CM) принимает передаваемые данные. Последующие исполнения команды SEND_PTP возвращают 16#7002, если CM все еще занят передачей. Когда операция передачи завершается, CM возвращает состояние 16#0000, если не было ошибок. Следующие исполнения команды SEND_PTP с REQ = 0 возвращают состояние 16#7000 (не занят).

Связи выходных значений с REQ:

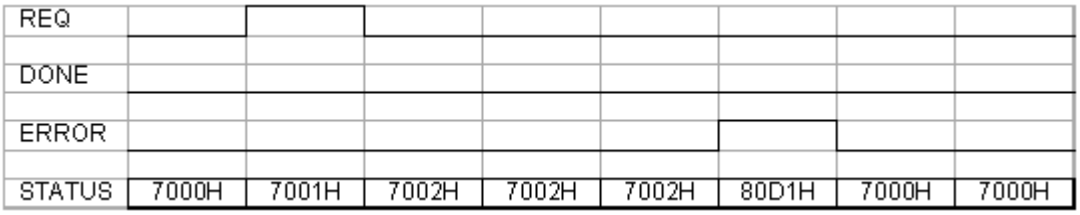
При этом предполагается, что команда вызывается периодически, чтобы проверить состояние процесса передачи. На следующем рисунке предполагается, что команда вызывается в каждом цикле (представленном значениями STATUS).



На следующем рисунке показано, как параметры DONE и STATUS оказываются действительными в течение только одного цикла, если к линии REQ прикладывается импульс (в течение одного цикла), чтобы инициировать операцию передачи.



На следующем рисунке показана связь параметров DONE, ERROR и STATUS в случае ошибки.



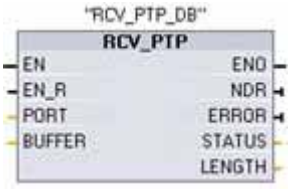
STATUS (W#16#....)	**Описание**
80D0	Новый запрос при активном передатчике
80D1	Передача прервана из-за отсутствия CTS в течение времени ожидания
80D2	Передача прервана из-за отсутствия сигнала готовности DSR от устройства передачи данных DCE
80D3	Передача прервана из-за переполнения очереди (передача более 1024 байтов)
7000	Не занят
7001	Занят при приеме запроса (первый вызов)
7002	Занят опросом (n-й вызов)

Взаимодействие параметров LENGTH и DATA для PTP_SEND

Минимальный размер данных, который может быть передан командой PTP_SEND, равен одному байту. Параметр DATA определяет размер данных, подлежащих передаче. Для этого параметра нельзя использовать данные типа BOOL или массивы типа BOOL.

Параметр LENGTH	Параметр DATA	Описание
LENGTH = 0	Не используется	Полные данные передаются так, как они определены в параметре DATA. Вам не нужно указывать количество передаваемых байтов, если LENGTH = 0.
LENGTH > 0	Элементарный тип данных	Значение LENGTH должно содержать число байтов этого типа данных. В противном случае ничего не передается и возвращается ошибка 8088H.
	Структура	Значение LENGTH может содержать число байтов, меньшее, чем полная длина структуры в байтах. В этом случае передаются только первые LENGTH байтов.
	Массив	Значение LENGTH может содержать число байтов, меньшее, чем полная длина массива в байтах. В этом случае передаются только элементы массива, которые полностью укладываются в байты LENGTH. Значение LENGTH должно быть кратным числу байтов элементов данных. В противном случае STATUS = 8088H, ERROR = 1, и передача не происходит.
	Строка	Передается полное распределение памяти формата строки. Значение LENGTH должно учитывать байты для максимальной длины, фактической длины и символов строки. Для типа данных STRING (строка) все длины и символы имеют размер по одному байту. Если параметр DATA использует строку символов в качестве фактического параметра, то значение LENGTH должно учитывать также два байта для двух полей длины.

8.6.6 Команда RCV_PTP



Команда RCV_PTP (прием данных двухточечного соединения) опрашивает сообщения, принятые в CM. Если сообщение имеется, то оно будет передано из CM в CPU. При ошибке выводится соответствующе значение параметра STATUS.

Значение параметра STATUS действительно, если NDR или ERROR принимает значение ИСТИНА. Значение параметра STATUS лает основание для завершения операции приема в CM. Обычно это положительное значение, указывающее, что операция приема была успешной и что процесс приема завершен нормально. Если значение STATUS отрицательно (устанавливается старший бит шестнадцатеричного значения), то это указывает, что операция приема была завершена из-за ошибки, например, контроля четности, кадрирования или переполнения.

Каждый модуль CM, используемый для двухточечной связи, имеет буфер максимальной емкостью до 1 Кбайта. Это может быть одно большое сообщение или несколько меньших сообщений.

Параметр	Тип параметра	Тип данных	Описание
EN_R	IN	Bool	Если этот вход принимает значение ИСТИНА, то модуль CM должен быть проверен на наличие принятых сообщений. Если сообщение было успешно принято, то оно будет передано из модуля в CPU. Если EN_R принимает значение ЛОЖЬ, то CM проверяется на наличие принятых сообщений и выход STATUS устанавливается, но сообщение не передается в CPU.
PORT	IN	PORT	Идентификатор коммуникационного порта: Этот логический адрес является константой, на которую можно ссылаться во вкладке "Constants" стандартной таблицы переменных.
BUFFER	IN	Variant	Этот параметр указывает на начальный адрес принимающего буфера. Этот буфер должен быть достаточно большим, чтобы принять сообщение максимальной длины. Булевы данные или булевы массивы не поддерживаются.
NDR	OUT	Bool	ИСТИНА в течение одного цикла, когда готовы новые данные и операция завершена без ошибок.
ERROR	OUT	Bool	ИСТИНА в течение одного цикла, если операция была завершена с ошибкой
STATUS	OUT	Word	Код условия выполнения
LENGTH	OUT	UInt	Длина возвращенного сообщения (в байтах)

STATUS (W#16#...)	Описание
0000	Отсутствует буфер
80E0	Сообщение завершено, так как приемный буфер полон
80E1	Сообщение завершено из-за ошибки контроля четности
80E2	Сообщение завершено из-за ошибки кадрирования

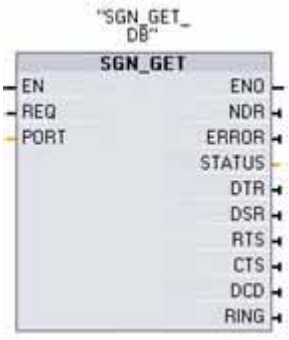
STATUS (W#16#...)	Описание
80E3	Сообщение завершено из-за ошибки переполнения
80E4	Сообщение завершено, так как расчетная длина превышает размер буфера
0094	Сообщение завершено, так как было принято максимальное число символов
0095	Сообщение завершено, из-за превышения времени приема сообщения
0096	Сообщение завершено, из-за превышения интервала между символами
0097	Сообщение завершено, из-за превышения времени ожидания ответа
0098	Сообщение завершено, так как условие длины "N+LEN+M" было выполнено
0099	Сообщение завершено, из-за того, что было выполнено условие окончания сообщения

8.6.7 Команда RCV_RST

Команда RCV_RST (сброс приемника) очищает приемный буфер в СМ.

Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Активизирует сброс при нарастающем фронте на этом разблокирующем входе
PORT	IN	PORT	Идентификатор коммуникационного порта: Порт должен быть указан, с помощью логического адреса модуля.
DONE	OUT	Bool	Если принимает значение ИСТИНА в течение одного цикла, то это указывает, что последний запрос был выполнен без ошибок.
ERROR	OUT	Bool	Если принимает значение ИСТИНА, то это показывает, что последний запрос был выполнен с ошибками. Кроме того, если этот выход принимает значение ИСТИНА, то выход STATUS будет содержать соответствующие коды ошибок.
STATUS	OUT	Word	Код ошибки

8.6.8 Команда SGN_GET



Команда SGN_GET (опрос сигналов RS232) считывает текущие состояния коммуникационных сигналов RS232. Эта функция действительна только для коммуникационного модуля RS232 (CM).

Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Получает состояния сигналов RS232 при нарастающем фронте на этом входе
PORT	IN	PORT	Идентификатор коммуникационного порта: Этот логический адрес является константой, на которую может быть сделана ссылка во вкладке "Constants" стандартной таблицы переменных.
NDR	OUT	Bool	ИСТИНА в течение одного цикла, когда новые данные готовы и операция завершена без ошибок
ERROR	OUT	Bool	ИСТИНА в течение одного цикла, если операция была завершена с ошибкой
STATUS	OUT	Word	Код условия выполнения
DTR	OUT	Bool	Терминал ввода данных готов, модуль готов (выход)
DSR	OUT	Bool	Набор данных готов, коммуникационный партнер готов (вход)
RTS	OUT	Bool	Запрос на передачу, модуль готов к передаче (выход)
CTS	OUT	Bool	Готовность к приему, коммуникационный партнер может принимать данные (вход)
DCD	OUT	Bool	Носитель данных распознан, уровень принимаемого сигнала (всегда 0, не поддерживается)
RING	OUT	Bool	Индикатор вызова, индикация поступающего вызова (всегда 0, не поддерживается)

STATUS (W#16#....)	Описание
80F0	CM является модулем RS485 и сигналы отсутствуют
80F1	Сигналы не могут быть установлены из-за аппаратного управления потоком
80F2	Сигнал готовности модема (DSR) не может быть установлен, так как модуль является терминальным оборудованием (DTE)
80F3	Терминал ввода данных (DTR) не может быть установлен, так как модуль является аппаратурой передачи данных (DCE)

8.6.9 Команда SGN_SET

Команда SGN_SET (установка состояний сигналов RS232) устанавливает состояния коммуникационных сигналов RS232. Эта функция действительна только для коммуникационного модуля RS232 (CM).

Параметр	Тип параметра	Тип данных	Описание
REQ	IN	Bool	Запускает операцию установки сигналов RS232 при нарастающем фронте на этом входе
PORT	IN	PORT	Идентификатор коммуникационного порта: Этот логический адрес является константой, на которую можно ссылаться во вкладке "Constants" стандартной таблицы переменных.
SIGNAL	IN	Byte	Выбирает сигналы, подлежащие установке: (допустимо несколько сигналов) • 01H = Установить RTS • 02H = Установить DTR • 04H = Установить DSR
RTS	IN	Bool	Запрос на передачу, модуль готов передавать значение, подлежащее установке (ИСТИНА или ЛОЖЬ)
DTR	IN	Bool	Терминал данных готов, модуль готов передавать значение, подлежащее установке (ИСТИНА или ЛОЖЬ)
DSR	IN	Bool	Набор данных готов (применим только к интерфейсам типа DCE) (не используется)
DONE	OUT	Bool	ИСТИНА в течение одного цикла, после того как последний запрос был выполнен без ошибок
ERROR	OUT	Bool	ИСТИНА в течение одного цикла, после того как последний запрос был выполнен с ошибкой
STATUS	OUT	Word	Код условия выполнения

STATUS (W#16#....)	Описание
80F0	CM является модулем RS485 и никакие сигналы не могут быть установлены
80F1	Сигналы не могут быть установлены из-за аппаратного управления потоком
80F2	Сигнал готовности модема (DSR) не может быть установлен, так как модуль является терминальным оборудованием (DTE)
80F3	Терминал ввода данных (DTR) не может быть установлен, так как модуль является аппаратурой передачи данных (DCE)

8.7 Ошибки

Возвращаемые значения команд PtP

Каждая команда PtP имеет три выхода, которые отображают состояние исполнения:

Параметр	Тип данных	Значение по умолчанию	Описание
DONE	Bool	ЛОЖЬ	ИСТИНА в течение одного цикла указывает, что последний запрос выполнен без ошибок.
ERROR	Bool	ЛОЖЬ	ИСТИНА указывает, что последний запрос выполнен с ошибками, с соответствующим кодом ошибки в параметре STATUS.
STATUS	Word	0	Два байта, содержащих класс и номер ошибки, если она имеется. Параметр STATUS сохраняет это значение на протяжении исполнения этой функции.

Классы общих ошибок и ошибки

Описание класса	Классы ошибок	Описание
Конфигурирование порта	80Ax	Используется для определения общих ошибок конфигурирования порта
Конфигурирование передачи	80Bx	Используется для определения общих ошибок конфигурирования передачи
Конфигурирование приема	80Cx	Используется для определения общих ошибок конфигурирования приема
Время выполнения передачи	80Dx	Используется для определения общих ошибок во время выполнения передачи
Время выполнения приема	80Ex	Используется для определения общих ошибок во время выполнения приема
Обработка сигналов	80Fx	Используется для определения общих ошибок, связанных с обработкой всех сигналов

Ошибки конфигурирования порта

Идентификатор события/ошибки	Описание
0x80A0	Этот протокол не существует
0x80A1	Эта скорость передачи не существует
0x80A2	Этот контроль четности не существует
0x80A3	Это число битов данных не существует
0x80A4	Это число стоповых битов не существует
0x80A5	Этот тип управления потоком не существует

Ошибки конфигурирования передачи

Идентификатор события/ошибки	Описание
0x80B0	Этот протокол не существует
0x80B1	Эта скорость передачи не существует
0x80B2	Этот контроль четности не существует
0x80B3	Это число битов данных не существует
0x80B4	Это число стоповых битов не существует
0x80B5	Этот тип управления потоком не существует

Ошибки конфигурирования приема

Идентификатор события/ошибки	Описание
0x80C0	Ошибка стартового условия
0x80C1	Ошибка конечного условия
0x80C3	Ошибка максимальной длины
0x80C4	Ошибка значения N (см. N+LEN+M)
0x80C5	Ошибка значения длины (см. MAXLEN или N+LEN+M)
0x80C6	Ошибка значения M (см. N+LEN+M)
0x80C7	Ошибка значения N- LEN -M (см. N+LEN+M)
0x80C8	Ошибка времени ожидания ответа, никаких сообщений не было принято в течение заданного интервала времени приема. (См. RCVTIME или MSGTIME)
0x80C9	Ошибка интервала времени между символами (см. CHARGAP)
0x80CA	Ошибка времени простоя линии (см. Простаивающая линия)
0x80CB	Заданная конечная последовательность сконфигурирована со всеми "безразличными" символами
0x80CC	Заданная начальная последовательность сконфигурирована со всеми "безразличными" символами

Ошибки сигналов

Идентификатор события/ошибки	Описание
0x80F0	Коммуникационный модуль является модулем RS485 и сигналы отсутствуют
0x80F1	Коммуникационный модуль является модулем RS232, но сигналы не могут быть установлены, так как разблокировано аппаратное управление потоком
0x80F2	Сигнал DSR не может быть установлен, так как модуль является устройством DTE

Ошибки во время передачи

Идентификатор события/ошибки	Описание
Граничное значение буфера	Был превышен общий доступный размер буфера передачи CP
0x80D0	Новый запрос был принят, когда передатчик был активен
0x80D1	Приемник издал запрос на управление потоком, чтобы остановить активную передачу и не возобновлять ее в течение указанного времени ожидания Эта ошибка генерируется также при аппаратном управлении потоком, если приемник не объявляет о готовности к приему (CTS) в течение указанного времени ожидания
0x80D2	Запрос на передачу был прерван, так как не был принят сигнал готовности (DSR) от DCE
0x80D3	Был превышен общий доступный размер буфера передачи CP
0x7000	Функция передачи не занята
0x7001	Функция передачи занята первым вызовом
0x7002	Функция передачи занята последующими вызовами (опросы после первого вызова)

Значения, возвращаемые во время приема

Идентификатор события/ошибки	Описание
0x80E0	Сообщение было завершено, так как приемный буфер полон
0x80E1	Сообщение было завершено в результате ошибки контроля четности
0x80E2	Сообщение было завершено в результате ошибки кадрирования
0x80E3	Сообщение было завершено в результате ошибки переполнения
0x80E4	Сообщение было завершено в результате того, что указанная длина превышает общий размер буфера
0x0094	Сообщение было завершено, так как была принята максимальная последовательность символов (MAXLEN)
0x0095	Сообщение было завершено, так как все сообщение не было принято за заданное время (MSGTIME)
0x0096	Сообщение было завершено, так как следующий символ не был принят на интервале времени ожидания следующего символа (CHARGAP)
0x0097	Сообщение было завершено, так как первый символ не был принят в течение заданного времени (RCVTIME)
0x0098	Сообщение было завершено, так как условие длины "n+len+m" было выполнено (N+LEN+M)
0x0099	Сообщение было завершено, так как было выполнено условие окончания сообщения (ENDSEQ)

Различные ошибки параметризации

Идентификатор события/ошибки	Описание
0x8n3A	Недопустимый указатель в параметре n
0x8070	Вся внутренняя память экземпляра используется
0x8080	Недействительный номер порта
0x8082	Параметризация потерпела неудачу, так как она уже выполняется в фоновом режиме
0x8083	Переполнение буфера. СМ вернул больше данных, чем допустимо.
0x8085	Параметр LEN параметр имеет значение 0 или больше, чем максимально допустимое значение
0x8088	Параметр LEN больше, чем область памяти, указанная в DATA

Инструментальные средства онлайнового режима и диагностики

9

9.1 Светодиоды состояния

CPU и модули вода/вывода используют светодиоды для предоставления информации о рабочем состоянии модуля или входов/выходов. У CPU имеются следующие индикаторы состояния:

- STOP/RUN
  - Постоянно горящий оранжевый свет указывает на состояние STOP
  - Постоянно горящий зеленый свет указывает на режим RUN
  - Мигающий (попеременно зеленый и оранжевый) указывает, что CPU находится в состоянии запуска
- ERROR
  - Мигающий красный указывает на ошибку, например, внутреннюю ошибку в CPU, ошибку карты памяти или ошибку конфигурирования (несогласованные модули)
  - Постоянно горящий красный указывает на неисправность аппаратуры
- MAINT (обслуживание) мигает всякий раз, как вы вставляете карту памяти. Затем CPU переходит в состояние STOP. После того как CPU перешел в состояние STOP, выполните одно из следующих действий, чтобы инициировать анализ карты памяти:
  - Переведите CPU в режим RUN
  - Выполните полное стирание памяти (MRES)
  - Выключите CPU и включите его снова

Описание	**STOP/RUN Оранжевый / Зеленый**	**ERROR Красный**	**MAINT Оранжевый**
Питание выключено	Выключен	Выключен	Выключен
Запуск, самотестирование, обновление программы ПЗУ	Мигающий (попеременно оранжевый и зеленый)	-	Выключен
Состояние STOP	Включен (оранжевый)	-	-
Режим RUN	Включен (оранжевый)	-	-
Удалите карту памяти	Включен (оранжевый)	-	Мигающий
Ошибка	Включен (оранжевый или зеленый)	Мигающий	-
Запрашивается обслуживание	Включен (оранжевый или зеленый)	-	Включен
Неисправность аппаратуры	Включен (оранжевый)	Включен	Выключен
Тестирование светодиода или в CPU повреждена программа ПЗУ	Мигающий (попеременно оранжевый и зеленый)	Мигающий	Мигающий

CPU предоставляет также два светодиода, которые указывают состояние связи через PROFINET. Откройте нижнюю крышку клеммного блока, чтобы увидеть светодиоды PROFINET.

- Link [соединение] (зеленый) включается, чтобы показать, что соединение выполнено успешно
- Rx/Tx (желтый) включается, чтобы показать активность передачи

CPU и каждый цифровой сигнальный модуль (SM) имеют по одному светодиоду канала ввода/вывода для каждого из цифровых входов и выходов. Светодиод канала ввода/вывода (зеленый) включается или выключается, чтобы показать состояние отдельного входа или выхода.

Кроме того, каждый цифровой SM имеет светодиод DIAG, который указывает состояние модуля:

- Зеленый указывает, что модуль готов к работе
- Красный указывает, что модуль неисправен или не готов к работе

Каждый аналоговый SM имеет по одному светодиоду канала ввода/вывода для каждого из аналоговых входов и выходов.

- Зеленый указывает, что канал сконфигурирован и активен
- Красный указывает на состояние ошибки отдельного входа или выхода

Кроме того, каждый аналоговый SM имеет светодиод DIAG, который указывает состояние модуля:

- Зеленый указывает, что модуль готов к работе
- Красный указывает, что модуль неисправен или не готов к работе

SM распознает наличие или отсутствие питания на модуле (с полевой стороны, если необходимо).

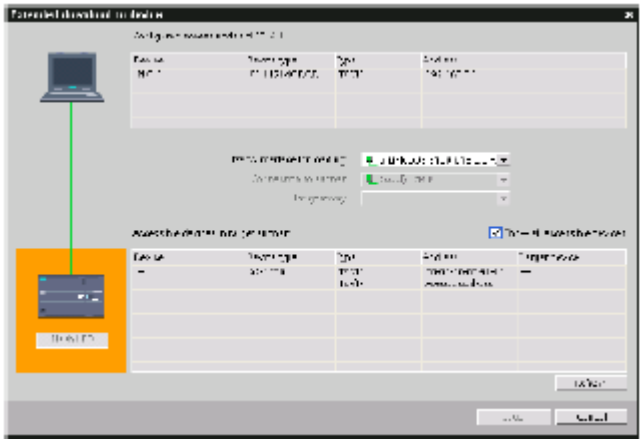
Описание	**DIAG (Красный / Зеленый)**	**Канал ввода/вывода (Красный / Зеленый)**
Питание с полевой стороны выключено	Мигающий красный	Мигающий красный
Не сконфигурирован или идет обновление	Мигающий зеленый	Выключен
Модуль сконфигурирован без ошибок	Включен (зеленый)	Включен (зеленый)
Сбойная ситуация	Мигающий красный	-
Ошибка ввода/вывода (при активной диагностике)	-	Мигающий красный
Ошибка ввода/вывода (при заблокированной диагностике)	-	Включен (зеленый)

9.2 Создание онлайнового соединения с CPU

Онлайновое соединение между устройством программирования и целевой системой необходимо для загрузки программ и данных проекта в целевую систему, а также, например, следующих действий:

- Тестирование программ пользователя
- Отображение и изменение режима работы CPU
- Отображение и установка даты и времени на CPU
- Отображение информации о модуле
- Сравнение онлайновых и оффлайновых блоков
- Диагностика аппаратуры

Затем вы сможете обратиться к данным в целевой системе в онлайновом или диагностическом представлении через панель задач "Online tools [Онлайновые инструментальные средства]".



Текущее онлайновое состояние устройства отображается пиктограммой справа рядом с устройством в отображении проекта.

Оранжевый цвет указывает на онлайновое соединение.

Выберите "Accessible Nodes [Доступные узлы]", чтобы найти CPU, находящиеся в сети.

Щелкните на "Go online [Перейти в онлайн]", чтобы соединиться с CPU в сети.

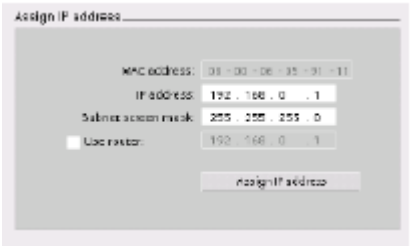
9.3 Установка IP-адреса и времени суток

Вы можете установить в онлайновом CPU IP-адрес и значение времени.

После подключения к онлайновому CPU из области "Online & diagnostics [Онлайновый режим и диагностика]" вы можете отобразить или изменить IP-адрес.

Дальнейшую информацию вы найдете в разделе об IP-адресе (стр. 84).

Вы можете также отобразить или установить параметры времени и даты онлайнового CPU.



9.4 Панель оператора для онлайнового CPU

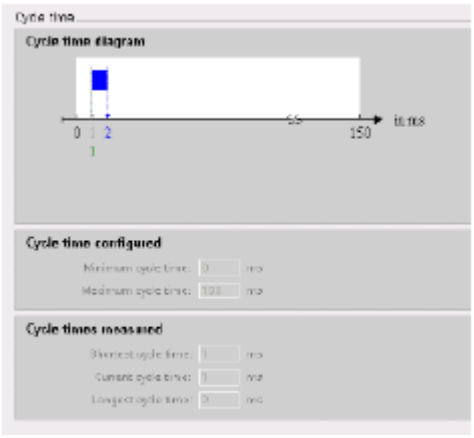
Панель задач "CPU operator panel [Панель оператора CPU]" отображает режим работы (STOP или RUN) онлайнового CPU: Эта панель показывает также, возникла ли в CPU ошибка и имеются ли принудительно установленные значения. Вы можете использовать панель оператора CPU для изменения режима работы онлайнового CPU.

