SIEMENS

Базовое системное руководство • 11/2015

Руководство по программированию S7-1200/S7-1500

STEP 7 (TIA Portal) и STEP 7 Safety в TIA Portal

http://www.siemens.com/simatic-programming-guideline

Гарантии и ответственность

Примечание

Примеры приложений не являются обязательными и не охватывают все возможные варианты конфигураций, оборудования или возникающих нештатных ситуаций. Примеры приложений не содержат решений, специфичных для конкретного клиента. Они предназначены только для поддержки типовых приложений. Вы несете ответственность за правильную эксплуатацию описанных продуктов. Эти примеры не освобождают Вас от ответственности за безопасную и профессиональную эксплуатацию установок, эксплуатацию и техническое обслуживание оборудования. Используя примеры приложений, Вы признаете, что мы не можем нести ответственность за любой ущерб, не входящий в ограниченную ответственность. Мы оставляем за собой право в любой момент вносить изменения в эти примеры приложений без предварительного извещения. Если есть какие-либо отличия между рекомендациями, содержащимися в данных примерах приложений, и другими публикациями компании Siemens (например, каталогами), то приотритетным является содержание других документов.

Мы не несем никакой ответственности за информацию, содержащуюся в настоящем документе.

Любые претензии к нам (в зависимости от юридической ситуации), как результат использования примеров, информационных программ, инжиниринговых данных, эксплуатационных характеристик и т.д., описанных в данных примерах приложений, не рассматриваются. Данное исключение не считается действительным в случае обязательной ответственности, например, в соответствии с “Законом об ответственности за качество продукции” (German Product Liability Act), в случае причинения ущерба здоровью, возникновения гарантийных обязательств, умышленного сокрытия дефектов или нарушения условий контракта (“Основные договорные обязательства”). Однако, возмещение ущерба в случае нарушения основных договорных обязательств ограничивается прогнозируемыми ситуациями, типичными для данного типа договора, кроме случаев умышленной порчи оборудования или причинению вреда жизни или здоровью вследствие грубой неосторожности. Приведенные выше положения не означают уменьшение степени ответственности за нанесенный Вам ущерб.

Не допускаются любые виды копирования или распространения данных примеров приложений или их частей без согласия с Siemens Industry Sector (Промышленны сектор Siemens).

Информационная безопасность

Компания Siemens предлагает продукты и решения с функциями промышленной безопасности, которые поддерживают надежную эксплуатацию предприятий, решений, оборудования и/или сетей. Они являются важными компонентами в глобальной концепции промышленной безопасности. Учитывая это, продукты и решения компании Siemens непрерывно совершенствуются. Компания Siemens настоятельно рекомендует регулярно проверять наличие обновлений для испульзуемых Вами продуктов.

Для безопасной эксплуатации продуктов и решений компании Siemens необходимо принимать соответствующие превентивные меры (например, концепция защиты ячеек) и интегрировать каждый компонент в глобальную концепцию промышленной безопасности, отвечающую самым современным требованиям. Также необходимо рассматривать использование продуктов сторонних производителей. Дополнительную информацию о промышленной безопасности можно найти на Интернет-странице: http://www.siemens.com/industrialsecurity.

Вы всегда можете оформить подписку на рассылку обновлений, выпускаемых для используемого Вами продукта. Дополнительную информацию можно найти на Интернет-странице: http://support.automation.siemens.com.

Содержание

1 Введение

Цели разработки нового поколения контроллеров SIMATIC

- Единая среда разработки для всех компонентов (контроллеры, устройства HMI, привода, и.т.д.)
- Однообразное программирование
- Повышенная производительность
- Полный набор команд для каждого языка
- Вся программа представлена с символьными именами
- Обработка данных без использования указателя
- Повторное использование созданных блоков

Цель руководства

Системная архитекутра нового поколения контроллеров SIMATIC S7-1200 и S7-1500 была обновлена и, при использовании TIA Portal, данные новшества дают преимущества при программировании и создании конфигурации контроллеров.

В данном документе описаны рекомендации и советы по эффективному программированию контроллеров S7-1200/1500. Некоторое различия в системной архитектуре с S7-300/400, также как и новые возможности при программировании описаны на простом и понятном языке. Это поможет Вам создавать стандартизованные и оптимальные решения задач автоматизации.

Приведенные примеры, могут быть использованы как в контроллерах S7-1200, так и в S7-1500.