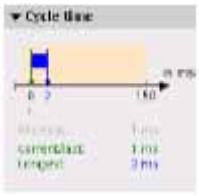
9.5 Контроль времени цикла и использования памяти

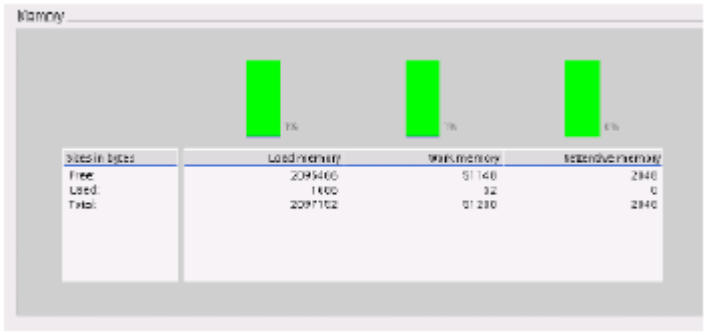
Вы можете контролировать время цикла и использование памяти онлайнового CPU.

После подключения к онлайновому CPU вы можете отобразить следующие измеренные значения:

- Время цикла
- Использование памяти



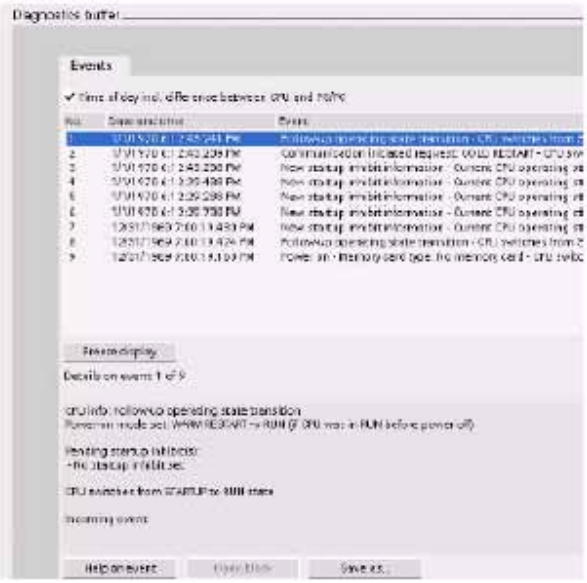




9.6 Отображение диагностических событий в CPU

В диагностическом буфере вы можете увидеть последние события в CPU. Диагностический буфер содержит следующие записи:

- Диагностические события
- Изменения в режиме работы CPU (переходы в STOP или RUN)



Первая запись соответствует последнему событию. Каждая запись в диагностическом буфере содержит дату и время регистрации события и его описание.

Максимальное число записей зависит от CPU. Поддерживается не более 50 записей.

Только последние 10 событий в диагностическом буфере сохраняются при отключении питания. Сброс CPU на заводские настройки очищает диагностический буфер, удаляя все записи.

9.7 Таблицы наблюдения для контроля программы пользователя

Таблица наблюдений позволяет осуществлять функции контроля и управления в информационных точках, когда CPU выполняет вашу программу. Этими информационными точками могут быть элементы образа процесса (I или Q), физические входы или выходы (I_:P или Q_:P), M или DB в зависимости от функции контроля и управления.

Функция контроля не изменяет процесс исполнения вашей программы. Она снабжает вас информацией об исполнении программы и данных программы в CPU.

Функции управления позволяют пользователю управлять последовательностью исполнения и данными программы. При использовании функций управления следует соблюдать осторожность. Эти функции могут существенно влиять на исполнение пользовательской или системной программы. Этими тремя функциями являются изменение, принудительное задание и разблокирование выходов в состоянии STOP.

С помощью таблицы наблюдения вы можете выполнять следующие онлайновые функции:

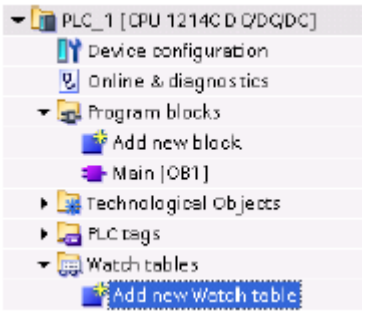
- Контроль состояния переменных
- Изменение значений отдельных переменных
- Принудительное присваивание переменной определенного значения

Вы можете выбрать, когда переменная должна наблюдаться или изменяться:

- Начало цикла: Значение считывается или записывается в начале цикла сканирования
- Конец цикла: Значение считывается или записывается в конце цикла сканирования
- Переключение в стоп

Для создания таблицы наблюдения:

1. Дважды щелкните на "Add new watch table [Добавить новую таблицу наблюдения]", чтобы открыть новую таблицу наблюдения.
2. Введите имя переменной, чтобы добавить переменную в таблицу наблюдения.



Для контроля переменных имеются следующие возможности:

- Monitor all [Контролировать все]: Эта команда запускает контроль видимых переменных в активной таблице наблюдения.
- Monitor now [Контролировать теперь]: Эта команда запускает контроль видимых переменных в активной таблице наблюдения. Таблица наблюдения выполняет контроль переменных немедленно и только один раз.

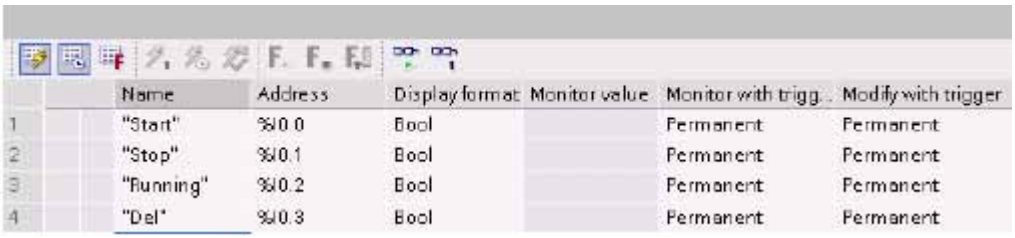
Для изменения переменных имеются в распоряжении следующие возможности:

- "Modify to 0 [Изменить на 0]" устанавливает значение выбранного адреса в "0".
- "Modify to 1 [Изменить на 1]" устанавливает значение выбранного адреса в "1".
- "Modify now [Изменить сейчас]" немедленно изменяет значение выбранных адресов на время одного цикла.
- "Modify with trigger [Инициирование изменений]" изменяет значение выбранных адресов.

  Эта функция не обеспечивает обратной связи, чтобы показать, что выбранные адреса были действительно изменены. Если требуется ответная реакция на изменения, используйте функцию "Modify now [Изменить сейчас]".

- "Enable peripheral outputs [Разблокировать периферийные выходы]" деактивизирует команду на блокировку выходов и имеется в распоряжении только тогда, когда CPU находится в состоянии STOP.

Для контроля переменных вы должны находиться в онлайновом соединении с CPU.



	Name	Address	Display format	Monitor value	Monitor with trigg...	Modify with trigger
1	"Start"	%I0.0	Bool		Permanent	Permanent
2	"Stop"	%I0.1	Bool		Permanent	Permanent
3	"Running"	%I0.2	Bool		Permanent	Permanent
4	"Del"	%I0.3	Bool		Permanent	Permanent

Различные функции могут быть выбраны с помощью кнопок в верхней части таблицы наблюдения.

Введите имя переменной для контроля и выберите формат отображения из ниспадающего списка. При наличии онлайнового соединения с CPU щелчок на кнопке "Monitor [Контролировать]" отобразит текущее значение информационной точки в поле "Monitor value [Контролируемое значение]".

Использование инициирования при контроле и изменении переменных ПЛК

Инициирование определяет, в какой точке цикла будут контролироваться или изменяться выбранные адреса.

Тип инициирования	**Описание**
Постоянное	Непрерывно регистрирует данные
В начале цикла сканирования	Постоянно: Постоянно регистрирует данные в начале цикла сканирования, после того как CPU прочитает входы
	Однократно: Регистрирует данные однократно в начале цикла сканирования, после того как CPU прочитает входы
В конце цикла сканирования	Постоянно: Постоянно регистрирует данные в конце цикла сканирования, перед тем как CPU запишет выходы
	Однократно: Регистрирует данные однократно в конце цикла сканирования, перед тем как CPU запишет выходы
При переходе в STOP	Постоянно: Постоянно регистрирует данные при переходах CPU в STOP
	Однократно: Регистрирует данные однократно после перехода CPU в STOP

Для изменения переменной ПЛК при заданном способе инициирования выберите начало или конец цикла.

- Изменение выхода: Лучшим инициирующим событием для изменения выхода является конец цикла, непосредственно перед которым CPU записывает выходы.

  Контролируйте значения выходов в начале цикла, чтобы определить, какое значение записано в физические выходы. Контролируйте также выходы перед тем, как CPU записывает значения в физические выходы, чтобы проверить логику программы и сравнить с фактическим поведением входов и выходов.

- Изменение входа: Лучшим инициирующим событием для изменения входа является начало цикла, непосредственно после того, как CPU считывает входы, и до того, как программа пользователя использует входные значения.

  Если вы изменяете входы в начале цикла, то вы должны также контролировать значение входов в конце цикла, чтобы убедиться, что значение входа в конце цикла не изменилось с начала цикла сканирования. Если имеется разница в значениях, то ваша пользовательская программа, возможно, записывает вход ошибочно.

Чтобы узнать, почему CPU перешел в STOP, используйте способ инициирования "Transition to STOP [Переход в STOP]", чтобы зарегистрировать последние значения процесса.

Разблокирование выходов в состоянии STOP

Таблица наблюдения дает вам возможность записывать значения в выходы, когда CPU находится в состоянии STOP. Эта функция позволяет вам проверять подключение выходов и гарантировать, что провод, подключенный к выходному контакту, инициирует сигнал высокого или низкого уровня на клемме устройства, к которой он подключен.

ПРЕДУПРЕЖДЕНИЕ
Хотя CPU находится в состоянии STOP, разблокирование физического выхода может активизировать точку процесса, к которой он подключен.

Вы можете изменять состояние выходов в режиме STOP, если выходы разблокированы. Если выходы заблокированы, то вы не можете их изменять в режиме STOP.

- Для разблокирования изменения выходов в состоянии STOP выберите опцию "Enable peripheral outputs [Разблокировать периферийные выходы]" команды "Modify [Изменять]" в меню "Online" или щелкните правой клавишей мыши на соответствующей строке таблицы наблюдения.
- Переход CPU в режим RUN блокирует опцию "Enable peripheral outputs".
- Если входам или выходам присвоены принудительные значения, то CPU не может разблокировать выходы в состоянии STOP. Функция присваивания принудительных значений сначала должна быть отменена.

Принудительное присваивание значений в CPU

CPU позволяет вам принудительно присваивать значения входам и выходам (форсировать входы и выходы), задавая адрес физического входа или выхода (I_:P или Q_:P) в таблице наблюдения и запуская форсирование.

В программе считывание физических входов перекрывается принудительно заданными значениями. Во время обработки программа использует принудительно заданные значения. Когда программа записывает физический выход, то значение выхода перекрывается принудительно заданным значением. Принудительно заданное значение появляется на физическом выходе и используется процессом.

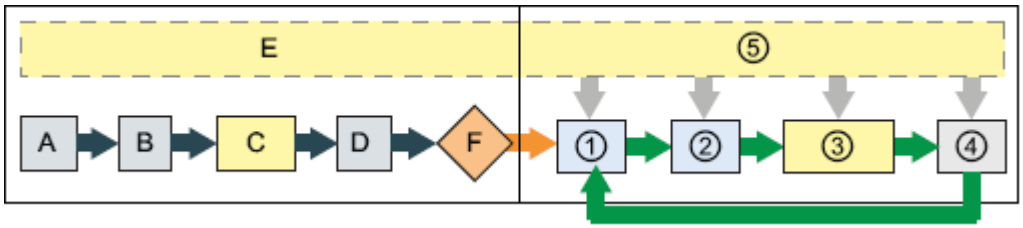
Если входу или выходу принудительное значение присваивается в таблице наблюдения, то эти действия становятся частью программы пользователя. И теперь, если даже программирующая программа будет закрыта, то принудительно установленные значения остаются активными в программе, исполняемой в CPU, пока вы их не отмените, перейдя в режим онлайн с помощью программирующей программы и остановив функцию форсирования. Программы с принудительно установленными входами и выходами, будучи загружены в другой CPU из карты памяти, будут продолжать форсировать входы и выходы, выбранные в программе.

Если CPU исполняет программу пользователя из защищенной от записи карты памяти, то вы не сможете инициировать или изменить форсирование входов/выходов из таблицы наблюдения, так как вы не можете заменить значения в защищенной от записи программе пользователя. Любая попытка принудительно изменить значения, защищенные от записи, генерирует ошибку. Если вы используете карту памяти для переноса программы пользователя, то все элементы на карте памяти будут переданы в CPU вместе с их принудительно установленными значениями.

Указание

Цифровым входам и выходам, назначенным HSC, PWM и PTO, значения не могут быть присвоены принудительно

Цифровые входы и выходы, используемые такими устройствами, как скоростной счетчик (HSC), широтно-импульсная модуляция (PWM) и вывод последовательности импульсов (PTO), назначаются во время конфигурирования устройства. Когда адреса цифровых входов и выходов назначаются этим устройствам, значения этих адресов не могут быть изменены функцией принудительного присваивания значений таблицы наблюдения.



Запуск

- A Функция принудительного присваивания значений не оказывает влияния на очистку области памяти входов (I).
- B Функция принудительного присваивания значений не оказывает влияния на инициализацию выходных значений.
- C При исполнении OB запуска CPU применяет принудительно установленные значения, когда программа пользователя обращается к физическому входу.
- D Не оказывается влияние на сохранение прерывающих событий в очереди.
- E Не оказывается влияние на разблокирование записи в выходы.

RUN

- ① При записи памяти выходов (Q) в физические выходы CPU применяет принудительно установленные значения, когда выходы обновляются.
- ② При чтении физических входов CPU применяет принудительно установленные значения непосредственно перед копированием входов в область памяти входов (I).
- ③ При исполнении программы пользователя (OB программного цикла), CPU применяет принудительно установленные значения, когда программа пользователя обращается к физическому входу или ведет запись в физический выход.
- ④ Функция принудительного присваивания значений не оказывает влияния на обработку коммуникационных запросов и самодиагностику.
- ⑤ Не оказывается влияние на обработку прерываний при исполнении любой части цикла.

Технические данные

A

A.1 Общие технические данные

Соответствие стандартам

Система автоматизации S7-1200 удовлетворяет следующим стандартам и тестовым техническим заданиям. Критерии тестирования для системы автоматизации S7-1200 основаны на этих стандартах и тестовых технических заданиях.

Допуск к эксплуатации CE

CE

Система автоматизации S7-1200 удовлетворяет требованиям и целям обеспечения безопасности перечисленных ниже директив ЕС и соответствует согласованным Европейским стандартам (EN) для программируемых контроллеров, опубликованным в официальных бюллетенях Европейского сообщества.

- Директива ЕС 2006/95/EC (Директива по низкому напряжению) "Электрическое оборудование, спроектированное для использования в определенных границах напряжения"
  - EN 61131-2:2007 Программируемые контроллеры - Требования к оборудованию и испытания
- Директива ЕС 2004/108/EC (Директива по ЭМС) "Электромагнитная совместимость"
  - Стандарт излучения
    EN 61000-6-4:2007: Промышленная среда
  - Стандарт помехозащищенности
    EN 61000-6-2:2005: Промышленная среда
- Директива ЕС 94/9/EC (ATEX) "Оборудование и системы защиты, предназначенные для использования в потенциально взрывоопасной газовой среде"
  - EN 60079-15:2005: Вид защиты 'n'

Декларация соответствия требованиям ЕС хранится для предоставления всем компетентным органам власти по адресу:

Siemens AG
IA AS RD ST ПЛК Amberg
Werner-von-Siemens-Str. 50
D92224 Amberg
Germany

Допуск к эксплуатации cULus

Выполнены требования Лабораторий страхователей (Underwriters Laboratories Inc.)

- Underwriters Laboratories [Лаборатории страхователей], Inc.: перечислены в UL 508 (Промышленные устройства управления)
- Canadian Standards Association [Канадская ассоциация стандартов]: CSA C22.2 номер 142 (Оборудование для управления процессами)

ВНИМАНИЕ
Серия SIMATIC S7-1200 удовлетворяет стандарту CSA. Логотип cULus указывает, что S7-1200 был проверен и сертифицирован Лабораториями страхователей (Underwriters Laboratories, UL) в соответствии со стандартами UL 508 и CSA 22.2 No. 142.

Сертификат FM

Factory Mutual Research [Совместные заводские исследования] (FM):
Стандартный класс допуска к эксплуатации №№ 3600 и 3611
Допущено для использования в:
Класс I, раздел 2, газовая группа A, B, C, D, класс температур T4A Ta = 40° C
Класс I, зона 2, IIC, класс температур T4 Ta = 40° C

Допуск к эксплуатации ATEX

EN 60079-0:2006: Взрывоопасные газовые среды – Общие требования
EN 60079-15:2005: Электрическое оборудование для потенциально взрывоопасных газовых сред;
Тип защиты 'n'
II 3 G Ex nA II T4

Необходимо выполнять следующие специальные условия для безопасного использования S7-1200:

- Устанавливайте модули в надлежащем корпусе, обеспечивающем минимальную степень защиты IP54 в соответствии с EN 60529, и учитывайте условия окружающей среды, в которых будет использоваться оборудование.
- Если температура при номинальных условиях превышает 70° C в точке ввода кабеля, или 80° C в точках разветвления проводов, то допустимая область температур выбранного кабеля должна соответствовать фактически измеренной температуре.
- Должны быть приняты меры предосторожности, чтобы предотвратить превышение номинального напряжения более чем на 40% при кратковременных нарушениях режима работы.

Допуск к эксплуатации C-Tick

Система автоматизации S7-1200 удовлетворяет требованиям стандартов AS/NZS 2064 (Class A)

Морской допуск к эксплуатации

Продукты S7-1200 периодически представляются на рассмотрение специальных агентств, относящихся к определенным рынкам и применениям, для получения допуска к эксплуатации. Если вам нужен список действующих в настоящее время допусков к эксплуатации для отдельных заказных номеров, обратитесь в местное представительство фирмы Siemens.

Классификационные общества:

- ABS (American Bureau of Shipping [Американское судовое бюро])
- BV (Bureau Veritas (Бюро Веритас)
- DNV (Det Norske Veritas [Норвежский Веритас])
- GL (Germanischer Lloyd [Германский Ллойд])
- LRS (Lloyds Register of Shipping [Судовой регистр Ллойда])
- Class NK (Nippon Kaiji Kyokai [Ниппон Кайдзи Кёкай - Япония])

Промышленная среда

Система автоматизации S7-1200 спроектирована для использования в промышленности.

Область применения	Требования к излучению помех	Требования к помехоустойчивости
Промышленность	EN 61000-6-4:2007	EN 61000-6-2:2005

Электромагнитная совместимость

Электромагнитная совместимость (ЭМС) – это способность электрического устройства работать должным образом в электромагнитной среде, не излучая электромагнитных помех такого уровня, который мог бы нарушить работу близлежащего электрического оборудования.

Электромагнитная совместимость – Помехозащищенность в соответствии с EN 61000-6-2	
EN 61000-4-2 Электростатический разряд	8 кВ – разряд через воздух на всех поверхностях 6 кВ – разряд через контакт с открытыми проводящими поверхностями
EN 61000-4-3 Излучаемое электромагнитное поле	от 80 до 1000 МГц, 10 В/м, 80%-ая амплитудная модуляция при частоте 1 кГц от 1-4 до 2,0 ГГц, 3 В/м, 80%-ая амплитудная модуляция при частоте 1 кГц от 2,0 до 2,7 ГГц, 1 В/м, 80%-ая амплитудная модуляция при частоте 1 кГц
EN 61000-4-4 Быстрые переходные помехи	2 кВ, 5 кГц в цепи связи с системой питания переменного и постоянного тока 2 кВ, 5 кГц на клемме входа или выхода
EN 6100-4-5 Устойчивость к резким скачкам	Системы переменного тока – синфазная помеха 2 кВ, противофазная помеха 1 кВ Системы постоянного тока - синфазная помеха 2 кВ, противофазная помеха 1 кВ Для систем постоянного тока (сигналы ввода/вывода, источники питания постоянного тока) требуется внешняя защита.
EN 61000-4-6 Наводки по цепям питания	от 150 кГц до 80 МГц, 10 $В_{эфф}$, 80%-ая амплитудная модуляция при частоте 1 кГц
EN 61000-4-11 Понижения напряжения	Системы переменного тока 0% в течение 1 цикла, 40% в течение 12 циклов и 70% в течение 30 циклов при частоте 60 Гц

Электромагнитная совместимость – Наведенные и излученные помехи в соответствии с EN 61000-6-4	
Наведенные помехи EN 55011, класс A, группа 1 от 0,15 МГц до 0,5 МГц от 0.5 МГц до 5 МГц от 5 МГц до 30 МГц	 <79 дБ (мкВ) квазипиковое; <66 дБ (мкВ) среднее значение <73 дБ (мкВ) квазипиковое; <60 дБ (мкВ) среднее значение <73 дБ (мкВ) квазипиковое; <60 дБ (мкВ) среднее значение
Излученные помехи EN 55011, класс A, группа 1 от 30 МГц до 230 МГц от 230 МГц до 1 ГГц	 <40dB (мкВ/м) квазипиковое; измеренное на расстоянии 10 м <47dB (мкВ/м) квазипиковое; измеренное на расстоянии 10 м

Условия окружающей среды

Условия окружающей среды - Транспортировка и хранение	
EN 60068-2-2, Тест Bb, сухое тепло и EN 60068-2-1, Тест Ab, холод	от -40° C до +70° C
EN 60068-2-30, Тест Db, влажное тепло	от 25° C до 55° C, влажность 95%
EN 60068-2-14, Тест Na, температурный удар	от -40° C до +70° C, время выдержки 3 часа, 2 цикла
EN 60068-2-32, свободное падение	0,3 м, 5 раз, в упаковке для отправки
Атмосферное давление	от 1080 до 660 гПа (соответствует высоте от -1000 до 3500 м)

Условия окружающей среды - Эксплуатация	
Диапазон температур окружающей среды (подача воздуха 25 мм под устройством)	от 0° C до 55° C при горизонтальном монтаже от 0° C до 45° C при вертикальном монтаже влажность 95% без конденсации
Атмосферное давление	от 1080 до 795 гПа (соответствует высоте от -1000 до 2000 м)
Концентрация загрязнений	$S0_2$: < 0,5 ‰; H_2S: < 0,1 ‰; относит. влажность < 60% без конденсации
EN 60068-2-14, Тест Nb, изменение температуры	от 5° C до 55° C, 3° C/мин.
EN 60068-2-27 Механический удар	15 g, импульс 11 мс, 6 ударов по каждой из 3 осей
EN 60068-2-6 Синусоидальные колебания	Монтаж на профильной шине: 3,5 мм, 5-9 Гц, 1 g, 9 - 150 Гц Монтаж на панели: 7,0 мм, 5-9 Гц, 2 g, от 9 до 150 Гц 10 качаний частоты по каждой оси, 1 октава в минуту

Испытание изоляции высоким напряжением	
Цепи тока с номинальным напряжением 24 В/5 В	520 В пост. тока (типовое испытание границ оптической развязки)
Цепи тока с напряжением 115/230 В относительно земли	Контрольное испытание 1500 В перем. тока /стандартное испытание 1950 В пост. тока
Цепи тока с напряжением 115/230 В относительно цепей тока 115/230 В	Контрольное испытание 1500 В перем. тока / стандартное испытание 1950 В пост. тока
Цепи тока с напряжением 115/230 В относительно цепей тока 24 В/5 В	Контрольное испытание 1500 В перем. тока / стандартное испытание 3250 В пост. тока

Класс защиты

- Класс защиты II в соответствии с EN 61131-2 (защитный провод не требуется)

Степень защиты

- IP20 Механическая защита, EN 60529
- Защита от прямого контакта с высоким напряжением, как например, при тестировании стандартным пробником. Необходима внешняя защита от пыли, грязи, воды и посторонних предметов диаметром < 12,5 мм.

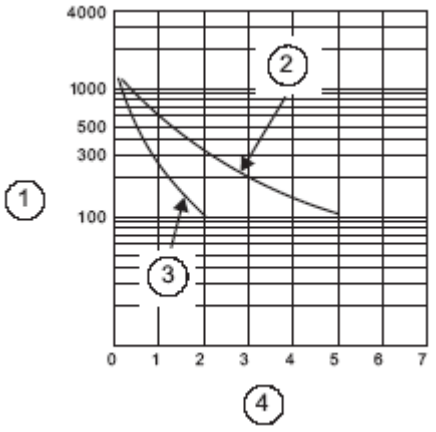
Номинальные напряжения

Номинальное напряжение	Допуск
24 В пост. тока	от 20,4 В пост. тока до 28,8 В пост. тока
120/230 В перем. тока	от 85 В перем. тока до 264 В перем. тока, от 47 до 63 Гц

ВНИМАНИЕ
Когда механический контакт включает выходное напряжение для CPU S7-1200 или любого цифрового сигнального модуля, он посылает сигнал "1" на цифровые выходы в течение примерно 50 микросекунд. Вы должны это учитывать, особенно, если вы используете устройства, которые реагируют на короткие импульсы.

Срок службы реле

Типовые эксплуатационные данные, предоставляемые в распоряжение изготовителями реле, приведены ниже. Фактические данные могут варьироваться в зависимости от того или иного приложения. Внешняя цепь защиты, согласованная с нагрузкой, увеличивает срок службы контактов.



① Срок службы (x 10^3 операций)

② 250 В перем. тока, омическая нагрузка,
30 В пост. тока, омическая нагрузка

③ 250 В перем. тока, индуктивная нагрузка
(коэффициент мощности = 0,4)
30 В пост. тока, индуктивная нагрузка
(L/R = 7 мс)

④ Номинальный рабочий ток (A)

A.2 CPU

A.2.1 Технические данные CPU 1211C

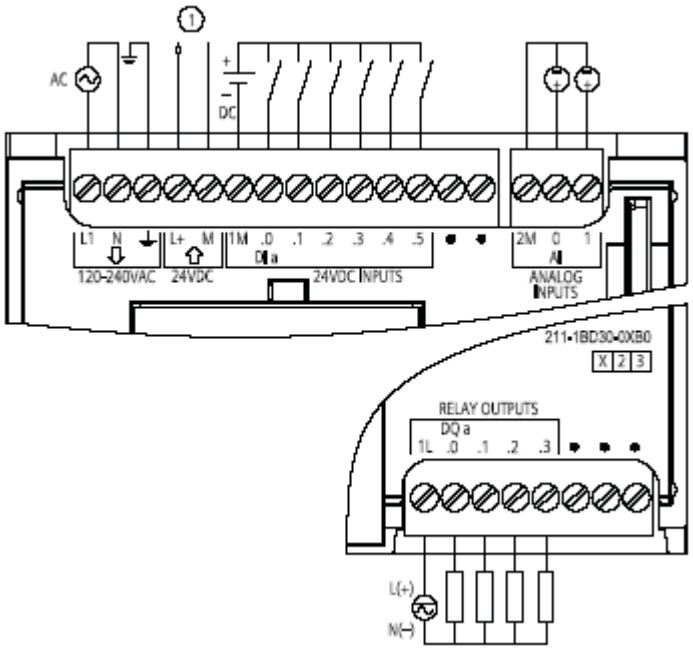
Технические данные			
Модель	**CPU 1211C AC/DC/Relay**	**CPU 1211C DC/DC/Relay**	**CPU 1211C DC/DC/DC**
Номер для заказа (MLFB)	6ES7 211-1BD30-0XB0	6ES7 211-1HD30-0XB0	6ES7 211-1AD30-0XB0
Общие данные			
Размеры Ш x В x Г (мм)	90 x 100 x 75		
Вес	420 грамм	380 грамм	370 грамм
Рассеиваемая мощность	10 Вт	8 Вт	
Располагаемый ток (шина CM)	макс. 750 мА (5 В пост. тока)		
Располагаемый ток (24 В пост. тока)	макс. 300 мА (питание датчиков)		
Потребление тока цифровым входом (24VDC)	4 мА/используемый вход		
Характеристики CPU			
Пользовательская память	Рабочая память 25 Кбайт / Загрузочная память 1 Мбайт / Сохраняемая память 2 Кбайта		
Встроенные цифровые входы/выходы	6 входов/4 выхода		
Встроенные аналоговые входы/выходы	2 входа		
Величина образа процесса	1024 байта входов (I) /1024 байта выходов (Q)		
Битовая память (M)	4096 байт		
Дополнительные сигнальные модули	Нет		
Дополнительные сигнальные платы	макс. 1 SB		
Дополнительные коммуникационные модули	макс. 3 CM		
Скоростные счетчики	3 всего Однофазные: 3 при 100 кГц Квадратурные: 3 при 80 кГц		
Импульсные выходы	2		
Входы для улавливания импульсов	6		
Прерывания с задержкой и циклические прерывания	Всего 4 с разрешением 1 мс		
Прерывания по фронту	6 по нарастающему и 6 по падающему (10 и 10 с дополнительной сигнальной платой)		
Карта памяти	Карта памяти SIMATIC (факультативно)		
Точность часов реального времени	+/- 60 секунд/месяц		
Длительность сохранения времени для часов реального времени	обычно 10 дней/мин. 6 дней при 40°C (не требующий обслуживания мощный конденсатор)		
Производительность			
Скорость выполнения булевых операций	0,1 мкс на команду		
Скорость выполнения команд над словами	12 мкс на команду		
Скорость выполнения арифметических команд	18 мкс на команду		

Технические данные			
Модель	**CPU 1211C AC/DC/Relay**	**CPU 1211C DC/DC/Relay**	**CPU 1211C DC/DC/DC**
Связь			
Число портов	1		
Тип	Ethernet		
Соединения	• 3 для устройств человеко-машинного интерфейса • 1 для устройства программирования • 8 для команд Ethernet в программе пользователя • 3 для соединения CPU с CPU		
Скорости передачи данных	10/100 Мбит/с		
Электрическая развязка (внешнего сигнала с логикой ПЛК)	Трансформатор с потенциальной развязкой, 1500 В пост. тока		
Тип кабеля	CAT5e экранированный		
Блок питания			
Диапазон напряжений	от 85 до 264 В перем. тока	от 20,4 до 28,8 В пост. тока	
Частота сети	от 47 до 63 Гц	--	
Входной ток только CPU при макс. нагрузке CPU CPU со всеми модулями расширения при макс. нагрузке	 60 мА при 120 В перем. тока 30 мА при 240 В перем. тока 180 мА при 120 В перем. тока 90 мА при 240 В перем. тока	 300 мА при 24 В пост. тока 900 мА при 24 В пост. тока	
Ток включения (макс.)	20 А при 264 В перем. тока	12 А при 28,8 В пост. тока	
Электрическая развязка (входного питания относительно логики)	1500 В перем. тока	Нет развязки	
Ток утечки на землю, от линии переменного тока на функциональную землю	0,5 мА макс.	-	
Время задержки (при потере питания)	20 мс при 120 В перем. тока 80 мс при 240 В перем. тока	10 мс при 24 В пост. тока	
Внутренний предохранитель, не заменяемый пользователем	3 А, 250 В, медленно перегорающий		
Питание датчиков			
Диапазон напряжений	от 20,4 до 28,8 В пост. тока	L+ минус мин. 4 В пост. тока	
Номинальный выходной ток (макс.)	300 мА (защищен от короткого замыкания)		
Максимальная пульсирующая помеха (<10 МГц)	< 1 В между пиками	Как на входном проводе	
Электрическая развязка (логики CPU относительно питания датчиков)	Нет развязки		
Цифровые входы			
Число входов	6		
Тип	Принимающий/поставляющий ток (IEC тип 1, принимающий ток)		
Номинальное напряжение	24 В пост. тока при 4 мА, номинальное значение		
Длительно допустимое напряжение	30 В пост. тока, макс.		
Импульсное напряжение	35 В пост. тока в течение 0,5 сек.		
Логический сигнал 1 (мин.)	15 В пост. тока при токе 2,5 мА		
Логический сигнал 0 (макс.)	5 В пост. тока при токе 1 мА		

Технические данные			
Модель	**CPU 1211C AC/DC/Relay**	**CPU 1211C DC/DC/Relay**	**CPU 1211C DC/DC/DC**
Электрическая развязка (полевая сторона относительно логики)	500 В перем. тока в течение 1 минуты		
Потенциально развязанные группы	1		
Постоянные времени фильтра	0,2; 0,4; 0,8; 1,6; 3,2; 6,4 и 12,8 мс (могут выбираться группами по 4 в каждой)		
Входные тактовые частоты HSC (макс.) (Логический уровень 1 от 15 до 26 В пост. тока)	Однофазные: 100 КГц Квадратурные: 80 КГц		
Число одновременно включенных входов	6		
Длина кабеля (в метрах)	500 экранированный, 300 неэкранированный, 50 экранированный для входов HSC		
Аналоговые входы			
Число входов	2		
Тип	Напряжение (несимметричное)		
Диапазон	От 0 до 10 В		
Полный диапазон (слово данных)	От 0 до 27648 (см. Представление аналогового входа для напряжения (стр. 361))		
Диапазон перерегулирования (слово данных)	От 27,649 до 32,511 (см. Представление аналогового входа для напряжения (стр. 361))		
Переполнение (слово данных)	От 32,512 до 32767 (см. Представление аналогового входа для напряжения (стр. 361))		
Разрешение	10 битов		
Максимальное выдерживаемое напряжение	35 В пост. тока		
Сглаживание	Отсутствует, слабое, среднее или сильное (см. Времена реакции аналогового входа (стр. 360) на единичный скачок)		
Подавление помех	10, 50 или 60 Гц (см. Времена реакции аналогового входа (стр. 360) для частот опроса)		
Полное сопротивление	≥100 КОм		
Электрическая развязка (полевая сторона относительно логики)	Нет		
Точность (25°C / от 0 до 55°C)	3,0% / 3,5% от всего диапазона		
Подавление синфазной помехи	40 дБ, от постоянного тока до 60 Гц		
Рабочий диапазон сигналов	Сигнал плюс напряжение синфазной помехи должно быть меньше +12 В и больше -12 В		
Длина кабеля (в метрах)	10 м, экранированная витая пара		
Цифровые выходы			
Число выходов	4		
Тип	Реле, сухой контакт		Транзисторный - MOSFET
Диапазон напряжений	от 5 до 30 В пост. тока или от 5 до 250 В перем. тока		от 20,4 до 28,8 В пост. тока
Логический сигнал 1 при макс. токе	--		мин. 20 В пост. тока
Логический сигнал 0 с нагрузкой 10 КОм	--		макс. 0,1 В пост. тока
Ток (макс.)	2,0 А		0,5 А
Ламповая нагрузка	30 Вт пост. тока / 200 Вт перем. тока		5 Вт
Сопротивление во включенном состоянии	макс. 0,2 Ом, если модуль новый		макс. 0,6 Ом
Ток утечки на выход	--		макс. 10 мкА
Ток включения	7 А при замкнутых контактах		макс. 8 А в течение 100 мс
Защита от перегрузки	Нет		
Электрическая развязка (полевая сторона относительно логики)	1500 В перем. тока в течение 1 минуты (катушка относительно контакта) Нет (катушка относительно логики)		500 В перем. тока в течение 1 минуты
Сопротивление изоляции	мин. 100 МОм, если модуль новый		--

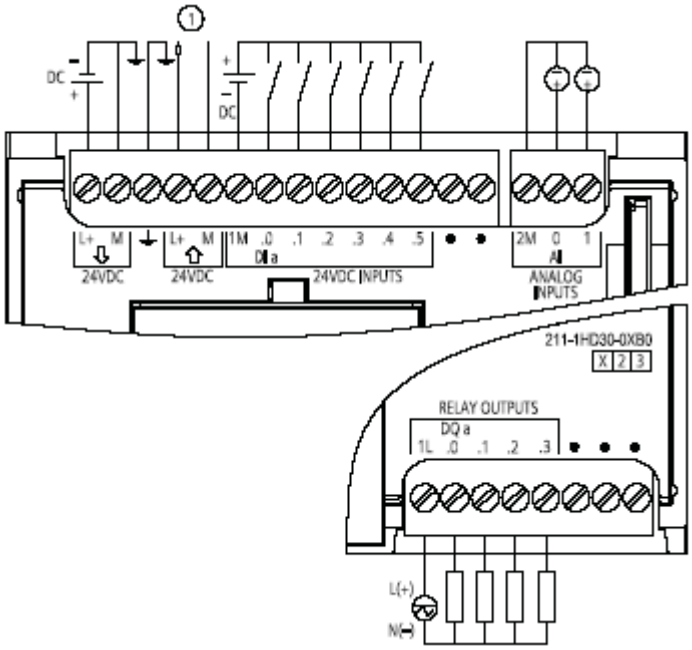
Технические данные			
Модель	**CPU 1211C AC/DC/Relay**	**CPU 1211C DC/DC/Relay**	**CPU 1211C DC/DC/DC**
Электрическая развязка между открытыми контактами	750 В перем. тока в течение 1 минуты		--
Потенциально развязанные группы	1		1
Индуктивное напряжение на клеммах	--		L+ минус 48 В пост. тока, потеря мощности 1 Вт
Задержка включения (от Qa.0 до Qa.3)	макс. 10 мс		макс. 1,0 мкс, из выкл. во вкл. макс. 3,0 мкс, из вкл. в выкл.
Частота генератора импульсов (Qa.0 и Qa.2)	Не рекомендуется		макс. 100 КГц, мин. 2 Гц
Механический срок службы (без нагрузки)	10 000 000 циклов откр./закр.		--
Срок службы контактов при номинальной нагрузке	100 000 циклов откр./закр.		--
Поведение при переходе из RUN в STOP	Последнее значение или заменяющее значение (значение по умолчанию 0)		
Число одновременно включенных выходов	4		
Длина кабеля (в метрах)	500 экранированный, 150 неэкранированный		

Схемы соединений



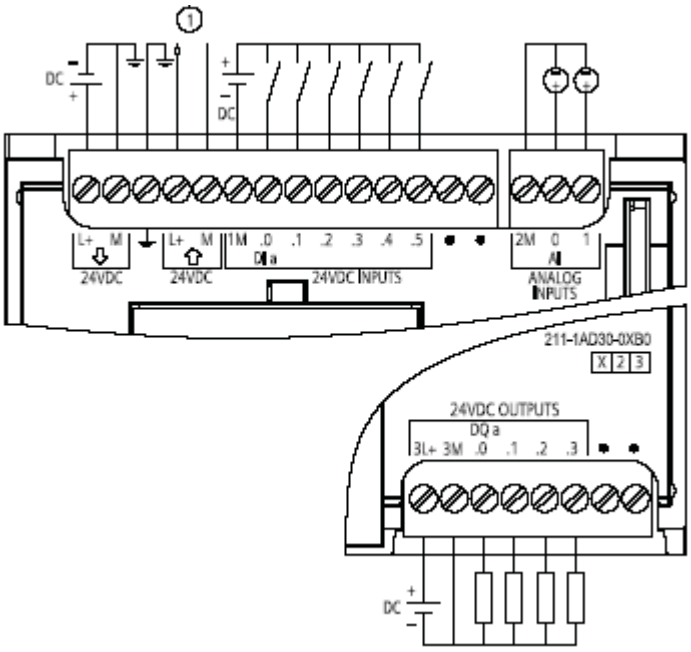
① Питание датчиков 24 В пост. тока

Рис. A-1. CPU 1211C AC/DC/Relay (6ES7 211-1BD30-0XB0)



① Питание датчиков 24 В пост. тока

Рис. A-2. CPU 1211C DC/DC/Relay (6ES7 211-1HD30-0XB0)



① Питание датчиков 24 В пост. тока

Рис. A-3. CPU 1211C DC/DC/DC (6ES7 211-1AD30-0XB0)

A.2.2 Технические данные CPU 1212C

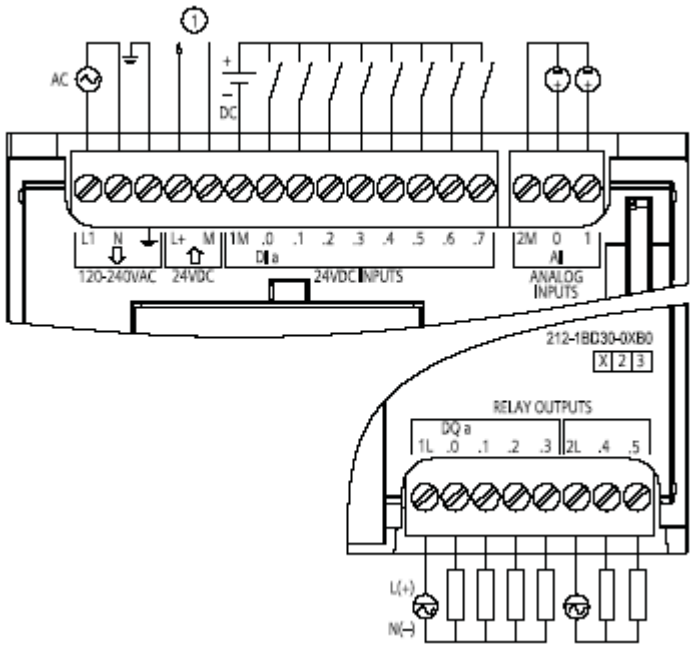
Технические данные			
Модель	**CPU 1212C AC/DC/Relay**	**CPU 1212C DC/DC/Relay**	**CPU 1212C DC/DC/DC**
Номер для заказа (MLFB)	6ES7 212-1BD30-0XB0	6ES7 212-1HD30-0XB0	6ES7 212-1AD30-0XB0
Общие данные			
Размеры Ш x В x Г (мм)	90 x 100 x 75		
Вес	425 грамм	385 грамм	370 грамм
Рассеиваемая мощность	11 Вт	9 Вт	
Располагаемый ток (SM и шина CM)	макс. 1000 мА (5 В пост. тока)		
Располагаемый ток (24 В пост. тока)	300 мА макс. (питание датчиков)		
Потребление тока цифровым входом (24 В пост. тока)	4 мА/используемый вход		
Характеристики CPU			
Пользовательская память	Рабочая память 25 Кбайт / Загрузочная память 1 Мбайт / Сохраняемая память 2 Кбайта		
Встроенные цифровые входы/выходы	8 входов/6 выходов		
Встроенные аналоговые входы/выходы	2 входа		
Величина образа процесса	1024 байта входов (I)/1024 байта выходов (Q)		
Битовая память (M)	4096 байт		
Дополнительные сигнальные модули	макс. 2 SM		
Дополнительные сигнальные платы	макс. 1 SB		

Технические данные			
Модель	**CPU 1212C AC/DC/Relay**	**CPU 1212C DC/DC/Relay**	**CPU 1212C DC/DC/DC**
Дополнительные коммуникационные модули	макс. 3 CM		
Скоростные счетчики	всего 4 Однофазные: 3 при тактовой частоте 100 кГц и 1 при тактовой частоте 30 кГц Квадратурные: 3 при тактовой частоте 80 кГц и 1 при тактовой частоте 20 кГц		
Импульсные выходы	2		
Входы для улавливания импульсов	8		
Прерывания с задержкой и циклические прерывания	Всего 4 с разрешением 1 мс		
Прерывания по фронту	8 по нарастающему и 8 по падающему (12 и 12 с дополнительной сигнальной платой)		
Карта памяти	Карта памяти SIMATIC (факультативно)		
Точность часов реального времени	+/- 60 секунд/месяц		
Длительность сохранения времени для часов реального времени	обычно 10 дней/мин. 6 дней при 40°C (не требующий обслуживания мощный конденсатор)		
Производительность			
Скорость выполнения булевых операций	0,1 мкс на команду		
Скорость выполнения команд над словами	12 мкс на команду		
Скорость выполнения арифметических команд	18 мкс на команду		
Связь			
Число портов	1		
Тип	Ethernet		
Соединения	• 3 для устройств человеко-машинного интерфейса • 1 для устройства программирования • 8 для команд Ethernet в программе пользователя • 3 для соединения CPU с CPU		
Скорости передачи данных	10/100 Мбит/с		
Электрическая развязка (внешнего сигнала с логикой ПЛК)	Трансформатор с потенциальной развязкой, 1500 В пост. тока		
Тип кабеля	CAT5e экранированный		
Блок питания			
Диапазон напряжений	от 85 до 264 В перем. тока	от 20,4 до 28,8 В пост. тока	
Частота сети	от 47 до 63 Гц	--	
Входной ток только CPU при макс. нагрузке CPU CPU со всеми модулями расширения при макс. нагрузке	80 мА при 120 В перем. тока 40 мА при 240 В перем. тока 240 мА при 120 В перем. тока 120 мА при 240 В перем. тока	400 мА при 24 В пост. тока 1200 мА при 24 В пост. тока	
Ток включения (макс.)	20 А при 264 В перем. тока	12 А при 28,8 В пост. тока	
Электрическая развязка (входного питания относительно логики)	1500 В перем. тока	Нет развязки	
Ток утечки на землю, от линии переменного тока на функциональную землю	макс. 0,5 мА	-	
Время задержки (при потере питания)	20 мс при 120 В перем. тока 80 мс при 240 В перем. тока	10 мс при 24 В пост. тока	

Технические данные			
Модель	**CPU 1212C AC/DC/Relay**	**CPU 1212C DC/DC/Relay**	**CPU 1212C DC/DC/DC**
Внутренний предохранитель, не заменяемый пользователем	3 А, 250 В, медленно перегорающий		
Питание датчиков			
Диапазон напряжений	от 20,4 до 28,8 В пост. тока	L+ минус мин. 4 В пост. тока	
Номинальный выходной ток (макс.)	300 мА (защищен от короткого замыкания)		
Максимальная пульсирующая помеха (<10 МГц)	< 1 В между пиками	Как на входном проводе	
Электрическая развязка (логики CPU относительно питания датчиков)	Нет развязки		
Цифровые входы			
Число входов	8		
Тип	Принимающий/поставляющий ток (IEC тип 1, принимающий ток)		
Номинальное напряжение	24 В пост. тока при 4 мА, номинальное значение		
Длительно допустимое напряжение	макс. 30 В пост. тока		
Импульсное напряжение	35 В пост. тока в течение 0,5 сек.		
Логический сигнал 1 (мин.)	15 В пост. тока при 2,5 мА		
Логический сигнал 0 (макс.)	5 В пост. тока при 1 мА		
Электрическая развязка (полевая сторона относительно логики)	500 В перем. тока в течение 1 минуты		
Потенциально развязанные группы	1		
Постоянные времени фильтра	0,2; 0,4; 0,8; 1,6; 3,2; 6,4 и 12,8 мс (могут выбираться группами по 4 в каждой)		
Входные тактовые частоты HSC (макс.) (Логический уровень 1 от 15 до 26 В пост. тока)	Однофазные: 100 КГц (от Ia.0 до Ia.5) и 30 КГц (от Ia.6 до Ia.7) Квадратурные: 80 КГц (от Ia.0 до Ia.5) и 20 КГц (от Ia.6 до Ia.7)		
Число одновременно включенных входов	8		
Длина кабеля (в метрах)	500 экранированный, 300 неэкранированный, 50 экранированный для входов HSC		
Аналоговые входы			
Число входов	2		
Тип	Напряжение (несимметричное)		
Диапазон	от 0 до 10 В		
Полный диапазон (слово данных)	от 0 до 27648 (Дальнейшую информацию вы найдете под заголовком Представление аналогового входа для напряжения (стр361))		
Диапазон перерегулирования (слово данных)	от 27,649 до 32,511 (Дальнейшую информацию вы найдете под заголовком Представление аналогового входа для напряжения (стр. 361))		
Переполнение (слово данных)	от 32,512 до 32767 (Дальнейшую информацию вы найдете под заголовком Представление аналогового входа для напряжения (стр. 361))		
Разрешение	10 битов		
Максимальное выдерживаемое напряжение	35 В пост. тока		
Сглаживание	Отсутствует, слабое, среднее или сильное (см. Времена реакции аналоговых входов (стр. 358) на единичный скачок)		
Подавление помех	10, 50 или 60 Гц (см. Времена реакции аналоговых входов (стр. 360) для частот опроса)		
Полное сопротивление	≥100 КОм		
Электрическая развязка (полевая сторона относительно логики)	Нет		
Точность (25°C / от 0 до 55°C)	3.0% / 3.5% от всего диапазона		
Подавление синфазной помехи	40 дБ, от постоянного тока до 60 Гц		

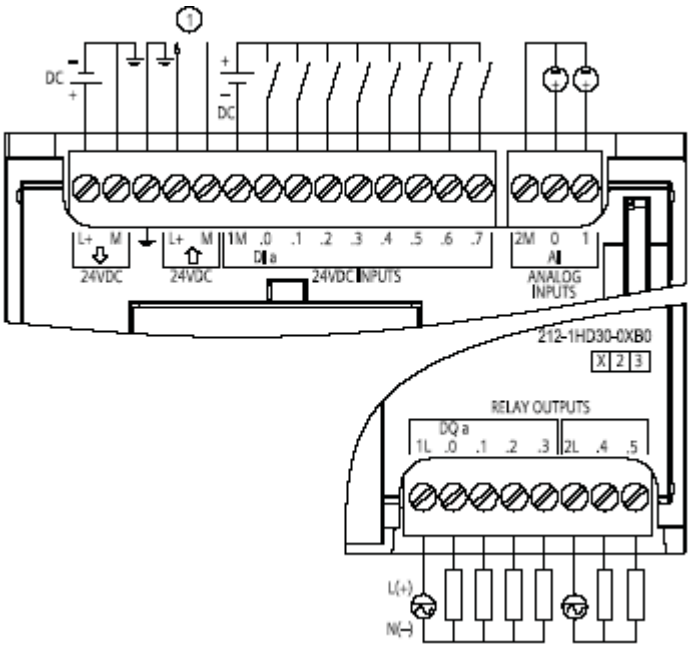
Технические данные			
Модель	**CPU 1212C AC/DC/Relay**	**CPU 1212C DC/DC/Relay**	**CPU 1212C DC/DC/DC**
Рабочий диапазон сигналов	Сигнал плюс напряжение синфазной помехи должно быть меньше +12 В и больше -12 В		
Длина кабеля (в метрах)	10, витой и экранированный		
Цифровые выходы			
Число выходов	6		
Тип	Реле, сухой контакт		Транзисторный - MOSFET
Диапазон напряжений	от 5 до 30 В пост. тока или от 5 до 250 В перем. тока		от 20,4 до 28,8 В пост. тока
Логический сигнал 1 при макс. токе	--		мин. 20 В пост. тока
Логический сигнал 0 с нагрузкой 10 КОм	--		макс. 0,1 В пост. тока
Ток (макс.)	2,0 А		0,5 А
Ламповая нагрузка	30 Вт DC / 200 Вт AC		5 Вт
Сопротивление во включенном состоянии	макс. 0,2 Ом, если модуль новый		макс. 0,6 Ом
Ток утечки на выход	--		макс. 10 мкА
Ток включения	7 А при замкнутых контактах		макс. 8 А в течение 100 мс
Защита от перегрузки	Нет		
Электрическая развязка (полевая сторона относительно логики)	1500 В перем. тока в течение 1 минуты (катушка относительно контакта) Нет (катушка относительно логики)		500 В перем. тока в течение 1 минуты
Сопротивление изоляции	мин. 100 МОм, если модуль новый		--
Электрическая развязка между открытыми контактами	750 В перем. тока в течение 1 минуты		--
Потенциально развязанные группы	2		1
Индуктивное напряжение на клеммах	--		L+ минус 48 В пост. тока, потеря мощности 1 Вт
Задержка включения (от Qa.0 до Qa.3)	макс. 10 мс		макс. 1,0 мкс, из выкл. во вкл. макс. 3,0 мкс, из вкл. в выкл.
Задержка включения (от Qa.4 до Qa.5)	макс. 10 мс		макс. 50 мкс, из выкл. во вкл. макс. 200 мкс, из вкл. в выкл.
Частота генератора импульсов (Qa.0 и Qa.2)	Не рекомендуется		макс. 100 КГц, мин. 2 Гц
Механический срок службы (без нагрузки)	10 000 000 циклов откр./закр.		--
Срок службы контактов при номинальной нагрузке	100 000 циклов откр./закр.		--
Поведение при переходе из RUN в STOP	Последнее значение или заменяющее значение (значение по умолчанию 0)		
Число одновременно включенных выходов	6		
Длина кабеля (в метрах)	500 экранированный, 150 неэкранированный		

Схемы соединений



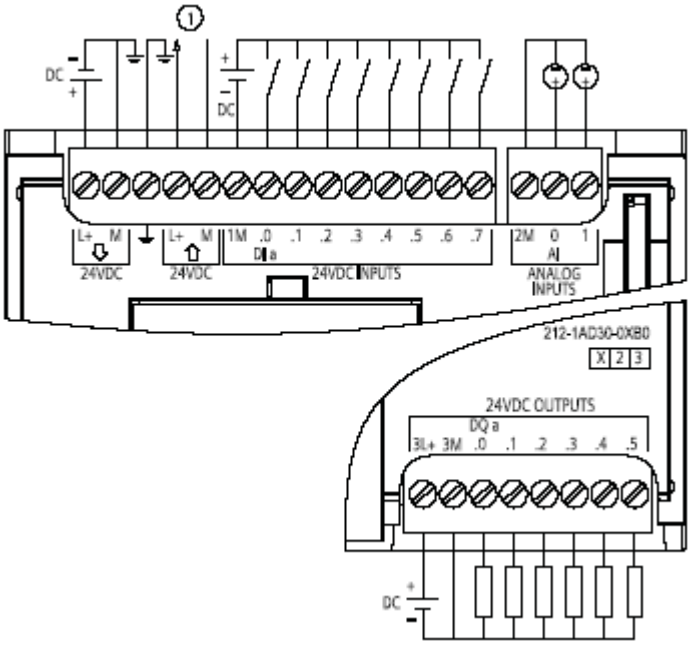
① Питание датчиков 24 В пост. тока

Рис. A-4.CPU 1212C AC/DC Relay (6ES7 212-1BD30-0XB0)



① Питание датчиков 24 В пост. тока

Рис. A-5. CPU 1212C DC/DC/Relay (6ES7 212-1HD30-0XB0)



① Питание датчиков 24 В пост. тока

Рис. A-6. CPU 1212C DC/DC/DC (6ES7 212-1AD30-0XB0)

A.2.3 Технические данные CPU 1214C

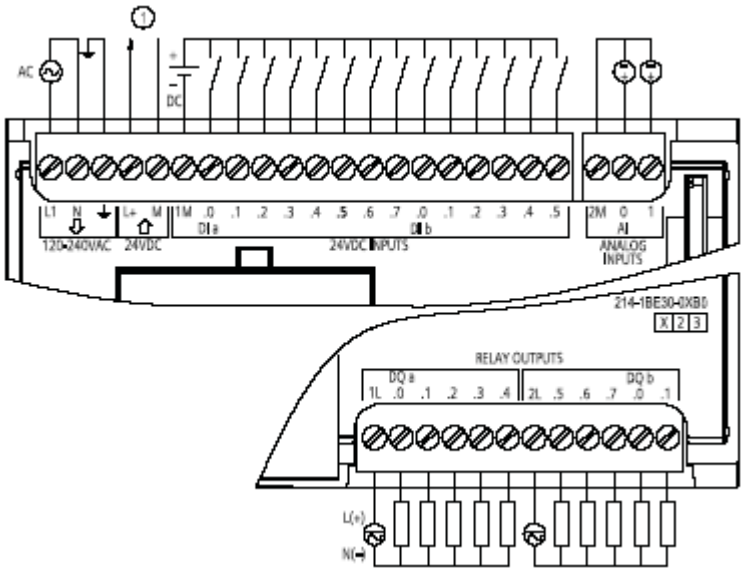
Технические данные			
Модель	**CPU 1214C AC/DC/Relay**	**CPU 1214C DC/DC/Relay**	**CPU 1214C DC/DC/DC**
Номер для заказа (MLFB)	6ES7 214-1BE30-0XB0	6ES7 214-1HE30-0XB0	6ES7 214-1AE30-0XB0
Общие данные			
Размеры Ш x В x Г (мм)	110 x 100 x 75		
Вес	475 грамм	435 грамм	415 грамм
Рассеиваемая мощность	14 Вт	12 Вт	
Располагаемый ток (SM и шина CM)	макс. 1600 мА (5 В пост. тока)		
Располагаемый ток (24 В пост. тока)	макс. 400 мА (питание датчиков)		
Потребление тока цифровым входом (24VDC)	4 мА/используемый вход		
Характеристики CPU			
Пользовательская память	Рабочая память 50 Кбайт / Загрузочная память 2 Мбайта / Сохраняемая память 2 Кбайта		
Встроенные цифровые входы/выходы	14 входов/10 выходов		
Встроенные аналоговые входы/выходы	2 входа		
Величина образа процесса	1024 байта входов (I)/1024 байта выходов (Q)		
Битовая память (M)	8192 байта		
Дополнительные сигнальные модули	макс. 8 SM		
Дополнительные сигнальные платы	макс. 1 SB		

Технические данные			
Модель	**CPU 1214C AC/DC/Relay**	**CPU 1214C DC/DC/Relay**	**CPU 1214C DC/DC/DC**
Дополнительные коммуникационные модули	макс. 3 CM		
Скоростные счетчики	6 всего Однофазные: 3 при тактовой частоте 100 кГц и 3 при тактовой частоте 30 кГц Квадратурные: 3 при тактовой частоте 80 кГц и 3 при тактовой частоте 20 кГц		
Импульсные выходы	2		
Входы для улавливания импульсов	14		
Прерывания с задержкой и циклические прерывания	Всего 4 с разрешением 1 мс		
Прерывания по фронту	12 по нарастающему и 12 по падающему (14 и 14 с дополнительной сигнальной платой)		
Карта памяти	Карта памяти SIMATIC (факультативно)		
Точность часов реального времени	+/- 60 секунд/месяц		
Длительность сохранения времени для часов реального времени	обычно 10 дней/мин. 6 дней при 40°C (не требующий обслуживания мощный конденсатор)		
Производительность			
Скорость выполнения булевых операций	0,1 мкс на команду		
Скорость выполнения команд над словами	12 мкс на команду		
Скорость выполнения арифметических команд	18 мкс на команду		
Связь			
Число портов	1		
Тип	Ethernet		
Соединения	• 3 для устройств человеко-машинного интерфейса • 1 для устройства программирования • 8 для команд Ethernet в программе пользователя • 3 для соединения CPU с CPU		
Скорости передачи данных	10/100 Мбит/с		
Электрическая развязка (внешнего сигнала с логикой ПЛК)	Трансформатор с потенциальной развязкой, 1500 В пост. тока		
Тип кабеля	CAT5e экранированный		
Блок питания			
Диапазон напряжений	от 85 до 264 В перем. тока	от 20,4 до 28,8 В пост. тока	
Частота сети	от 47 до 63 Гц	--	
Входной ток только CPU при макс. нагрузке CPU CPU со всеми модулями расширения при макс. нагрузке	 100 мА при 120 В перем. тока 50 мА при 240 В перем. тока 300 мА при 120 В перем. тока 150 мА при 240 В перем. тока	 500 мА при 24 В пост. тока 1500 мА при 24 В пост. тока	
Ток включения (макс.)	20 А при 264 В перем. тока	12 А при 28,8 В пост. тока	
Электрическая развязка (входного питания относительно логики)	1500 В перем. тока	Нет развязки	
Ток утечки на землю, от линии переменного тока на функциональную землю	макс. 0,5 мА	-	
Время задержки (при потере питания)	20 мс при 120 В перем. тока 80 мс при 240 В перем. тока	10 мс при 24 В пост. тока	

Технические данные			
Модель	**CPU 1214C AC/DC/Relay**	**CPU 1214C DC/DC/Relay**	**CPU 1214C DC/DC/DC**
Внутренний предохранитель, не заменяемый пользователем	3 А, 250 В, медленно перегорающий		
Питание датчиков			
Диапазон напряжений	от 20,4 до 28,8 В пост. тока	L+ минус мин. 4 В пост. тока	
Номинальный выходной ток (макс.)	400 мА (защищен от короткого замыкания)		
Максимальная пульсирующая помеха (<10 МГц)	< 1 В между пиками	Как на входном проводе	
Электрическая развязка (логики CPU относительно питания датчиков)	Нет развязки		
Цифровые входы			
Число входов	14		
Тип	Принимающий/поставляющий ток (IEC тип 1, принимающий ток)		
Номинальное напряжение	24 В пост. тока при 4 мА, номинальное значение		
Длительно допустимое напряжение	макс. 30 В пост. тока		
Импульсное напряжение	35 В пост. тока в течение 0,5 сек.		
Логический сигнал 1 (мин.)	15 В пост. тока при 2,5 мА		
Логический сигнал 0 (макс.)	5 В пост. тока при 1 мА		
Электрическая развязка (полевая сторона относительно логики)	500 В перем. тока в течение 1 минуты		
Потенциально развязанные группы	1		
Постоянные времени фильтра	0,2; 0,4; 0,8; 1,6; 3,2; 6,4 и 12,8 мс (могут выбираться группами по 4 в каждой)		
Входные тактовые частоты HSC (макс.) (Логический уровень 1 от 15 до 26 В пост. тока)	Однофазные: 100 КГц (от Ia.0 до Ia.5) и 30 КГц (от Ia.6 до Ib.5) Квадратурные: 80 КГц (от Ia.0 до Ia.5) и 20 КГц (от Ia.6 до Ib.5)		
Число одновременно включенных входов	14		
Длина кабеля (в метрах)	500 экранированный, 300 неэкранированный, 50 экранированный для входов HSC		
Аналоговые входы			
Число входов	2		
Тип	Напряжение (несимметричное)		
Диапазон	От 0 до 10 В		
Полный диапазон (слово данных)	От 0 до 27648 (Дальнейшую информацию вы найдете под заголовком Представление аналогового входа для напряжения (стр. 361))		
Диапазон перерегулирования (слово данных)	от 27,649 до 32,511 (Дальнейшую информацию вы найдете под заголовком Представление аналогового входа для напряжения (стр. 361))		
Переполнение (слово данных)	от 32,512 до 32767 (Дальнейшую информацию вы найдете под заголовком Представление аналогового входа для напряжения (стр. 361))		
Разрешение	10 битов		
Максимальное выдерживаемое напряжение	35 В пост. тока		
Сглаживание	Отсутствует, слабое, среднее или сильное (см. Время реакции аналогового входа (стр. 360) на единичный скачок)		
Подавление помех	10, 50 или 60 Гц (см. Время реакции аналогового входа (стр. 360) для частот опроса)		
Полное сопротивление	≥100 КОм		
Электрическая развязка (полевая сторона относительно логики)	Нет		
Точность (25°C / от 0 до 55°C)	3.0% / 3.5% от всего диапазона		
Подавление синфазной помехи	40 дБ, от постоянного тока до 60 Гц		

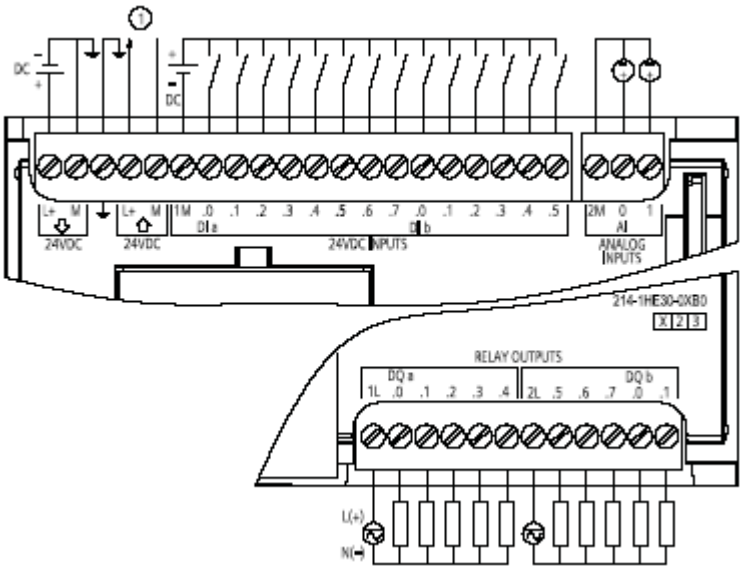
Технические данные			
Модель	**CPU 1214C AC/DC/Relay**	**CPU 1214C DC/DC/Relay**	**CPU 1214C DC/DC/DC**
Рабочий диапазон сигналов	Сигнал плюс напряжение синфазной помехи должно быть меньше +12 В и больше -12 В		
Длина кабеля (в метрах)	10, витой и экранированный		
Цифровые выходы			
Число выходов	10		
Тип	Реле, сухой контакт		Транзисторный - MOSFET
Диапазон напряжений	от 5 до 30 В пост. тока или от 5 до 250 В перем. тока		от 20,4 до 28,8 В пост. тока
Логический сигнал 1 при макс. токе	--		мин. 20 В пост. тока
Логический сигнал 0 с нагрузкой 10 КОм	--		макс. 0,1 В пост. тока
Ток (макс.)	2,0 А		0,5 А
Ламповая нагрузка	30 Вт пост. тока / 200 Вт перем. тока		5 Вт
Сопротивление во включенном состоянии	макс. 0,2 Ом, если модуль новый		макс. 0,6 Ом
Ток утечки на выход	--		макс. 10 мкА
Ток включения	7 А при замкнутых контактах		макс. 8 А в течение 100 мс
Защита от перегрузки	Нет		
Электрическая развязка (полевая сторона относительно логики)	1500 В перем. тока в течение 1 минуты (катушка относительно контакта) Нет (катушка относительно логики)		500 В перем. тока в течение 1 минуты
Сопротивление изоляции	мин. 100 МОм, если модуль новый		--
Электрическая развязка между открытыми контактами	750 В перем. тока в течение 1 минуты		--
Потенциально развязанные группы	2		1
Индуктивное напряжение на клеммах	--		L+ минус 48 В пост. тока, потеря мощности 1 Вт
Задержка включения (от Qa.0 до Qa.3)	макс. 10 мс		макс. 1,0 мкс, из выкл. во вкл. макс. 3,0 мкс, из вкл. в выкл.
Задержка включения (от Qa.4 до Qb.1)	макс. 10 мс		макс. 50 мкс, из выкл. во вкл. макс. 200 мкс, из вкл. в выкл.
Частота генератора импульсов (Qa.0 и Qa.2)	Не рекомендуется		макс. 100 КГц, мин. 2 Гц
Механический срок службы (без нагрузки)	10 000 000 циклов откр./закр.		--
Срок службы контактов при номинальной нагрузке	100 000 циклов откр./закр.		--
Поведение при переходе из RUN в STOP	Последнее значение или заменяющее значение (значение по умолчанию 0)		
Число одновременно включенных выходов	10		
Длина кабеля (в метрах)	500 экранированный, 150 неэкранированный		

Схемы соединений



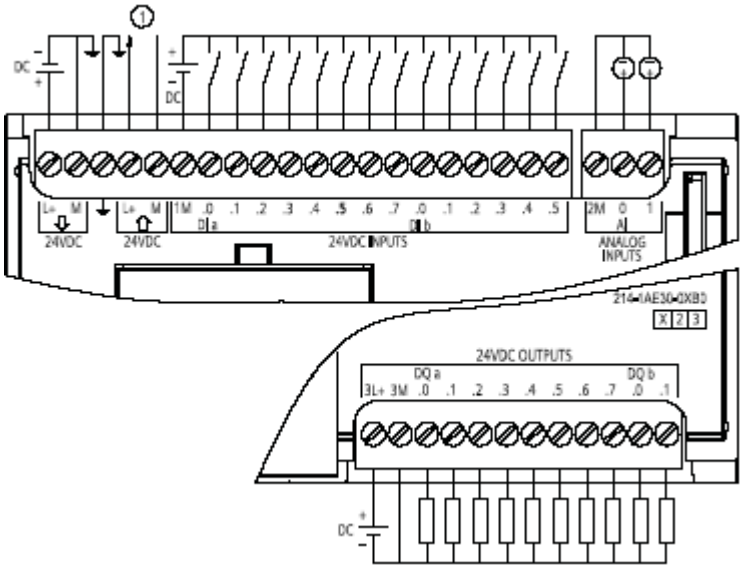
① Питание датчиков 24 В пост. тока

Рис. A-7. CPU 1214C AC/DC/Relay (6ES7 214-1BE30-0XB0)



① Питание датчиков 24 В пост. тока

Рис. A-8. CPU 1214C DC/DC/Relay (6ES7 214-1HE30-0XB0)



① Питание датчиков 24 В пост. тока

Рис. A-9. CPU 1214C DC/DC/DC (6ES7 214-1AE30-0XB0)

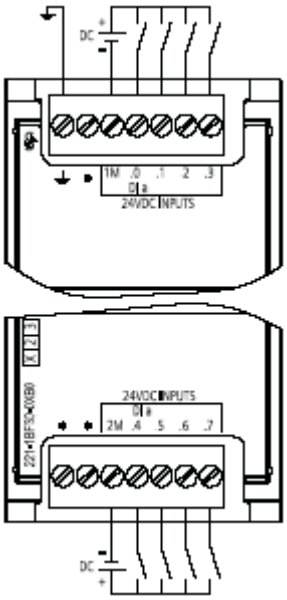
A.3 Цифровые сигнальные модули (SM)

A.3.1 Технические данные цифрового модуля ввода SM 1221

Технические данные		
Модель	**SM 1221 DI 8x24VDC**	**SM 1221 DI 16x24VDC**
Номер для заказа (MLFB)	6ES7 221-1BF30-0XB0	6ES7 221-1BH30-0XB0
Общие данные		
Размеры Ш x В x Г (мм)	45 x 100 x 75	
Вес	170 грамм	210 грамм
Рассеиваемая мощность	1,5 Вт	2,5 Вт
Потребляемый ток (шина SM)	105 мА	130 мА
Потребляемый ток (24 В пост. тока)	4 мА / используемый вход	4 мА / используемый вход
Цифровые входы		
Число входов	8	16
Тип	Принимающий/поставляющий ток (IEC тип 1, принимающий ток)	
Номинальное напряжение	24 В пост. тока при 4 мА, номинальное значение	
Длительно допустимое напряжение	макс. 30 В пост. тока	
Импульсное напряжение	35 В пост. тока в течение 0,5 сек.	
Логический сигнал 1 (мин.)	15 В пост. тока при 2,5 мА	
Логический сигнал 0 (макс.)	5 В пост. тока при 1 мА	
Электрическая развязка (полевая сторона относительно логики)	500 В перем. тока в течение 1 минуты	
Потенциально развязанные группы	2	4
Постоянные времени фильтра	0,2; 0,4; 0,8; 1,6; 3,2; 6,4 и 12,8 мс (могут выбираться группами по 4 в каждой)	
Число одновременно включенных входов	8	16
Длина кабеля (в метрах)	500 экранированный, 300 неэкранированный	

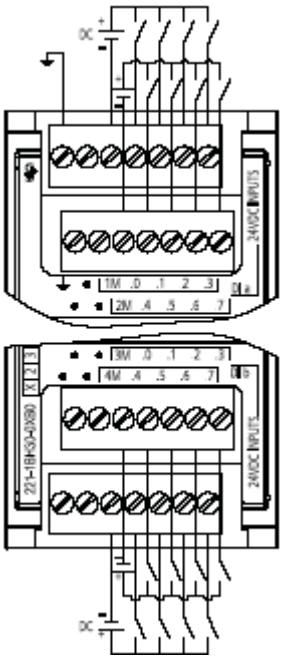
Схемы соединений

SM 1221 DI 8 x 24 VDC



6ES7 221-1BF30-0XB0

SM 1221 DI 16 x 24 VDC



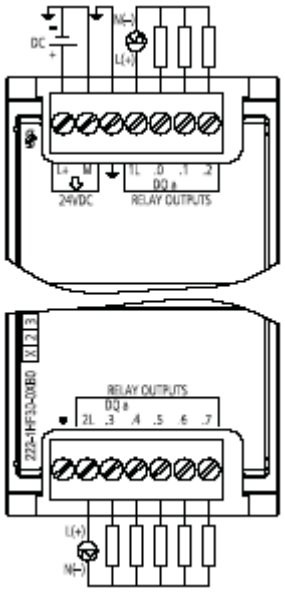
6ES7 221-1BH30-0XB0

A.3.2 Технические данные цифрового модуля вывода SM 1222

Технические данные				
Модель	**SM 1222 DQ 8xRelay**	**SM1222 DQ 16xRelay**	**SM1222 DQ 8x24VDC**	**SM1222 DQ 16x24VDC**
Номер для заказа (MLFB)	6ES7 222-1HF30-0XB0	6ES7 222-1HH30-0XB0	6ES7 222-1BF30-0XB0	6ES7 222-1BH30-0XB0
Общие данные				
Размеры Ш x В x Г (мм)	45 x 100 x 75			
Вес	190 грамм	260 грамм	180 грамм	220 грамм
Рассеиваемая мощность	4,5 Вт	8,5 Вт	1,5 Вт	2,5 Вт
Потребляемый ток (шина SM)	120 мА	135 мА	120 мА	140 мА
Потребляемый ток (24 В пост. тока)	11 мА / Используется катушка реле		--	
Цифровые выходы				
Число выходов	8	16	8	16
Тип	Реле, сухой контакт		Транзисторный - MOSFET	
Диапазон напряжений	от 5 до 30 В пост. тока или от 5 до 250 В перем. тока		от 20,4 до 28,8 В пост. тока	
Логический сигнал 1 при макс. токе	--		мин. 20 В пост. тока	
Логический сигнал 0 с нагрузкой 10 КОм	--		макс. 0,1 В пост. тока	
Ток (макс.)	2,0 А		0,5 А	
Ламповая нагрузка	30 Вт пост. тока/200 Вт перем. тока		5 Вт	
Сопротивление включенного контакта	макс. 0,2 Ом, если модуль новый		макс. 0,6 Ом	
Ток утечки на выход	--		макс. 10 мкА	
Ток включения	7 А при замкнутых контактах		макс. 8 А в течение 100 мс	
Защита от перегрузки	Нет			
Электрическая развязка (полевая сторона относительно логики)	1500 В перем. тока в течение 1 минуты (катушка относительно контакта) Нет (катушка относительно логики)		500 В перем. тока в течение 1 минуты	
Сопротивление изоляции	мин. 100 МОм, если модуль новый		--	
Электрическая развязка между открытыми контактами	750 В перем. тока в течение 1 минуты		--	
Потенциально развязанные группы	2	4	1	1
Ток на провод (макс.)	10 А		4 А	8 А
Индуктивное напряжение на клеммах	--		L+ минус 48 В, потеря мощности 1 Вт	
Задержка включения	макс. 10 мс		макс. 50 мкс, из выкл. во вкл. макс. 200 мкс, из вкл. в выкл.	
Механический срок службы (без нагрузки)	10 000 000 циклов откр./закр.		--	
Срок службы контактов при номинальной нагрузке	100 000 циклов откр./закр.		--	
Поведение при переходе из RUN в STOP	Последнее значение или заменяющее значение (значение по умолчанию 0)			
Число одновременно включенных выходов	8	16	8	16
Длина кабеля (в метрах)	500 экранированный, 150 неэкранированный			

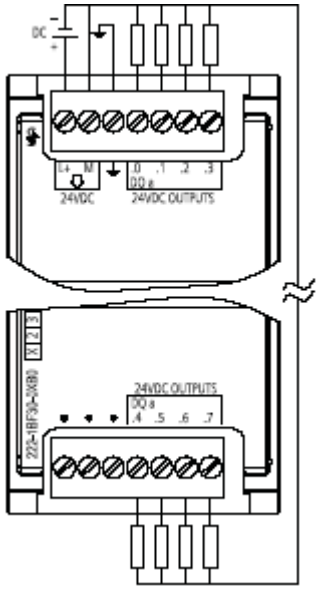
Схемы соединений

SM 1222 DQ 8 x Relay



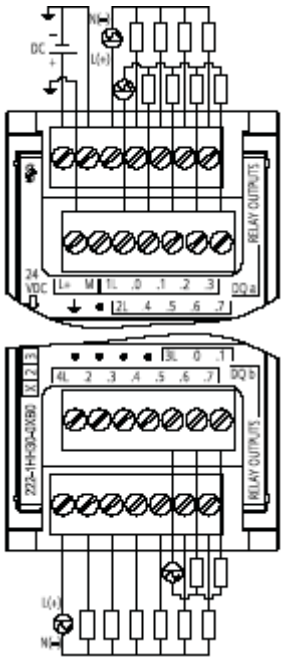
6ES7 222-1HF30-0XB0

SM 1222 DQ 8 x 24 VDC



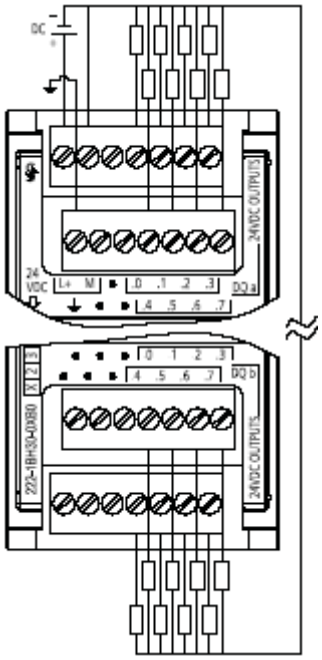
6ES7 222-1BF30-0XB0

SM 1222 DQ 16 x Relay



6ES7 222-1HH30-0XB0

SM 1222 DQ 16 x 24 VDC



6ES7 222-1BH30-0XB0

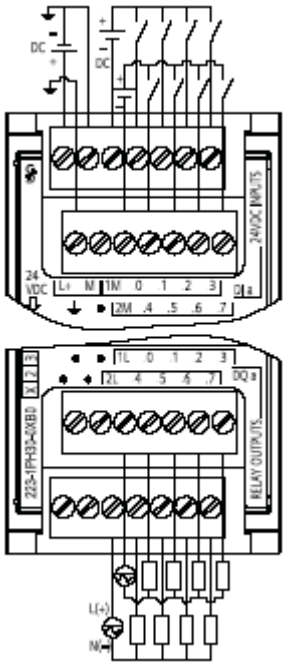
A.3.3 Технические данные цифрового модуля ввода/вывода SM 1223

Технические данные				
Модель	**SM 1223 DI 8x24 VDC, DQ 8xRelay**	**SM 1223 DI 16x24 VDC, DQ 16xRelay**	**SM 1223 DI 8x24 VDC, DQ 8x24 VDC**	**SM 1223 DI 16x24 VDC, DQ16x24 VDC**
Номер для заказа (MLFB)	6ES7 223-1PH30-0XB0	6ES7 223-1PL30-0XB0	6ES7 223-1BH30-0XB0	6ES7 223-1BL30-0XB0
Размеры Ш x В x Г (мм)	45 x 100 x 75	70 x 100 x 75	45 x 100 x 75	70 x 100 x 75
Вес	230 грамм	350 грамм	210 грамм	310 грамм
Рассеиваемая мощность	5,5 Вт	10 Вт	2,5 Вт	4,5 Вт
Потребляемый ток (шина SM)	145 мА	180 мА	145 мА	185 мА
Потребляемый ток (24 В пост. тока)	4 мА / используемый вход 11 мА / используемая катушка реле		4 мА / используемый вход	
Цифровые входы				
Число входов	8	16	8	16
Тип	Принимающий/поставляющий ток (IEC тип 1, принимающий ток)			
Номинальное напряжение	24 В пост. тока при 4 мА, номинальное значение			
Длительно допустимое напряжение	30 В пост. тока макс.			
Импульсное напряжение	35 В пост. тока в течение 0,5 сек.			
Логический сигнал 1 (мин.)	15 В пост. тока при 2,5 мА			
Логический сигнал 0 (макс.)	5 В пост. тока при 1 мА			
Электрическая развязка (полевая сторона относительно логики)	500 В перем. тока в течение 1 минуты			
Потенциально развязанные группы	2	2	2	2
Постоянные времени фильтра	0,2; 0,4; 0,8; 1,6; 3,2; 6,4 и 12,8 мс (могут выбираться группами по 4 в каждой)			
Число одновременно включенных входов	8	16	8	16
Длина кабеля (в метрах)	500 экранированный, 300 неэкранированный			
Цифровые выходы				
Число выходов	8	16	8	16
Тип	Реле, сухой контакт		Транзисторный - MOSFET	
Диапазон напряжений	от 5 до 30 В пост. тока или от 5 до 250 В перем. тока		от 20,4 до 28,8 В пост. тока	
Логический сигнал 1 при макс. токе	--		20 В пост. тока, мин.	
Логический сигнал 0 с нагрузкой 10 КОм	--		0,1 В пост. тока, макс.	
Ток (макс.)	2,0 А		0,5 А	
Ламповая нагрузка	30 Вт DC / 200 Вт AC		5 Вт	
Сопротивление включенного контакта	макс. 0,2 Ом, если модуль новый		макс. 0,6 Ом	
Ток утечки на выход	--		макс. 10 мкА	
Ток включения	7 А при замкнутых контактах		макс. 8 А в течение 100 мс	
Защита от перегрузки	Нет			
Электрическая развязка (полевая сторона относительно логики)	1500 В перем. тока в течение 1 минуты (катушка относительно контакта) Нет (катушка относительно логики)		500 В перем. тока в течение 1 минуты	
Сопротивление изоляции	мин. 100 МОм, если модуль новый		--	

Технические данные				
Модель	**SM 1223 DI 8x24 VDC, DQ 8xRelay**	**SM 1223 DI 16x24 VDC, DQ 16xRelay**	**SM 1223 DI 8x24 VDC, DQ 8x24 VDC**	**SM 1223 DI 16x24 VDC, DQ16x24 VDC**
Электрическая развязка между открытыми контактами	750 В перем. тока в течение 1 минуты		--	
Потенциально развязанные группы	2	4	1	1
Ток на провод	10A	8 A	4 A	8 A
Индуктивное напряжение на клеммах	--		L+ минус 48 В, потеря мощности 1 Вт	
Задержка включения	макс. 10 мс		макс. 50 мкс из выкл. во вкл. макс. 200 мкс, из вкл. в выкл.	
Механический срок службы (без нагрузки)	10 000 000 циклов откр./закр.		--	
Срок службы контактов при номинальной нагрузке	100 000 циклов откр./закр.		--	
Поведение при переходе из RUN в STOP	Последнее значение или заменяющее значение (значение по умолчанию 0)			
Число одновременно включенных выходов	8	16	8	16
Длина кабеля (в метрах)	500 экранированный, 150 неэкранированный			

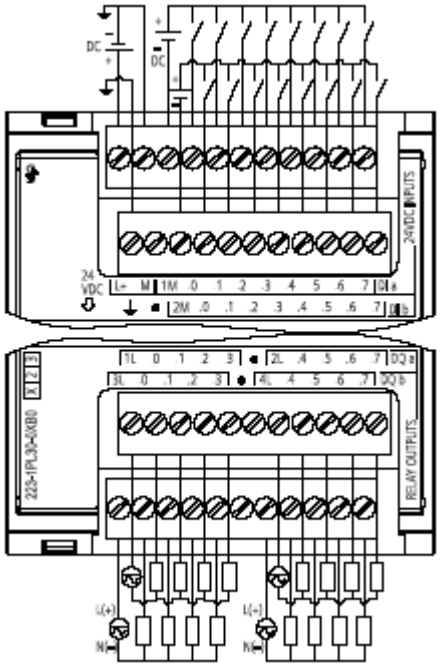
Схемы соединений

SM 1223 DI 8 x 24 VDC, DQ 8 x Relay



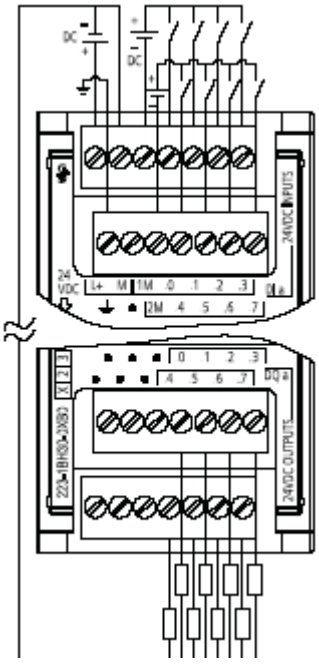
6ES7 223-1PH30-0XB0

SM1223 DI 16 x 24 VDC, DQ 16 x Relay



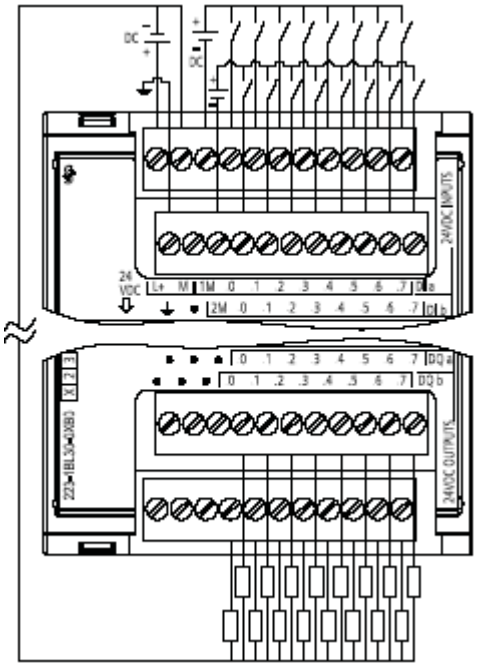
6ES7 223-1PL30-0XB0

SM 1223 DI 8 x 24 VDC, DQ 8 x 24 VDC



6ES7 223-1BH30-0XB0

SM 1223 DI 16 x 24 VDC, DQ 16 x 24 VDC



6ES7 223-1BL30-0XB0

A.4 Аналоговые сигнальные модули (SM)

A.4.1 Технические данные аналоговых сигнальных модулей SM 1231, SM 1232, SM 1234

Технические данные			
Модель	**SM 1231 AI 4x13bit**	**SM 1231 AI 8x13bit**	**SM 1234 AI 4x13bit AQ 2x14bit**
Номер для заказа (MLFB)	6ES7 231-4HD30-0XB0	6ES7 231-4HF30-0XB0	6ES7 234-4HE30-0XB0
Общие данные			
Размеры Ш x В x Г (мм)	45 x 100 x 75	45 x 100 x 75	45 x 100 x 75
Вес	180 грамм	180 грамм	220 грамм
Рассеиваемая мощность	1,5 Вт	1,5 Вт	2,0 Вт
Потребляемый ток (шина SM)	80 мА	90 мА	80 мА
Потребляемый ток (24 В пост. тока)	45 мА	45 мА	60 мА (без нагрузки)
Аналоговые входы			
Число входов	4	8	4
Тип	Напряжение или Ток (дифференциально): выбирается группами по 2		
Диапазон	±10 В, ±5 В, ±2,5 В или от 0 до 20 мА		
Полный диапазон (слово данных)	от -27,648 до 27,648		
Диапазон положительной/ отрицательной перегрузки (слово данных)	Напряжение: от 32,511 до 27,649 / от -27,649 до -32,512 Ток: от 32,511 до 27,649 / от 0 до -4864 (Дальнейшую информацию вы найдете под заголовками Представление аналогового входа для напряжения, Представление аналогового входа для тока (стр. 361))		
Положительное/отрицательное переполнение (слово данных)	Напряжение: от 32,767 до 32,512 / от -32,513 до -32,768 Ток: от 32,767 до 32,512 / от -4865 до -32,768 (Дальнейшую информацию вы найдете под заголовками Представление аналогового входа для напряжения, Представление аналогового входа для тока (стр. 361))		
Разрешение	12 битов + знаковый бит		
Максимальное выдерживаемое напряжение/ток	±35 В / ±40 мА		
Сглаживание	Отсутствует, слабое, среднее или сильное (см. Времена реакции аналоговых входов (стр. 360) на единичный скачок)		
Подавление помех	400, 60, 50 или 10 Гц (см. Времена реакции аналоговых входов (стр. 360) для частот опроса)		
Полное сопротивление	≥ 9 МОм (напряжение) / 250 Ом (ток)		
Электрическая развязка (полевая сторона относительно логики)	Нет		
Точность (25°C / от 0 до 55°C)	±0,1% / ±0,2% полного диапазона		
Время преобразования аналог-цифра	625 мкс (подавление 400 Гц)		
Подавление синфазной помехи	40 дБ, от постоянного тока до 60 Гц		
Рабочий диапазон сигналов	Сигнал плюс напряжение синфазной помехи должно быть меньше +12 В и больше -12 В		
Длина кабеля (в метрах)	100 метров, витой и экранированный		
Диагностика			
Положительное/отрицательное переполнение	Да[1]	Да[1]	Да[1]
Замыкание на землю (только для режима Напряжение)	Неприменимо	Неприменимо	Да, на выходах
Обрыв провода (только для режима Ток)	Неприменимо	Неприменимо	Да, на выходах
Низкое напряжение 24 В пост. тока	Да	Да	Да

[1] Если к входу приложить напряжение больше +30 В пост. тока или меньше -15 В пост. тока, то результирующее значение будет неизвестно, и соответствующее положительное или отрицательное переполнение, возможно, не будет активно.

Технические данные			
Модель	**SM 1231 AI 4x13bit**	**SM 1231 AI 8x13bit**	**SM 1234 AI 4x13bit AQ 2x14bit**
Технические данные			
Модель	**SM 1232 AQ 2x14bit**	**SM 1232 AQ 4x14bit**	**SM 1234 AI 4x13bit AQ 2x14bit**
Номер для заказа (MLFB)	6ES7 232-4HB30-0XB0	6ES7 232-4HD30-0XB0	6ES7 234-4HE30-0XB0
Общие данные			
Размеры Ш x В x Г (мм)	45 x 100 x 75	45 x 100 x 75	45 x 100 x 75
Вес	180 грамм	180 грамм	220 грамм
Рассеиваемая мощность	1,5 Вт	1,5 Вт	2,0 Вт
Потребляемый ток (шина SM)	80 мА	80 мА	80 мА
Потребляемый ток (24 В пост. тока)	45 мА (без нагрузки)	45 мА (без нагрузки)	60 мА (без нагрузки)
Аналоговые выходы			
Число выходов	2	4	2
Тип	Напряжение или ток		
Диапазон	±10 В или от 0 до 20 мА		
Разрешение	Напряжение: 14 битов; Ток: 13 битов		
Полный диапазон (слово данных)	Напряжение: от -27,648 до 27,648; ток: от 0 до 27,648 (Дальнейшую информацию вы найдете под заголовками Представление аналогового выхода для напряжения и Представление аналогового выхода для тока) (стр. 362)		
Точность (25°C / от 0 до 55°C)	±0,3% / ±0,6% полного диапазона		
Время установления (95% нового значения)	Напряжение: 300 мкс (R), 750 мкс (1 мкФ) ; Ток: 600 мкс (1 мГн), 2 мс (10 мГн)		
Полное сопротивление нагрузки	Напряжение: ≥ 1000 Ом; Ток: ≤ 600 Ом		
Поведение при переходе из RUN в STOP	Последнее значение или заменяющее значение (значение по умолчанию 0)		
Электрическая развязка (полевая сторона относительно логики)	Нет		
Длина кабеля (в метрах)	100 метров, витой и экранированный		
Диагностика			
Положительное/отрицательное переполнение	Да	Да	Да[1]
Замыкание на землю (только для режима Напряжение)	Да	Да	Да, на выходах
Обрыв провода (только для режима Ток)	Да	Да	Да, на выходах
Низкое напряжение 24 В пост. тока	Да	Да	Да

[1] Если к входу приложить напряжение больше +30 В пост. тока или меньше -15 В пост. тока, то результирующее значение будет неизвестно, и соответствующее положительное или отрицательное переполнение, возможно, не будет активно.

Времена реакции аналогового входа

Время реакции на ступенчатый сигнал аналоговых сигнальных модулей (мс)				
Напряжение от 0 до 10 В, измеренное при 95%				
Выбор сглаживания	**Подавляемая частота**			
	400 Гц	**60 Гц**	**50 Гц**	**10 Гц**
Отсутствует	4	18	22	100
Слабое	9	52	63	320
Среднее	32	203	241	1200
Сильное	61	400	483	2410
Частота опроса				
• 4 канала	• 0,625	• 4,17	• 5	• 25
• 8 каналов	• 1,25	• 4,17	• 5	• 25

Время реакции на ступенчатый сигнал аналоговых входов CPU (мс)			
Напряжение от 0 до 10 В, измеренное при 95%			
Выбор сглаживания	**Подавляемая частота**		
	60 Гц	**50 Гц**	**10 Гц**
Нет	63	65	130
Слабое	84	93	340
Среднее	221	258	1210
Сильное	424	499	2410
Частота опроса	**4,17**	**5**	**25**

Представление аналогового входа для напряжения

Система	**Диапазон измерения напряжения**						
Десятичная	**Шестнадцатеричная**	**±10 В**	**±5 В**	**±2,5 В**		**От 0 до 10 В**	
32767	7FFF	11,851 В	5,926 В	2,963 В	Переполнение	11,851 В	Переполнение
32512	7F00						
32511	7EFF	11,759 В	5,879 В	2,940 В	Перерегули-рование	11,759 В	Перерегули-рование
27649	6C01						
27648	6C00	10 В	5 В	2,5 В	Номинальный диапазон	10 В	Номинальный диапазон
20736	5100	7,5 В	3,75 В	1,875 В		7,5 В	
1	1	361,7 мкВ	180,8 мкВ	90,4 мкВ		361,7 мкВ	
0	0	0 В	0 В	0 В		0 В	
-1	FFFF					Отрицательные значения не поддерживаются	
-20736	AF00	-7,5 В	-3,75 В	-1,875 В			
-27648	9400	-10 В	-5 В	-2,5 В			
-27649	93FF				Отрицательный выброс		
-32512	8100	-11,759 В	-5,879 В	-2,940 В			
-32513	80FF				Отрицательное переполнение		
-32768	8000	-11,851 В	-5,926 В	-2,963 В			

Представление аналогового входа для тока

Система	**Диапазон измерения тока**		
Десятичная	**Шестнадцатеричная**	**От 0 до 20 мА**	
32767	7FFF	23.70 мА	Переполнение
32512	7F00		
32511	7EFF	23.52 мА	Перерегулирование
27649	6C01		
27648	6C00	20 мА	Номинальный диапазон
20736	5100	15 мА	
1	1	723.4 nA	
0	0	0 мА	
-1	FFFF		Отрицательный выброс
-4864	ED00	-3.52 мА	
-4865	ECFF		Отрицательное переполнение
-32768	8000		

Представление аналогового выхода для напряжения

Система		Диапазон выходного напряжения	
Десятичная	**Шестнадцатеричная**	**± 10 В**	
32767	7FFF	См. прим. 1	Переполнение
32512	7F00	См. прим. 1	
32511	7EFF	11,76 В	Перерегулирование
27649	6C01		
27648	6C00	10 В	Номинальный диапазон
20736	5100	7,5 В	
1	1	361,7 мкВ	
0	0	0 В	
-1	FFFF	-361,7 мкВ	
-20736	AF00	-7,5 В	
-27648	9400	-10 В	
-27649	93FF		Отрицательный выброс
-32512	8100	-11,76 В	
-32513	80FF	См. прим.1	Отрицательное переполнение
-32768	8000	См. прим. 1	

[1] При положительном или отрицательном переполнении аналоговые выходы ведут себя в соответствии со свойствами, установленными в конфигурации устройства для аналогового сигнального модуля. В параметре "Reaction to CPU STOP [Реакция на переход CPU в STOP]" выберите Use substitute value [Использовать заменяющее значение] или Keep last value [Сохранить последнее значение].

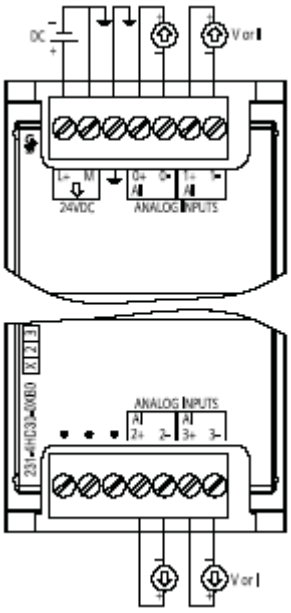
Представление аналогового выхода для тока

Система		Диапазон выходного тока	
Десятичная	**Шестнадцатеричная**	**± 20 мА**	
32767	7FFF	См. прим. 1	Переполнение
32512	7F00	См. прим. 1	
32511	7EFF	23.52 мА	Перерегулирование
27649	6C01		
27648	6C00	20 мА	Номинальный диапазон
20736	5100	15 мА	
1	1	723.4 нА	
0	0	0 мА	
-1	FFFF		Отрицательный выброс
-32512	8100		
-32513	80FF	См. прим.1	Отрицательное переполнение
-32768	8000	См. прим. 1	

1 При положительном или отрицательном переполнении аналоговые выходы ведут себя в соответствии со свойствами, установленными в конфигурации устройства для аналогового сигнального модуля. В параметре "Reaction to CPU STOP [Реакция на переход CPU в STOP]" выберите Use substitute value [Использовать заменяющее значение] или Keep last value [Сохранить последнее значение].

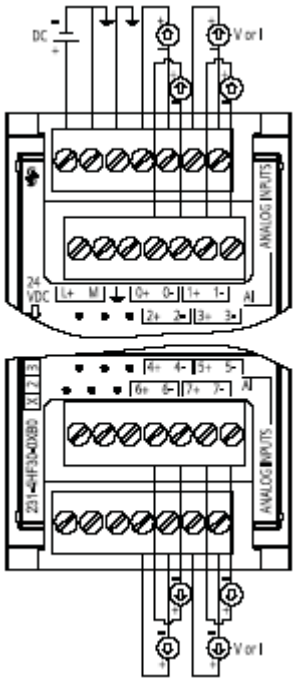
Схемы соединений

SM 1231 AI 4 x 13 Bit



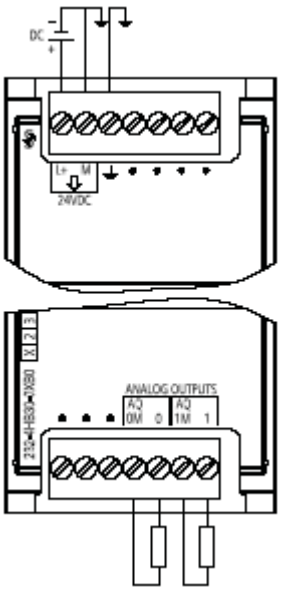
6ES7 231-4HD30-0XB0

SM 1231 AI 8 x 13 Bit



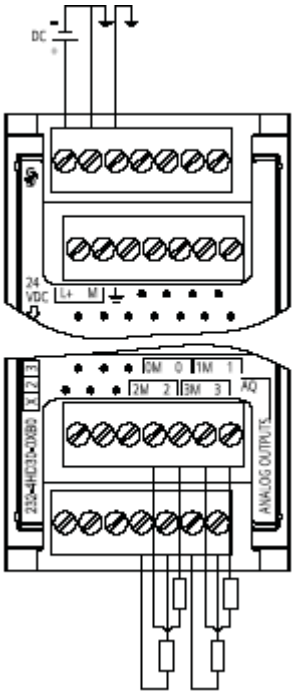
6ES7 231-4HF30-0XB0

SM 1232 AQ 2 x 14 Bit



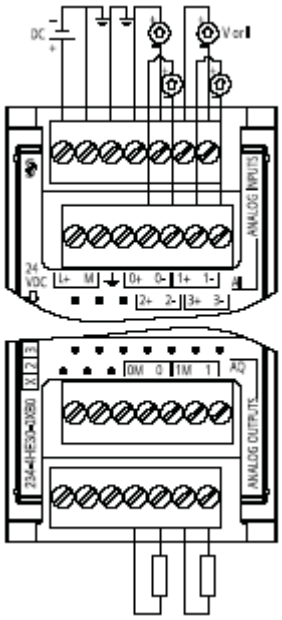
6ES7 232-4HB30-0XB0

SM 1232 AQ 4 x 14 Bit



6ES7 232-4HD30-0XB0

SM 1234 AI 4 x 13 Бит / AQ 2 x 14 бит



6ES7 234-4HE30-0XB0

A.4.2 Технические данные аналоговых сигнальных модулей SM 1231 RTD и TC

SM 1231 RTD аналоговый сигнальный модуль для подключения термосопротилений

Новый модуль расширения SM 1231 RTD предназначен для подключения термосопротивлений и могут использоваться совместно с S7-1200.

Сигнальный модуль	Заказной номер
SM 1231 AI 4 x RTD x 16 bit	6ES7 231-5PD30-0XB0

Модуль SM 1231 RTD измеряет значение сопротивления датчиков, подключенных к его входам. Значение может быть представлено в виде температуры или сопротивления.

● В случае измерения сопротивления, номинальный диапазон всей шкалы будет десятичным, максимальное значение 27648.
● В случае измерения температуры, значение будет в градусах, умноженных на 10. (например, 25,3 градуса будет отображаться как десятичное число 253).

Технические данные SM 1231 RTD

Модель	SM 1231 AI 4 x RTD x 16bit
Номер для заказа (MLFB)	6ES7 231-5PD30-0XB0
Размеры Ш x В x Г (мм)	45 x 100 x 75
Вес	220 г
Рассеиваемая мощность	1.5 Вт.
Потребляемый ток (шина SM)	80 мА
Потребляемый ток (24 В пост. тока) [1]	40 мА
Число входов	4
Тип	Доступные типы RTD
Диапазон	См. таблицу подбора датчиков RTD
Полный диапазон (слово данных)	См. таблицу подбора датчиков RTD
Диапазон положительной/ отрицательной перегрузки (слово данных)	См. таблицу подбора датчиков RTD
Положительное/отрицательное переполнение (слово данных)	См. таблицу подбора датчиков RTD
Разрешение Температура Сопротивление	 0.1°C/0.1°F 15 бит плюс знак
Максимальное напряжение	± 35 V
Подавление синфазной помехи	85 dB для выбранной установки фильтра (10 Гц, 50 Гц, 60 Гц, 60 Гц и 400 Гц)
Полное сопротивление	≥ 10 MΩ
Гальваническая развязка полевая сторона относительно логики полевая сторона от 24 VDC 24 VDC от логики	 500 VAC 500 VAC 500 VAC
Гальваническая развязка каналов	Нет
Точность	См. таблицу подбора датчиков RTD
Повторяемость	0.05% всего диапазона
Макс. рассеиваемая датчиком мощность	0.5 мВт
Принцип измерения	Интегрирование
Время обновления	См. таблицу выбора фильтра
Длина кабеля (в метрах)	100 метров до датчика
Сопротивление провода	20 Ω, 2.7 Ω для 10 Ω RTD max.
Ослабление синфазного сигнала	> 120dB
Диагностика	
Положительное/отрицательное переполнение [2]	
Обрыв провода[3]	
Низкое напряжение 24 В пост. тока [2]	

[1] от 20.4 до 28.8 VDC (Class 2, ограничение мощности или питание от CPU)

[2] Положительное, отрицательное переполнение и диагностика низкого напряжения питания можно диагностировать по значению аналогового сигнала, если отлючена функция диагностики в конфигурации модуля.
[3] Если диагностика обрыва провода отключена и возникают данные условия, модуль может выдавать случайные значения.

Таблица подбора датчиков RTD

Диапазоны и точность измерения для различных типов датчиков поддерживаемых модулем 1231 RTD указаны, в следующей таблице:

RTD Тип	Коэффициент Alpha	Ом	Нижний минимум	Нижний предел номинального диапазона	Верхний предел номинального диапазона	Верхний максимум	Точность при температуре 25℃	Точность при 0℃ .. 55℃
Pt	0.003850 ITS90 DIN EN 60751	10	-243.0℃	-200.0℃	850.0℃	1000.0℃	± 1.0℃	± 2.0℃
		50					± 0.5℃	± 1.0℃
		100						
		200						
		500						
		1000						
Pt	0.003902 0.003916 0.003920	100	-243.0℃	-200.0℃	850.0℃	1000.0℃	± 0.5℃	± 1.0℃
		200						
		500						
		1000						
Pt	0.003910	10	-273.2℃	-240.0℃	1100.0℃	1295℃	± 1.0℃	± 2.0℃
		50					± 0.8℃	± 1.6℃
		100						
		500						
Ni	0.006720 0.006180	100	-105.0℃	-60.0℃	250.0℃	295.0℃	± 0.5℃	±1.6℃
		120						
		200						
		500						
		1000						
LG-Ni	0.005000	1000						
Ni	0.006170	100	-105.0℃	-60.0℃	180.0℃	212.4℃	± 0.5℃	± 1.0℃
Cu	0.004270	10	-240.0℃	-200.0℃	280.0℃	312.0℃	± 0.7℃	± 1.4℃
Cu	0.004260	10	-60.0℃	-50.0℃	200.0℃	240.0℃	± 1.0℃	± 2.0℃
		50					± 0.6℃	± 1.2℃
		100						
Cu	0.004280	10	-240.0℃	-200.0℃	200.0℃	240.0℃	± 1.0℃	± 2.0℃
		50					± 0.7℃	± 1.4℃
		100						
Сопротивление								
Диапазон		150	n/a	0	150 Ω	176.383 Ω	± 0.05%	± 0.1%
		300	n/a	0	300 Ω	352.767 Ω	± 0.05%	± 0.1%
		600	n/a	0	600 Ω	705.534 Ω	± 0.05%	± 0.1%

Примечание:

Модуль будет показывать значение 32767 на каждом канале к которому не подключен датчик. Если активирована диагностика обрыва провода, то в этом случае модуль будет сигнализировать об это красным цветом светодиода.

Таблица выбора фильтра

Частота шумоподавления (Гц)	Время интегрирования (мс)	4/2 проводная схема, 4-х канальный модуль, время обновления (сек)	3 проводная схема, 4-х канальный модуль, время обновления (сек)
10	100	1.222	2.444
50	20	0.262	0.524
60	16.67	0.222	0.444
400[1]	10	0.142	0.284

[1] Для того, чтобы достичь заявленной точности и разрешения при фильтре 400 Гц, время интегрирования устанавливается равным 10 мс. Этот фильтр также отсекает помехи частот 100 Гц и 200 Гц.

Примечание:
После подачи питания на модуль, происходит внутренняя калибровка аналогово- цифрового преобразователя. В течение этого времени, модуль показывет значение 32767 на каждом канале до появления настоящего значения на данном канале. Алгоритм контроллера должен учитывать это время.

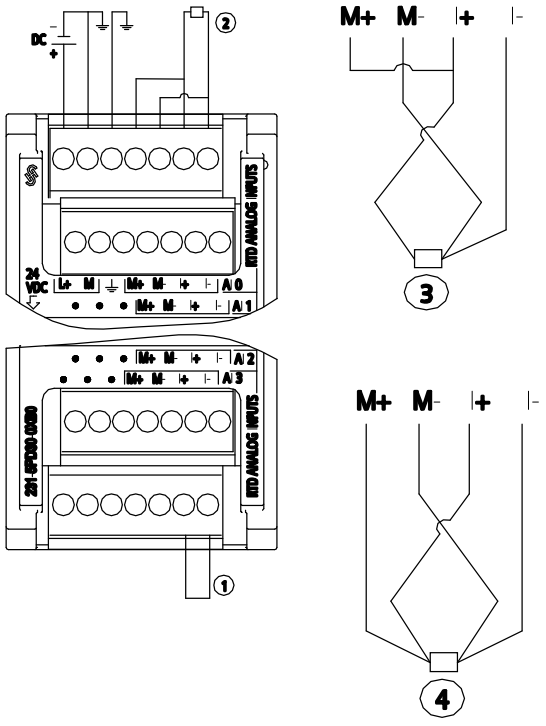
SM 1231 AI 4 x RTD x 16bit Схема подключения

1. Перемычка на неиспользуемых входах RTD

2. 2-х проводная схема RTD

3. 3-х проводная RTD

4. 4-х проводная RTD



SM 1231 TC аналоговый сигнальный модуль для подключения термопар

Новый модуль расширения SM 1231 TC предназначен для подключения термопар термосопротивлений и могут использоваться совместно с S7-1200.

Сигнальный модуль	Заказной номер
SM 1231 TC 4 x 16 bit	6ES7 231-5QD30-0XB0

Модуль SM 1231 RTD измеряет значение напряжения датчиков, подключенных к его входам. Значение может быть представлено в виде температуры или напряжения.
● В случае измерения напряжения, номинальный диапазон всей шкалы будет десятичным, максимальное значение 27648.
● В случае измерения температуры, значение будет в градусах, умноженных на 10. (например, 25,3 градуса будет отображаться как десятичное число 253).

Технические данные SM 1231 TC

Модель	SM 1231 AI 4 x 16 Bit Thermocouple
Номер для заказа (MLFB)	6ES7 231-5QD30-0XB0
Размеры Ш x В x Г (мм)	45 x 100 x 75
Вес	180 г.
Рассеиваемая мощность	1.5 Вт
Потребляемый ток (шина SM)	80 мА
Потребляемый ток (24 В пост. тока) [1]	
Число входов	4
Тип	TC
Диапазон	См. таблицу подбора термопар
Полный диапазон (слово данных)	Напряжение: от -27,648 до 27,648
Диапазон положительной/ отрицательной перегрузки (слово данных)	Напряжение: от 32,511 до 27,649 / от -27,49 до -32,512
Положительное/отрицательное переполнение (слово данных)	Напряжение: от 32,767 до 32,512 / от -32,513 до -32,768
Разрешение Температура Напряжение	0.1°C/0.1°F 15 бит + знак
Максимальное напряжение	± 35 В
Подавление синфазной помехи	85 dB 50 Hz/60 Hz/400 Hz
Ослабление синфазного сигнала	> 120dB при 120 VAC
Полное сопротивление	≥ 1 MΩ
Гальваническая развязка полевая сторона относительно логики полевая сторона от 24 VDC 24 VDC от логики	 500 VAC 500 VAC 500 VAC
Гальваническая развязка каналов	120 В AC
Точность	См. таблицу подбора термопар
Повторяемость	±0.5% FS
Время обновления	См. таблицу выбора фильтра
Погрешность холодного спая	±1.5℃
Длина кабеля (в метрах)	100 метров для датчика

Сопротивление провода	100 Ω максимально
Положительное/отрицательное переполнение [2]	
Обрыв провода[3]	
Низкое напряжение 24 В пост. тока [2]	

[1] от 20.4 до 28.8 VDC (Class 2, ограничение мощности или питание от CPU)
[2] Положительное, отрицательное переполнение и диагностика низкого напряжения питания можно диагностировать по значению аналогового сигнала, если отлючена функция диагностики в конфигурации модуля.
[3] Если диагностика обрыва провода отключена и возникают данные условия, модуль может выдавать случайные значения.

Основные данные по термопарам
Термопары представляют собой два разнородных металла электрически соединенные друг с другом. Генерируемое напряжение пропорционально температуре точки спая. Это напряжение очень мало; один микровольт может означать большое значение в градусах.
Измерение напряжения от термопары, компенсация для дополнительных точек соединения и последующая линеаризация результата составляют основу измерения температуры при помощи термопар.

Когда вы подключаете термопару к модулю EM 1231 Thermocouple, к модулю присоединяются два провода, соединенные с разными металлами, образующими термопару. Место, где два разнородных провода соединяются друг с другом, образует датчик термопары.

Еще две термопары образуются там, где два разнородных провода присоединяются к сигнальному разъему. Температура соединительного блока порождает напряжение, которое прибавляется к напряжению от термопары датчика. Если это напряжение не компенсируется, то сообщаемая температура отличается от температуры датчика.

Для компенсации термопар разъема используется компенсатор холодного спая. Таблицы термопар основываются на температуре холодного спая, обычно равной нулю по шкале Цельсия. Компенсатор холодного спая модуля компенсирует разъем до нуля по шкале Цельсия. Компенсатор холодного спая восстанавливает напряжение, добавляемое термопарами разъема. Температура модуля измеряется внутри. Эта температура преобразуется в значение, прибавляемое к результату преобразования датчика. Скорректированный результат преобразования датчика затем линеаризуется с помощью таблиц для термопар.

SM 1231 Thermocouple Таблица подбора термопар
Диапазоны и точность измерения для различных типов термопар поддерживаемых модулем 1231 указаны, в следующей таблице:

Тип термопар	Нижний минимум	Нижний предел номинального диапазона	Верхний предел номинального диапазона	Верхний максимум	Точность при температуре 25℃	Точность при 0℃ .. 55℃
J	-210.0℃	-150.0℃	1200.0℃	1200.0℃	±0.3℃	±0.6℃
K	-270.0℃	-200.0℃	1300.0℃	1372.0℃	±0.4℃	±1.0℃
T	-270.0℃	-200.0℃	400.0℃	400.0℃	±0.5℃	±1.0℃
E	-270.0℃	-100.0℃	1000.0℃	1000.0℃	±0.3℃	±0.6℃

R & S	-50.0℃	-200.0℃	1768.0℃	1768.0℃	±1.0℃	±2.5℃
N	-270.0℃	0.0℃	1300.0℃	1300.0℃	±1.0℃	±1.6℃
C	0.0℃	100.0℃	2315.0℃	2315.0℃	±0.7℃	±2.7℃
TXK/XK(L)	-200.0℃	-150.0℃	800.0℃	800.0℃	±0.6℃	±1.2℃
Напряжение	-32511 -94.0715mV	-27648 -80mV	27648 80mV	32511 94.071mV	±0.05%	±0.1%

[1] Внутренная ошибка холодного сплава ±1.5℃ для всех диапазонов. Добавляется к ошибке, приведенной в таблице.

Таблица выбора фильтра

Частота шумоподавления (Гц)	Время интегрирования (мс)	4-х канальный модуль, время обновления (сек)
10	100	1.205
50	20	0.245
60	16.67	0.205
400	10 [1]	

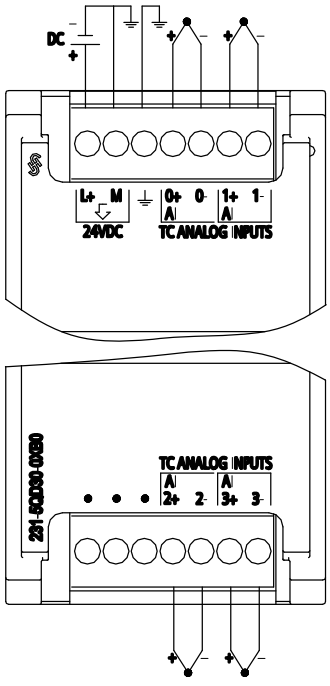
[1] Для того, чтобы достичь заявленной точности и разрешения при фильтре 400 Гц, время интегрирования устанавливается равным 10 мс. Этот фильтр также отсекает помехи частот 100 Гц и 200 Гц

Рекомендуется использовать значение времени интегрирования -100мс, при использовании термопар. При использовании меньших значений, повторяемость измерений может снижаться.

.

Примечание:
После подачи питания на модуль, происходит внутренняя калибровка аналогово-цифрового преобразователя. В течение этого времени, модуль показывет значение 32767 на каждом канале до появления настоящего значения на данном канале. Алгоритм контроллера должен учитывать это время.

SM 1231 4 AI Thermocouple Схема подключения



A.5 Сигнальные платы (SB)

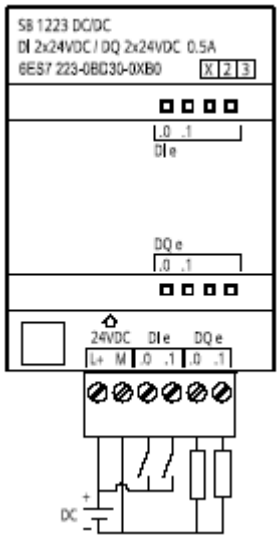
A.5.1 Технические данные SB 1223 2 X 24 VDC Input / 2 X 24 VDC Output

Технические данные цифровой сигнальной платы

Технические данные	
Модель	**SB 1223 DI 2x24VDC, DQ 2x24VDC**
Номер для заказа (MLFB)	6ES7 223-0BD30-0XB0
Общие данные	
Размеры Ш x В x Г (мм)	38 x 62 x 21
Вес	40 грамм
Рассеиваемая мощность	1,0 Вт
Потребляемый ток (шина SM)	50 мА
Потребляемый ток (24 В пост. тока)	4 мА / используемый вход
Цифровые входы	
Число входов	2
Тип	IEC Тип 1, потребляющий ток
Номинальное напряжение	24 В пост. тока при 4 мА, номинальное значение
Длительно допустимое напряжение	макс. 30 В пост. тока
Импульсное напряжение	35 В пост. тока в течение 0,5 сек.
Логический сигнал 1 (мин.)	15 В пост. тока при 2,5 мА
Логический сигнал 0 (макс.)	5 В пост. тока при 1 мА
Входные тактовые частоты HSC (макс.)	20 кГц (от 15 до 30 В пост. тока) 30 кГц (от 15 до 26 В пост. тока)
Электрическая развязка (полевая сторона относительно логики)	500 В перем. тока в течение 1 минуты
Потенциально развязанные группы	1
Постоянные времени фильтра	0,2; 0,4; 0,8; 1,6; 3,2; 6,4 и 12,8 мс, выбирается группами по 2
Число одновременно включенных входов	2
Длина кабеля (в метрах)	500 экранированный, 300 неэкранированный
Цифровые выходы	
Число выходов	2
Тип выхода	Транзисторный - MOSFET
Диапазон напряжений	от 20,4 до 28,8 В пост. тока
Логический сигнал 1 при макс. токе	мин. 20 В пост. тока
Логический сигнал 0 с нагрузкой 10 КОм	макс. 0,1 В пост. тока
Ток (макс.)	0,5 А
Ламповая нагрузка	5 Вт
Сопротивление включенного контакта	макс. 0,6 Ом
Ток утечки на выход	макс. 10 мкА
Частота генератора импульсов	макс. 20 кГц, мин. 2 Гц

Технические данные	
Модель	**SB 1223 DI 2x24VDC, DQ 2x24VDC**
Ток включения	макс. 5 A в течение 100 мс
Защита от перегрузки	Нет
Электрическая развязка (полевая сторона относительно логики)	500 В перем. тока в течение 1 минуты
Потенциально развязанные группы	1
Токи на провод	1 A
Индуктивное напряжение на клеммах	L+ минус 48 В, потеря мощности 1 Вт
Задержка включения	макс. 2 мкс из выкл. во вкл. макс. 10 мкс из вкл. в выкл.
Поведение при переходе из RUN в STOP	Последнее значение или заменяющее значение (значение по умолчанию 0)
Число одновременно включенных выходов	2
Длина кабеля (в метрах)	500 экранированный, 150 неэкранированный

Схема подключения SB 1223 2 x 24 VDC Input / 2 x 24 VDC Output

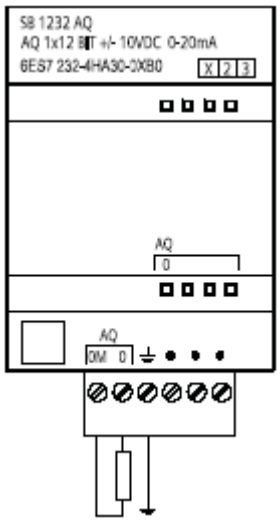


A.5.2 Технические данные SB 1232 с 1 аналоговым выходом

Технические данные аналоговой сигнальной платы

Технические данные	
Модель	**SB 1223 AQ 1x12bit**
Номер для заказа (MLFB)	6ES7 232-4HA30-0XB0
Общие данные	
Размеры Ш x В x Г (мм)	38 x 62 x 21 мм
Вес	40 грамм
Рассеиваемая мощность	1,5 Вт
Потребляемый ток (шина SM)	15 мА
Потребляемый ток (24 В пост. тока)	40 мА (без нагрузки)
Аналоговые выходы	
Число выходов	1
Тип	Напряжение или ток
Диапазон	±10 В или от 0 до 20 мА
Разрешение	Напряжение: 12 битов Ток: 11 битов
Полный диапазон (слово данных)	Напряжение: от -27,648 до 27,648 Ток: От 0 до 27,648
Точность (25℃ / от 0 до 55℃)	±0,5% / ±1% полного диапазона
Время установления (95% нового значения)	Напряжение: 300 мкс (R), 750 мкс (1 мкФ) Ток: 600 мкс (1 мГн), 2 мс (10 мГн)
Полное сопротивление нагрузки	Напряжение: ≥ 1000 Ом Ток: ≤ 600 Ом
Поведение при переходе из RUN в STOP	Последнее значение или заменяющее значение (значение по умолчанию 0)
Электрическая развязка (полевая сторона относительно логики)	Нет
Длина кабеля (в метрах)	10 метров, витой и экранированный
Диагностика	
Положительное/отрицательное переполнение	Да
Замыкание на землю (только для режима Напряжение)	Да
Обрыв провода (только для режима Ток)	Да

Схема подключения SB 1232 с 1 аналоговым выходом

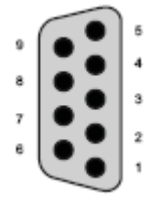


A.6 Коммуникационные модули (CM)

A.6.1 Технические данные CM 1241 RS485

Таблица A-1.Коммуникационный модуль CM 1241 RS485

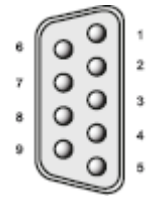
Технические данные	
Номер для заказа (MLFB)	6ES7 241-1CH30-0XB0
Размеры и вес	
Размеры	30 x 100 x 75 мм
Вес	150 грамм
Передатчик и приемник	
Диапазон синфазного напряжения	от -7 В до +12 В, 1 секунда, 3 $V_{эфф}$ постоянно
Дифференциальное выходное напряжение передатчика	мин. 2 В при R_L = 100 Ом мин. 1,5 В при R_L = 54 Ом
Оконечная нагрузка и смещение	10 КОм для +5 В на B, PROFIBUS Pin 3 10 КОм для GND на A, PROFIBUS Pin 8
Полное входное сопротивление приемника	мин. 5,4 КОм, включая оконечную нагрузку
Пороговая чувствительность приемника	мин. +/- 0,2 В, тип. гистерезис 60 мВ
Потенциальная развязка сигнала RS485 с массой сигнала RS485 с общим контактом логики CPU	500 В перем. тока, 1 минута
Длина кабеля, экранированного	макс. 1000 м
Технические данные блока питания	
Мощность потерь	1.1 Вт
из +5 В пост. тока	220 мА

Контакт	Описание	Разъем (розетка)	Контакт	Описание
1 GND	Земля логики и системы связи		6 PWR	+5 В с последовательно включенным резистором 100 Ом: Выход
2	Не подключен		7	Не подключен
3 TxD+	Сигнал B (RxD/TxD+): Вход/Выход		8 TXD-	Сигнал A (RxD/TxD-): Вход/Выход
4 RTS	Запрос на передачу (уровень TTL): Выход		9	Не подключен
5 GND	Земля логики и системы связи		SHELL	Подключение к массе

A.6.2 Технические данные CM 1241 RS232

Коммуникационный модуль CM 1241 RS232

Технические данные	
Номер для заказа (MLFB)	6ES7 241-1AH30-0XB0
Размеры и вес	
Размеры	30 x 100 x 75 мм
Вес	150 грамм
Передатчик и приемник	
Выходное напряжение передатчика	мин. +/- 5 В при R_L = 3 кОм
Выходное напряжение передатчика	макс. +/- 15 В пост. тока
Полное входное сопротивление приемника	мин. 3 кОм
Пороговая чувствительность приемника	мин. 0,8 В низкий уровень, макс. 2,4 В высокий уровень тип. гистерезис 0,5 В
Входное напряжение приемника	макс. +/- 30 В пост. тока
Потенциальная развязка сигнала RS 232 с массой сигнала RS 232 с общим контактом логики CPU	500 В перем. тока, 1 минута
Длина кабеля, экранированного	10 м макс.
Технические данные блока питания	
Мощность потерь	1,1 Вт
из +5 В пост. тока	220 мА

Контакт	**Описание**	**Разъем (штекер)**	**Контакт**	**Описание**
1 DCD	Детектирование данных и несущей: Вход		6 DSR	Набор данных готов: Вход
2 RxD	Данные, получаемые из DCE: Вход		7 RTS	Запрос на передачу: Выход
3 TxD	Данные, передаваемые в DCE: Выход		8 CTS	Готовность к приему: Вход
4 DTR	Готовность терминала к передаче данных: Выход		9 RI	Индикатор звонка (не используется)
5 GND	Земля логики		SHELL	Подключение к массе

A.7 Карты памяти SIMATIC

Технические данные карт памяти

Номер для заказа	**Емкость**
6ES7 954-8LF00-0AA0	24 Мбайта
6ES7 954-8LB00-0AA0	2 Мбайта

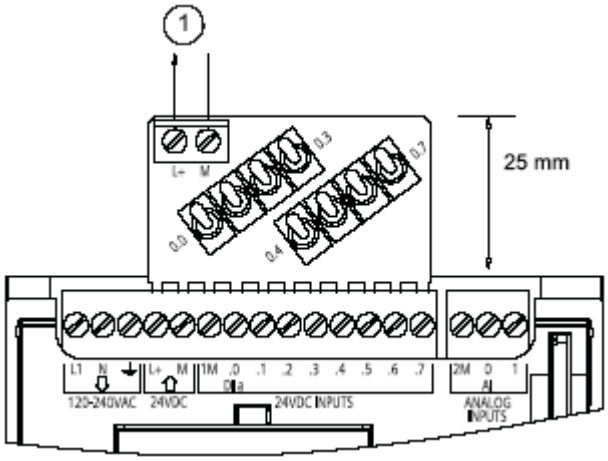
A.8 Имитаторы входов

Модель	8-позиционный имитатор	14-позиционныц имитатор
Номер для заказа (MLFB)	6ES7 274-1XF30-0XA0	6ES7 274-1XH30-0XA0
Размеры Ш x В x Г (мм)	43 x 35 x 23	67 x 35 x 23
Вес	20 грамм	30 грамм
Точки ввода	8	14
Используется с CPU	CPU 1211C, CPU 1212C	CPU 1214C

ПРЕДУПРЕЖДЕНИЕ

Эти имитаторы входов непригодны для использования во взрывоопасных производственных помещениях класса I, раздел 2, и класса I, зона 2. При использовании в помещениях класса I, раздел 2, и класса I, зона 2, выключатели могут привести к появлению искры или возникновению взрыва.

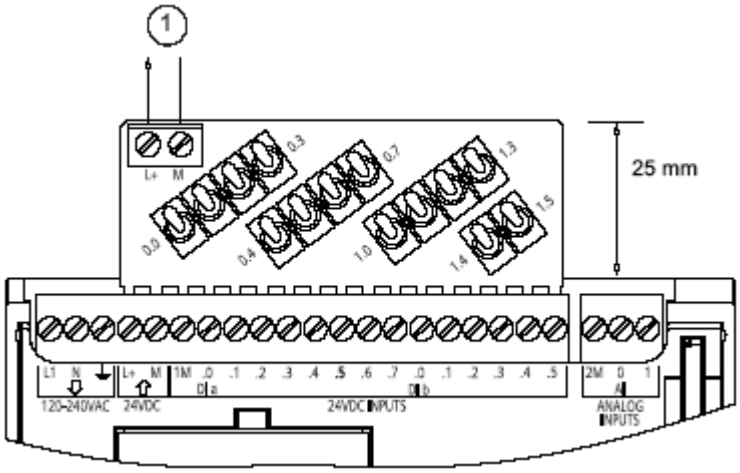
8-позиционный имитатор



① Питание датчиков 24 В пост. тока

6ES7 274-1XF30-0XA0

14-позиционныц имитатор



① Питание датчиков 24 В пост. тока

6ES7 274-1XH30-0XA0

A.9 Кабель для расширения ввода/вывода

Технические данные	
Номер для заказа (MLFB)	6ES7 290-6AA30-0XA0
Длина кабеля	2 м
Вес	200 г

Кабель для расширения ввода/вывода имеет розетку и штекер.

1. Вставьте штекер в розетку шинного соединителя на правой стороне сигнального модуля.
2. Вставьте розетку в ответный разъем шинного соединителя на левой стороне сигнального модуля.
   - Вдвиньте крючкообразный выступ розетки в корпус у шинного соединителя
   - Вожмите розетку в ответный разъем шинного соединителя.

B

Расчет баланса мощностей

У CPU имеется внутренний источник питания, который обеспечивает электропитанием как сам CPU, так и модули расширения и других потребителей напряжения 24 В пост. тока.

Имеется три типа модулей расширения:

- Сигнальные модули (SM) устанавливаются с правой стороны от CPU. Каждый CPU допускает подключение максимально возможного числа сигнальных модулей независимо от баланса мощностей.
  - CPU 1214 допускает 8 сигнальных модулей
  - CPU 1212 допускает 2 сигнальных модуля
  - CPU 1211 не допускает сигнальных модулей
- Коммуникационные модули (CM) устанавливаются с левой стороны от CPU. К каждому CPU можно подключить не более 3 коммуникационных модулей независимо от баланса мощностей.
- Сигнальные платы (SB) устанавливаются сверху CPU. На каждом CPU можно установить не более 1 сигнальной платы.

С помощью следующей информации вы можете рассчитать, какую мощность CPU может предоставить в распоряжение вашей конфигурации.

Каждый CPU поставляет питание напряжением 5 В пост. тока и 24 В пост. тока:

- CPU поставляет питание напряжением 5 В пост. тока для модулей расширения, если такие модули подключены. Если потребности в питании 5 В пост. тока превышают мощность, которую CPU может обеспечить, вы должны удалять модули расширения до тех пор, пока не будет соблюден баланс мощностей. У каждого CPU имеется источник питания датчиков напряжением 24 В пост. тока, который поставляет его для локальных входов и катушек реле модулей расширения. Если потребности в питании 24 В пост. тока превышают мощность, которую CPU может обеспечить, вы можете добавить внешний источник питания 24 В пост. тока, чтобы обеспечить этим питание модули расширения. Источник питания 24 В пост. тока вы должны вручную подключить к входам и катушкам реле.

ПРЕДУПРЕЖДЕНИЕ

Подключение внешнего источника питания 24 В пост. тока параллельно с источником питания датчиков может привести к конфликту этих двух источников, так как каждый из них стремится установить свой собственный уровень выходного напряжения.

Результатом этого конфликта может быть сокращение срока службы или немедленный выход из строя одного или обоих источников питания с последующим непредсказуемым поведением ПЛК. Такое непредсказуемое поведение может привести к гибели людей, тяжким телесным повреждениям и/или имущественному ущербу.

Находящийся в CPU источник питания постоянного тока для датчиков и любой внешний источник питания должны подключаться к разным точкам. При этом между источниками питания допускается не более одного соединения.

Некоторые из входных портов питания 24 В в системе ПЛК соединены между собой, причем общий провод логики соединяет несколько клемм M. Вход источника питания 24 В на CPU, вход питания катушек на SM и не имеющий потенциальной развязки вход питания аналогового модуля являются примерами цепей, которые соединяются друг с другом, если в технических данных они обозначены, как не имеющие потенциальной развязки. Все не имеющие потенциальной развязки клеммы M должны быть присоединены к одному и тому же внешнему опорному потенциалу.

⚠ ПРЕДУПРЕЖДЕНИЕ

Подключение не имеющих потенциальной развязки клемм M к разным опорным потенциалам вызовет протекание непредусмотренных токов, которые могут вызвать повреждение или непредсказуемое поведение ПЛК и подключенного оборудования.

Такое повреждение или непредсказуемое поведение могут привести к гибели людей, тяжким телесным повреждениям и/или имущественному ущербу.

Всегда обеспечивайте подключение всех не имеющих потенциальной развязки клемм M в системе ПЛК к одному и тому же опорному потенциалу.

Информацию о возможностях источника питания CPU и потребной мощности сигнальных модулей вы найдете в технических данных (стр. 329).

Указание

Если баланс мощностей CPU нарушен, то, возможно, вы не сможете подключить максимальное количество модулей, разрешенное для вашего CPU.

B.2 Пример расчета потребности в мощности

В следующем примере рассчитаны потребности в мощности для ПЛК, содержащего один CPU 1214C AC/DC/Relay, 3 SM 1223 с 8 входами постоянного тока и 8 релейными выходами и одного SM 1221 с 8 входами постоянного тока. В этом примере имеется всего 46 входов и 34 выхода.

Указание

CPU уже выделил мощность, необходимую для управления катушками внутренних реле. У вас нет необходимости включать потребности в мощности катушек внутренних реле в расчет баланса мощностей.

CPU в этом примере предоставляет достаточно мощности питания 5 В пост. тока для сигнальных модулей, но мощности источника питания датчиков 24 В пост. тока не хватает для всех входов и катушек реле модулей расширения. Входам и выходам необходимо 448 мА, а CPU предоставляет только 400 мА. Эта установка требует дополнительного источника питания не менее 48 мА при 24 В пост. тока для управления всеми включенными входами и выходами 24 В пост. тока.

Баланс мощностей CPU	**5 В пост. тока**	**24 В пост. тока**
CPU 1214C AC/DC/Relay	1600 мА	400 мА
	Минус	
Потребности системы	**5 В пост. тока**	**24 В пост. тока**
CPU 1214C, 14 входов	-	14 * 4 мА = 56 мА
3 SM 1223, напряжение 5 В	3 * 145 мА = 435 мА	-
1 SM 1221, напряжение 5 В	1 * 105 мА = 105 мА	-
3 SM 1223, 8 входов каждый	-	3 * 8 * 4 мА = 96 мА
3 SM 1223, 8 катушек реле каждый	-	3 * 8 * 11 мА = 264 мА
1 SM 1221, 8 входов	-	8 * 4 мА = 32 мА
Общие потребности	540 мА	448 мА
	Равно	
Баланс токов	**5 В пост. тока**	**24 В пост. тока**
Общий баланс токов	1060 мА	(48 мА)

B.3 Расчет вашей потребности в мощности

Для определения количества энергии (или тока), которое CPU S7-1200 может предоставить для вашей конфигурации, используйте следующую таблицу. Дальнейшую информацию вы найдете в технических данных (стр. 329) для имеющейся в распоряжении мощности вашей модели CPU и потребностях в мощности ваших сигнальных модулей.

Баланс мощностей CPU	**5 В пост. тока**	**24 В пост. тока**
	Минус	
Потребности системы	**5 В пост. тока**	**24 В пост. тока**
Общие потребности		
	Равно	
Баланс токов	**5 В пост. тока**	**24 В пост. тока**
Общий баланс токов		

C

Номера для заказа

CPU		Номер для заказа
CPU 1211C	CPU 1211C DC/DC/DC	6ES7 211-1AD30-0XB0
	CPU 1211C AC/DC/Relay	6ES7 211-1BD30-0XB0
	CPU 1211C DC/DC/Relay	6ES7 211-1HD30-0XB0
CPU 1212C	CPU 1212C DC/DC/DC	6ES7 212-1AD30-0XB0
	CPU 1212C AC/DC/Relay	6ES7 212-1BD30-0XB0
	CPU 1212C DC/DC/Relay	6ES7 212-1HD30-0XB0
CPU 1214C	CPU 1214C DC/DC/DC	6ES7 214-1AE30-0XB0
	CPU 1214C AC/DC/Relay	6ES7 214-1BE30-0XB0
	CPU 1214C DC/DC/Relay	6ES7 214-1HE30-0XB0

Сигнальные модули, коммуникационные модули, и сигнальные платы		Номер для заказа
Сигнальные модули	SM 1221 8 x 24 VDC Input	6ES7 221-1BF30-0XB0
	SM 1221 16 x 24 VDC Input	6ES7 221-1BH30-0XB0
	SM 1222 8 x 24 VDC Output	6ES7 222-1BF30-0XB0
	SM 1222 16 x 24 VDC Output	6ES7 222-1BH30-0XB0
	SM 1222 8 x Relay Output	6ES7 222-1HF30-0XB0
	SM 1222 16 x Relay Output	6ES7 222-1HH30-0XB0
	SM 1223 8 x 24 VDC Input / 8 x 24 VDC Output	6ES7 223-1BH30-0XB0
	SM 1223 16 x 24 VDC Input / 16 x 24 VDC Output	6ES7 223-1BL30-0XB0
	SM 1223 8 x 24 VDC Input / 8 x Relay Output	6ES7 223-1PH30-0XB0
	SM 1223 16 x 24 VDC Input / 16 x Relay Output	6ES7 223-1PL30-0XB0
	SM 1231 4 x Analog Input	6ES7 231-4HD30-0XB0
	SM 1231 8 x Analog Input	6ES7 231-4HF30-0XB0
	SM 1231 4 x RTD	6ES7 231-5PD30-0XB0
	SM 1231 8 x RTD	6ES7 231-5PF30-0XB0
	SM 1231 4 x TC	6ES7 231-5QD30-0XB0
	SM 1231 8 x TC	6ES7 231-5QF30-0XB0
	SM 1232 2 x Analog Output	6ES7 232-4HB30-0XB0
	SM 1232 4 x Analog Output	6ES7 232-4HD30-0XB0
	SM 1234 4 x Analog Input / 2 x Analog Output	6ES7 234-4HE30-0XB0
Коммуникацион-ные модули	CM 1241 RS232	6ES7 241-1AH30-0XB0
	CM 1241 RS485	6ES7 241-1CH30-0XB0
Сигнальные платы	SB 1223 2 x 24 VDC Input / 2 x 24 VDC Output	6ES7 223-0BD30-0XB0
	SB 1232 1 Analog Output	6ES7 232-4HA30-0XB0

Устройства человеко-машинного интерфейса	**Номер для заказа**
KTP400 Basic (Монохромный, PN)	6AV6 647-0AA11-3AX0
KTP600 Basic (Монохромный, PN)	6AV6 647-0AB11-3AX0
KTP600 Basic (Цветной, PN)	6AV6 647-0AD11-3AX0
KTP1000 Basic (Цветной, PN)	6AV6 647-0AF11-3AX0
TP1500 Basic (Цветной, PN)	6AV6 647-0AG11-3AX0

Пакет программирования	**Номер для заказа**
STEP 7 Basic v10.5	6ES7 822-0AA0-0YA0

Карты памяти, другая аппаратура и запасные части		**Номер для заказа**
Карты памяти	SIMATIC MC 2 MB	6ES7 954-8LB00-0AA0
	SIMATIC MC 24 MB	6ES7 954-8LF00-0AA0
Другая аппаратура	Блок питания PSU 1200	6EP1 332-1SH71
	Коммутатор CSM 1277 Ethernet - 4 порта	6GK7 277-1AA00-0AA0
	Имитатор (1214C/1211C - 8 позиций)	6ES7 274-1XF30-0XA0
	Имитатор (1214C - 14 позиций)	6ES7 274-1XH30-0XA0
	Кабель расширения ввода/вывода, 2 м	6ES7 290-6AA30-0XA0
Запасные части	Клеммный блок, 7 клемм, луженые	6ES7 292-1AG30-0XA0
	Клеммный блок, 8 клемм, луженые (4 на упаковку)	6ES7 292-1AH30-0XA0
	Клеммный блок, 11 клемм, луженые (4 на упаковку)	6ES7 292-1AL30-0XA0
	Клеммный блок, 12 клемм, луженые (4 на упаковку)	6ES7 292-1AM30-0XA0
	Клеммный блок, 14 клемм, луженые (4 на упаковку)	6ES7 292-1AP30-0XA0
	Клеммный блок, 20 клемм, луженые (4 на упаковку)	6ES7 292-1AV30-0XA0
	Клеммный блок, 3 клемм, позолоченные (4 на упаковку)	6ES7 292-1BC0-0XA0
	Клеммный блок, 6 клемм, позолоченные (4 на упаковку)	6ES7 292-1BF30-0XA0
	Клеммный блок, 7 клемм, позолоченные (4 на упаковку)	6ES7 292-1BG30-0XA0
	Клеммный блок, 11 клемм, позолоченные (4 на упаковку)	6ES7 292-1BL30-0XA0

Документация	**Номер для заказа**
Программируемый контроллер S7-1200, системное руководство	
• немецкий	• 6ES7 298-8FA30-8AH0
• английский	• 6ES7 298-8FA30-8BH0
• французский	• 6ES7 298-8FA30-8CH0
• испанский	• 6ES7 298-8FA30-8DH0
• итальянский	• 6ES7 298-8FA30-8EH0
• китайский	• 6ES7 298-8FA30-8FH0
S7-1200 Easy Book [Простое введение]	
• немецкий	• 6ES7 298-8FA30-8AQ0
• английский	• 6ES7 298-8FA30-8BQ0
• французский	• 6ES7 298-8FA30-8CQ0
• испанский	• 6ES7 298-8FA30-8DQ0
• итальянский	• 6ES7 298-8FA30-8EQ0
• китайский	• 6ES7 298-8FA30-8FQ0

Предметный указатель

А

Адреса в памяти, 58, 60
Аппаратное управление потоком, 282
Архитектура опроса, 291
Архитектура опроса master-устройства, 291
Архитектура опроса slave-устройства, 291
Арифметические команды, 127
Арифметические операции с плавающей точкой, 134

Б

Баланс мощностей, 23, 373
  пример, 375, 376
Безопасность
  кодовый блок, 99
  утерянный пароль, 58
  CPU, 57
Библиотека протокола USS, 210
Блок
  виды, 39
Блок данных
  глобальный блок данных, 58, 95
  организационный блок(OB), 92
  экземплярный блок данных, 58
Блок данных (DB), 95
Блок передачи (T-блок), 267
Блоки
  блоки данных (DB), 39
  организационные блоки (OB), 39, 45
  функции (FC), 39
  функциональные блоки (FB), 39

В

Включаемый сигнал RTS, 283
Возвращаемое значение (RET), 145
Возвращаемые значения команд PtP, 312
Время ожидания, 282
Время цикла, 50, 51
Всплывающие подсказки, 17
Вставка устройства
  неопределенный CPU, 70
Входы/выходы
  адресация, 63
  индикаторы состояния аналоговых входов/выходов, 318
  индикаторы состояния цифровых входов/выходов, 318
  индуктивные нагрузки, 37
Вывод последовательности импульсов (PTO), 206
Вызов блока
  вызов в качестве одно- или многоэкземплярного блока, 112
  основы, 39
Выходные параметры, 94

Г

Глобальная библиотека
  USS, 210
Глобальный блок данных, 58, 95
Горячая линия, 3
Граничное значение, 133

Д

Двоичная логика, 103
Двухточечная связь, 279
Диагностический буфер, 54, 322
Длина
  сообщения, 288
Длина сообщения, 288
Длина m, 289
Длина n, 289
Добавление устройства
  неопределенный CPU, 79
Документация, 17
Допуск к эксплуатации ATEX, 330
Допуск к эксплуатации CE, 329
Допуск к эксплуатации C-Tick, 331
Допуск к эксплуатации cULus, 330
Доступ к онлайновой информационной системе, 15

З

Загрузка в устройство, 261
Задержка включения RTS, 284
Задержка выключения RTS, 284
Зазор при монтаже, 22
Защита ноу-хау, 99
Защита от копирования, 99
Защита паролем
  кодовый блок, 99
  пустая передаточная карта, 58

утерянный пароль, 58
CPU, 57
Значение времени, установка параметров онлайнового CPU, 320
Значения, возвращаемые во время приема, 314

И

Имитаторы входов, 371
Импульсный таймер (TP), 112
Индикаторы состояния аналоговых входов и выходов, 318
Индикаторы состояния цифровых входов и выходов, 318
Индуктивные нагрузки, 37
Интерфейс PROFINET
свойства адреса Ethernet, 85, 258
свойства синхронизации времени, 277
Информационная система, 16
печать, 19
освобождение окна, 18
отображение содержания и предметного указателя, 18
расширение, 18
Исполнение программы, 39
Исполнение событий, 46
Использование одного FB с несколькими DB
пример, 95

К

Карта памяти
конфигурирование параметров запуска, 70
передаточная карта, 71
программная карта, 73
пустая передаточная карта, 58
утерянный пароль, 58
Класс защиты, 333
Класс приоритета
обзор, 45
Клеммный блок
удаление и повторная установка, 33
установка, 33
Кодовые блоки, 90
Кодовый блок
защита ноу-хау, 99
DB (блок данных), 95
FB (функциональный блок), 94
FC (функция), 93
Коды состояния USS, 221
Команда активизации прерываний EN_AIRT, 203
Команда активизации скоростного счетчика (HSC), 119
Команда блочной передачи (MOVE_BLK), 136
Команда выбора (SEL), 148
Команда вычитания (SUB), 127
Команда деактивизации прерываний DIS_AIRT, 191
Команда декодирования (DECO), 147
Команда деления DIV, 127
Команда задержки включения (TON), 112
Команда задержки включения с запоминанием (TONR), 112
Команда задержки выключения (TOF), 112
Команда заполнения (FILL_BLK), 138
Команда инвертирования (INV), 146
Команда кодирования (ENCO), 147
Команда масштабирования (SCALE_X), 143
Команда мультиплексирования (MUX), 148
Команда нахождения вне заданного диапазона, 126
Команда нахождения в заданном диапазоне, 126
Команда непрерываемого заполнения (UFILL_BLK), 138
Команда непрерываемой передачи (UMOVE_BLK), 136
Команда нормализации (NORM), 143
Команда обнаружения нарастающего (положительного) фронта, 109
Команда обнаружения падающего (отрицательного) фронта, 109
Команда образования абсолютного значения (ABS), 131
Команда округления, 142
Команда округления до ближайшего большего целого числа (CEIL), 142
Команда округления до ближайшего меньшего целого числа (FLOOR), 142
Команда перехода (JMP), 144
Команда преобразования вещественного числа в целое (TRUNC), 142
Команда прерывания с задержкой CAN_DINT, 200
Команда прерывания с задержкой SRT_DINT, 200
Команда сброса, 106
Команда сброса таймера (RT), 112
Команда сдвига влево (SHL), 150
Команда сдвига вправо (SHR), 150
Команда сложения (ADD), 127
Команда умножения (MUL), 127
Команда установки, 106
Команда циклического сдвига влево (ROL), 151
Команда циклического сдвига вправо (ROR), 151
Команда AND, 146
Команда ATTACH, 197
Команда CTRL_PWM, 206

Команда DEC (уменьшение на 1), 130
Команда DETACH, 197
Команда INC (увеличение на 1), 130
Команда JMPN, 144
Команда MAX (максимум), 132
Команда MIN (минимум), 132
Команда MOD (modulo), 128
Команда MOVE, 136
Команда NEG (отрицание), 129
Команда Not OK, 126
Команда OK, 126
Команда OR, 146
Команда PID_Compact, 203
Команда PORT_CFG (конфигурирование порта), 294
Команда RCV_CFG (конфигурирование приема), 298
Команда RCV_PTP (прием данных двухточечного соединения), 308
Команда RCV_RST (сброс приемника), 309
Команда RE_TRIGR, 176
Команда RT (сброс таймера), 112
Команда S_CONV, 157
Команда SEND_CFG (конфигурирование передачи), 296
Команда SEND_PtP (передача данных двухточечного соединения), 305
Команда SGN_GET (опрос сигналов RS232), 310
Команда SGN_SET (установка сигналов RS232), 311
Команда STP (остановка цикла ПЛК), 177
Команда STRG_VAL, 157
Команда SWAP (обмен байтов), 140
Команда T_ADD, 153
Команда T_CONV, 153
Команда T_DIFF, 153
Команда T_SUB, 153
Команда TCON, 188
Команда TDISCON, 188
Команда TRCV, 188
Команда TRCV_C, 181, 271
Команда TSEND, 188
Команда TSEND_C, 181, 267
Команда USS_DRV, 213
Команда USS_PORT, 216
Команда USS_RPM, 217
Команда USS_WPM, 219
Команда VAL_STRG, 157
Команда XOR (исключающее ИЛИ), 146
Команды
  арифметика с плавающей точкой, 134
  блочная передача (MOVE_BLK), 136
  возвращаемое значение (RET), 145
  время, 153
  выбор (SEL), 148
  вычитание (SUB), 127
  граничное значение, 133
  дата, 153
  двоичная логика, 103
  декодирование (DECO), 147
  деление (DIV), 127
  заполнение (FILL_BLK), 138
  значение в строку: S_CONV, 157
  значение в строку: VAL_STRG, 157
  инвертирование (INV), 146
  календарь, 153
  кодирование (ENCO), 147
  коды состояния USS, 221
  масштабирование (SCALE_X), 143
  метка, 144
  мультиплексирование (MUX), 148
  нарастающий (положительный) фронт, 109
  непрерываемое заполнение (UFILL_BLK), 138
  непрерываемая передача (UMOVE_BLK), 136
  нормализация (NORM), 143
  обмен байтов, 140
  образование абсолютного значения (ABS), 131
  округления до ближайшего большего целого числа, 142
  округление, 142
  округление до ближайшего меньшего целого числа (FLOOR), 142
  округление до целого (TRUNC), 142
  падающий (отрицательный) фронт, 109
  передача, 136
  переход (JMP), 144
  преобразование, 141
  прерывание: ATTACH, 197
  прерывание: CAN_DINT, 300
  прерывание: DETACH, 197
  прерывание: DIS_AIRT, 203
  прерывание: EN_AIRT, 203
  прерывание: SRT_DINT, 200
  сброс, 106
  сдвиг влево (SHL), 150
  сдвиг вправо (SHR), 150
  скоростной счетчик (HSC), 119
  сложение (ADD), 127
  сравнение, 125
  строка в значение: S_CONV, 157
  строка в значение: STRG_VAL, 157
  счетчик, 116
  таймер, 112
  таймер: RT (сброс таймера), 112

таймер: TOF (задержка выключения), 112
таймер: TON (задержка включения), 112
таймер: TONR (задержка включения с запоминанием), 112
таймер: TP (импульс), 112
умножение (MUL), 127
установка, 106
циклический сдвиг влево (ROL), 151
циклический сдвиг вправо (ROR), 151
часы, 155
часы: запись системного времени (WR_SYS_T), 155
часы: считывание местного времени (RD_LOC_T), 155
часы: считывание системного времени (RD_SYS_T), 155
AND, 146
CTRL_PWM, 206
DEC (уменьшение на 1), 130
GET_ERROR, 181
INC (увеличение на 1), 130
IN_RANGE, 126
MAX (максимум), 132
MIN (минимум), 132
MOD (modulo), 128
NEG (отрицание), 129
Not OK, 126
OK, 126
OR, 147
OUT_RANGE, 126
PID_Compact, 203
PORT_CFG (конфигурирование порта), 284
RCV_CFG (конфигурирование приема), 298
RCV_PtP (прием данных через двухточечное соединение), 308
RCV_RST (сброс приемника), 309
RE_TRIGR, 50, 176
SEND_CFG (конфигурирование передачи), 296
SEND_PTP (передача данных через двухточечное соединение), 282
SGN_GET (опрос сигналов RS232), 310
SGN_SET (установка сигналов RS232), 311
STP (остановка цикла ПЛК), 177
T_ADD, 153
T_CONV, 153
T_DIFF, 153
T_SUB, 153
TCON, 188
TDISCON, 188
TRCV, 188
TRCV_C, 181, 271
TSEND, 188
TSEND_C, 181, 267
USS_DRV, 213
USS_PORT, 216
USS_RPM, 217
USS_WPM, 219
XOR (исключающее ИЛИ), 146
Команды для календаря, 153
Команды для расчетов, связанных с временем, 153
Команды для часов, 153
запись системного времени (WR_SYS_T), 155
считывание местного времени (RD_LOC_T), 155
считывание системного времени (RD_SYS_T), 155
Команды преобразования, 141
Команды преобразования значения в строку символов, 157
Команды сравнения, 125
Команды счета, 116
Команды Ethernet
TCON, 188
TDISCON, 188
TRCV, 188
TRCV_C, 181
TSEND, 188
TSEND_C, 181
Коммуникации
аппаратное соединение, 265
архитектура опроса, 291
библиотеки, 279
нагрузка, 51
параметры передачи и приема, 284
управление потоком, 282
IP-адрес, 84, 252
Коммуникации на основе TCP/IP, 249
Коммуникационные интерфейсы
конфигурирование, 281
программирование, 290
Коммуникационные модули
RS232 и RS485, 259
Коммуникационный модуль (CM), 292
добавление модулей, 81
добавление нового устройства, 78
конфигурация устройств, 77
обзор, 14
прием данных, 308
потребности в мощности, 373
технические данные, 369
удаление, 30
установка, 30
Коммуникационный модуль (CM), библиотека USS, 210
Конец сообщения, 288

Контактная информация, 3
Контекстно-чувствительная помощь, 17
Контроль времени цикла, 176
Контроль времени цикла, онлайновый CPU, 321
Контроль использования памяти, онлайновый CPU, 321
Контроль программы, 102
Контроль четности, 282
Конфигурация устройств, 77, 252
  выявление, 79
  добавление модулей, 81
  добавление нового устройства, 78
  конфигурирование модулей, 81
  конфигурирование CPU, 80
  порт Ethernet, 84, 257
  создание сетевого соединения, 83
  PROFINET, 84, 257
Конфигурирование
  времени цикла, 51
  коммуникационных интерфейсов, 281
  логического соединения устройства человеко-машинного интерфейса с CPU, 264
  обмена данными между ПЛК, 265
  параметров запуска, 41, 70
  порта Industrial Ethernet, 84, 257
  портов, 281
  принимаемого сообщения, 264
  HSC (скоростного счетчика), 124
  IP-адреса, 84, 257
  PROFINET, 82, 239
Конфигурирование аппаратуры, 77
  добавление модулей, 81
  добавление нового устройства, 78
  конфигурирование модулей, 81
  конфигурирование CPU, 80
  порт Ethernet, 84, 287
  создание сетевого соединения, 81
  PROFINET, 84, 287
Конфигурирование команды TRCV_C, 272
Конфигурирование команды TSEND_C, 249
Конфигурирование параметров
  модулей, 82
  порта Ethernet, 84, 257
  CPU, 80
  PROFINET, 84, 257
Конфигурирование параметров передачи, 268
Конфигурирование параметров приема, 272
Конфигурирование параметров связи
  передача, 268
  прием, 272
Конфигурирование портов, 281
  команды, 290
Конфигурирование передаваемых сообщений, 284
Конфигурирование принимаемых сообщений, 285
Конфигурирование сообщений
  команды, 290
  передача, 284
  прием, 285

Л

Ламповые нагрузки, 38
Латентный период, 48
Линейная программа, 89

М

Максимальная длина сообщения, 288
Маска подсети, 84, 257
Метка перехода, 144
Модули
  коммуникационный модуль (CM), 14
  конфигурирование параметров, 82
  сигнальная плата (SB), 13
  сигнальный модуль (SM), 14
  сравнительная таблица, 12
Модули ввода/вывода
  таблицы наблюдения, 323
Монтаж
  заземление, 35
  зазор, 22
  индуктивные нагрузки, 37
  клеммный блок, 33
  коммуникационный модуль (CM), 30
  ламповые нагрузки, 38
  монтажные размеры, 24
  обзор, 21
  потенциальная развязка, 35
  сигнальная плата (SB), 32
  сигнальный модуль (SM), 28
  указания, 21
  указания по подключению, 34, 38
  CPU, 26
Морской допуск к эксплуатации, 331

Н

Напряжение аналоговых сигнальных модулей, 361
Начало сообщения, 286
Неопределенный CPU, 79
Номинальные напряжения, 334

О

Обмен данными через Ethernet, 188
Общие ошибки параметризации PtP, 312
Общие технические данные, 329
Окружающая среда
  промышленная, 331
Онлайновая помощь, 17
  печать, 19
  освобождение окна, 18
  отображение содержания и предметного указателя, 18
  расширение, 18
Онлайновый CPU, 319
  контроль времени цикла, 321
  контроль использования памяти, 321
  панель оператора, 320
Онлайн, переход в режим онлайн, 319
Организационный блок
  вызов, 45
  классы приоритета, 45
  конфигурирование режима функционирования, 93
  несколько циклических OB, 93
  обработка, 92
  создание, 93
  функция, 45
Организация очередей, 46
Освобождение окна помощи, 18
Отображение содержания и предметного указателя (онлайновая информационная система), 18
Ошибки
  диагностические ошибки, 49
  команды PtP, 312
  ошибки времени, 48
Ошибки во время передачи, 314
Ошибки конфигурирования передачи, 313
Ошибки конфигурирования порта, 312
Ошибки конфигурирования приема, 313
Ошибки обработки сигналов, 313

П

Память
  временная память, 62
  загрузочная память, 52
  рабочая память, 52
  системная память, 55
  сохраняемая память, 52
  тактовые меркеры, 53
  I (образ процесса на входах), 60
  L (локальная память), 58
  M (битовая память), 61
  Q (образ процесса на выходах), 61
Параметризация, 94
Параметры запуска, 41, 70
Пароль, 58
Пауза, 284, 286
Первые шаги
  всплывающие подсказки, 17
  информационная система, 17
  каскадные всплывающие подсказки, 17
  контекстно-чувствительная помощь, 17
  онлайновая помощь, 17
  подсказка, 17
  портальное и проектное представления, 16
Передаточная карта, 71
  конфигурирование параметров запуска, 70
  пустая передаточная карта, 58
  утерянный пароль, 58
Передача данных, инициирование, 305
Переменный ток (AC)
  индуктивные нагрузки, 37
Переход из RUN в STOP, 56
ПЛК
  использование блоков, 88
  обзор, 11
  разработка системы, 87
Поддержка, 3
Позиция символа
  длина сообщения, 289
Поддержка пользователя, 3
Помощь, 16
  печать, 19
  освобождение окна, 18
  отображение содержания и предметного указателя, 18
  расширение, 18
Портальное представление, 15
  добавление модулей, 81
  добавление нового устройства, 78
  конфигурирование модулей, 82
  конфигурирование порта Ethernet, 84, 257
  конфигурирование CPU, 80
  PROFINET, 84, 257
Портал TIA
  добавление модулей, 81
  добавление нового устройства, 78
  конфигурация устройств, 77
  конфигурирование модулей, 82
  конфигурирование CPU, 80
  портальное представление, 16
  порт Ethernet, 84, 257
  проектное представление, 16

создание сетевого соединения, 83
установка, 15
PROFINET, 84, 257
Последовательность символов
конец сообщения, 288
начало сообщения, 286
Последовательный обмен данными, 279
Постоянный ток
индуктивные нагрузки, 37
Потребности в мощности
расчет, 375, 376
Преобразование строки символов в значение, 157
Прерывания
обзор, 45
Приоритеты обработки, 46
Программирование
команды PtP, 290
линейное, 89
неопределенный CPU, 79
поток сигнала (EN и ENO), 98
структурированная, 89
FBD (функциональная блок-схема, функциональный план), 98
LAD (цепная логика, контактный план), 97
Программирование обмена данными через PtP, 290
Программная карта
конфигурирование параметров запуска, 73
Программное управление потоком, 283
Проект
защита кодового блока, 99
ограничение доступа CPU, 57
передаточная карта, 71
программная карта, 73
пустая передаточная карта, 58
утерянный пароль, 58
Проектирование системы с ПЛК, 87
Проектное представление, 16
добавление модулей, 81
добавление нового устройства, 78
конфигурирование модулей, 82
конфигурирование параметров CPU, 80
конфигурирование порта Ethernet, 84, 257
конфигурация устройств, 77
создание сетевого соединения, 83
PROFINET, 84, 257
Промежуток между символами, 288
Простаивающая линия, 284
Протокол
связи, 279
свободно программируемая связь, 279
Профильная шина, 25

P

Распечатка тем помощи, 19
Расширение возможностей S7-1200, 12
Режим RUN, 42, 44

C

Светодиодные индикаторы, 292, 317
Связь через сеть, 251
Сертификат FM, 330
удаление, 32
Сигнальная плата (SB)
добавление модулей, 81
добавление нового устройства, 78
конфигурация устройств, 77
обзор, 13
потребности в мощности, 373
сравнительная таблица, 13
удаление, 32
установка, 32
Сигнальные модули
технические данные SM 1221, 351
технические данные SM 1222, 353
технические данные SM 1223, 355
Сигнальный модуль (SM)
добавление модулей, 81
добавление нового устройства, 78
конфигурация устройств, 77
обзор, 14
потребности в мощности, 373
сравнительная таблица, 13
удаление, 28
установка, 28
Символ конца сообщения, 288
Символ начала сообщения, 286
Синхронизирующий сетевой протокол (NTP), 277
Скоростной счетчик, 121
Скорость передачи, 281
Создание сетевого соединения, 83
Состояние STOP, 42, 325
Сравнительная таблица
модели CPU, 12
устройства человеко-машинного интерфейса, 20
Сравнительная таблица модулей, 13
Срок службы реле, 334
Стоповые биты, 282
Структура программы, 90
Структурное программирование, 89, 90
Схемы соединений
сигнальный модуль SM 1221, 352
сигнальный модуль SM 1222, 354

сигнальный модуль SM 1223, 357
CPU 1211C, 339
CPU 1212C, 344
CPU 1214C, 349
SB 1223, 366
SB 1232, 368
SM 1231, 1232, 1234, 363
Счетчик
скоростной (HSC), 121
скоростной (HSC): конфигурирование, 124

T

Таблицы наблюдения, 102, 323
Таймерная команда TOF (задержка выключения), 112
Таймерная команда TON (задержка включения), 112
Таймерная команда TONR (задержка включения с запоминанием), 112
Таймерная команда TP (импульс), 112
Таймерные команды, 112
Тестирование программы, 102
Техническая поддержка, 3
Технические данные, 329
Технические данные
аналоговые сигнальные модули, 358
допуск к эксплуатации ATEX, 330
допуск к эксплуатации CE, 329
допуск к эксплуатации C-Tick, 331
допуск к эксплуатации cULus, 330
защита, 307
имитаторы входов, 371
карты памяти, 343
коммуникационный модуль CM 1241 RS232, 370
коммуникационный модуль CM 1241 RS485, 369
морской допуск к эксплуатации, 331
напряжение аналоговых сигнальных модулей, 360
номинальные напряжения, 308
общие, 329
промышленная среда, 331
сертификат FM, 330
сигнальный модуль SM 1221, 351
сигнальный модуль SM 1222, 353
сигнальный модуль SM 1223, 355
срок службы реле, 334
схема подключения SM 1221, 352
схема подключения SM 1222, 354
схема подключения SM 1223, 357
схемы соединений: SM 1231, 1232, 1234, 363
условия окружающей среды, 333
цифровые сигнальные платы (SB), 364
электромагнитная совместимость (ЭМС), 332
CPU 1211C, 335
CPU 1212C, 340
CPU 1214C, 345
SB 1223, 364
SB 1223, 367
Технические данные аналоговых сигнальных модулей, 358
Технические данные карт памяти, 370
Технические данные цифровой сигнальной платы (SB), 364
Технические данные CM 1241 RS232, 370
Технические данные CM 1241 RS485, 369
Тип данных DTL (Data and Time Long), 67
Тип данных STRING, 65
Типы данных, 64
массивы, 66
DTL, 67
STRING, 65

У

Указания
заземление, 35
индуктивные нагрузки, 37
ламповые нагрузки, 38
монтаж, 21
последовательность монтажа, 25
по подключению, 34, 36
электрическая развязка, 35
Указания для потенциальной развязки, 35
Указания по подключению
заземление, 36
предпосылки, 34
Управление потоком, 282
конфигурирование, 282
Уровень защиты
кодовый блок, 99
утерянный пароль, 58
CPU, 57
Условия конца сообщения, 288
Условия начала сообщения, 286
Условия окружающей среды, 333
Устройства человеко-машинного интерфейса
конфигурирование обмена данными через PROFINET, 262
обзор, 20
создание сетевого соединения, 83
Утерянный пароль, 58

Ф

Функциональный блок (FB)
выходные параметры, 94
начальное значение, 94
экземплярный блок данных, 94
Функция (FC), 93

Ч

Часы
часы реального времени, 53
Числа
вещественные, 65
с плавающей точкой, 65

Ш

ШИМ (широтно-импульсная модуляция)
команда CTRL_PWM, 206
Шинный соединитель, 13

Э

Экземплярный блок данных, 58
Электромагнитная совместимость (ЭМС), 332

C

CPU
баланс мощностей, 23
восстановление утерянного пароля, 58
время цикла, 51
добавление модулей, 81
добавление нового устройства, 78
загрузка в устройство, 261
заземление, 35
защита паролем, 57
индуктивные нагрузки, 37
исполнение программы, 40
конфигурация устройств, 77
конфигурирование параметров, 80
конфигурирование связи с устройствами человеко-машинного интерфейса, 262
ламповая нагрузка, 38
неопределенный CPU, 79
обзор, 11
обработка запуска, 43
онлайн, 320
панель оператора для онлайнового CPU, 320
параметры запуска, 41, 70
передаточная карта, 71
переход в онлайн, 319
порт Ethernet, 84, 257
последовательность монтажа, 26
потребности в мощности, 373
программная карта, 73
пустая передаточная карта, 58
режимы работы, 42
сигнальная плата (SB), 13
создание передаточной карты, 71
создание программной карты, 73
создание сетевого соединения, 83
состояние STOP, 325
сравнительная таблица, 12
схемы соединений 1211C, 339
схемы соединений 1212C, 344
схемы соединений 1214C, 349
таблицы наблюдения, 323
технические данные 1211C, 335
технические данные 1212C, 340
технические данные 1214C, 345
указания для потенциальной развязки, 35
указания по подключению, 34, 36
уровни защиты, 57
утерянный пароль, 58
IP-адрес, 84, 257
MAC-адрес, 275
PROFINET, 84, 257
CTS, 282

D

DB (блок данных), 95

E

EN и ENO (поток сигнала), 98
Ethernet
IP-адрес, 84, 257
создание сетевого соединения, 83

F

FB (функциональный блок), 94
FBD (функциональная блок-схема, функциональный план), 98
FC (функция), 93

H

HSC (скоростной счетчик), 121
конфигурирование, 124

I

IP-адрес, 84, 85, 257, 258
конфигурирование, 84, 257
назначение, 252, 260
назначение в режиме онлайн, 255
IP-адрес маршрутизатора, 85, 258
IP-адрес, установка в онлайновом CPU, 320
IP-маршрутизатор, 85, 258

L

LAD (цепная логическая схема, контактный план), 97

M

MAC-адрес, 84, 257, 275
MB_COMM_LOAD, 222
MB_MASTER, 225
MB_SLAVE, 237
MODBUS, 222
MB_Master, 225
MB_SLAVE, 237

P

PROFINET, 249
создание сетевого соединения, 83
тестирование сети, 259
IP-адрес, 84, 257
PTO (вывод последовательности импульсов), 206
PtP-связь, 279

R

RTS, 282
RTS всегда включен, 282

S

S7-1200
баланс мощностей, 23
время цикла, 51
добавление модулей, 81
добавление нового устройства, 78
заземление, 35
зазор, 22
защита паролем, 57
индуктивные нагрузки, 37
клеммный блок, 33
коммуникационный модуль (CM), 14
конфигурация устройств, 77
конфигурирование модулей, 82
конфигурирование параметров CPU, 80
ламповые нагрузки, 38
монтажные размеры, 24
параметры запуска, 41, 70
передаточная карта, 71
порт Ethernet, 84, 257
программная карта, 73
пустая передаточная карта, 58
расширение возможностей, 13
сигнальная плата (SB), 13
сигнальный модуль (SM), 14
создание сетевого соединения, 83
сравнительная таблица моделей CPU, 12
указания для потенциальной развязки, 35
указания по подключению, 34, 36
указания по установке CPU, 26
установка и удаление устройств, обзор, 25
установка CM, 30
установка SB, 32
установка SM, 28
устройства человеко-машинного интерфейса, 20
утерянный пароль, 58
IP-адрес, 84 257
PROFINET, 84, 257
CPU, 11
SB 1223, схема подключения, 366
SB 1223, технические данные, 364, 367
SB 1232 схема подключения, 368
STEP 7
добавление модулей, 81
добавление нового устройства, 78
конфигурация устройств, 77
конфигурирование модулей, 82
конфигурирование CPU, 80
портальное представление, 16
порт Ethernet, 84, 257
проектное представление, 16
создание сетевого соединения, 83
установка, 15
PROFINET, 84, 257

T

TSAP (точка доступа к услугам транспортного уровня), 369, 272

X

XON / XOFF, 283