Ключевые моменты данного руководства

Следующие возможности TIA Portal описаны в данном документе:

- Новшества S7-1200/1500
  - Языки программирования
  - Оптимизированные блоки
  - Типы данных и инструкции
- Рекомендации по программированию
  - Операционная система и пользовательская программа
  - Принцип хранения данных
  - Символьная адресация
  - Библиотеки
- Рекомендации по аппаратно-независимому программированию
- Рекомендации по STEP 7 Safety в TIA Portal
- Обзор наиболее важных рекомендаций

Преимущества и новые возможности

Большинство преимуществ достигается при применении следующих рекомендаций:

- Мощная программа пользователя
- Понятные программные структуры
- Интуитивные и эффективные программные решения

Дополнительная информация

При программировании контроллеров SIMATIC, задача программиста - создать понятную и читаемую пользовательскую программу. Каждый пользователь использует свою стратегию, например, как создавать и называть теги, блоки или комментарии. Различный подход к созданию программ пользователя, которые могут быть понятны только самому программисту.

Руководство по программированию позволяет Вам придерживаться определенного стиля и набора правил при создании программ. Данный механизм, например описывает назначение тегов и имен блоков, что упрощает задачу разработчику, например, при программировании на SCL.

Вы можете использовать данные правила и рекомендации в будущем; они являются предпосылками (не стандарт для программирования) к консистентному программированию.

Примечание

Вы можете найти руководство по программированию S7-1200 и S7-1500 по следующей ссылке:
https://support.industry.siemens.com/cs/ww/en/view/81318674

2 Инновации в S7-1200/1500

2.1 Введение

В основном, программирование контроллеров SIMATIC осталось таким же, как в S7-300/400 и для S7-1500. Имеются одинаковые языки программирования, такие как LAD, FBD, STL, SCL или GRAPH, такие же типы блоков, например огранизационные блоки (OB), функциональные блоки (FB), функции (FC) или блоки данных (DB). Т.е. уже созданные программы для S7-300/400 могут быть использованы для S7-1500, а программы на LAD, FBD и SCL использоваться на контроллере S7-1200.

Помимо этого, имеется множество других нововведений, которые упростят программирование и которые помогут создавать мощный и экономичный в плане памяти код.

Мы не только рекомендуем создание программ, которые подойдут 1:1 для контроллеров S7-1200/1500, но также использовать новые возможности и где это возможно, применять их. С новыми возможностями, Вы получаете программный код, который, например

- оптимальный по использованию памяти для CPU
- легкий для понимания
- прост для дальнейшей эксплуатации программы

Примечание

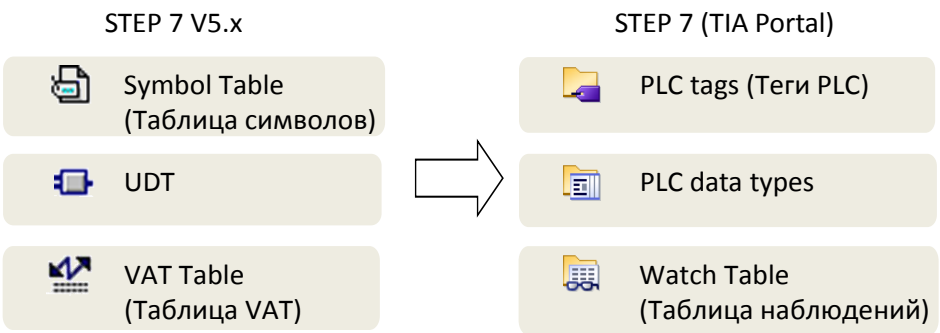
Информация по миграции с S7-300/S7-400 в S7-1500 доступна по следующей ссылке:
https://support.industry.siemens.com/cs/ww/en/view/109478811

2.2 Термины

Основные термины в TIA Portal

Некоторые термины были изменены для более удобной работы в TIA Portal .

Рисунок 2-1: Новые термины в TIA Portal



Термины для тегов и параметров

При работе с тегами, функциями, и функциональными блоками, большинство используется неправильно или некорректно. На следующем изображении показаны данные отличия.

Рисунок 2-2: Термины для тегов и параметров

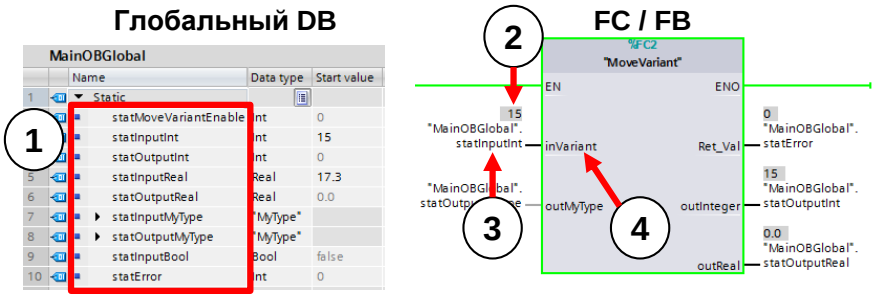


Таблица 2-1: Термины тегов и параметров

	Термин	Описание
1.	Тег	Теги отображаются с именем/идентификатором и назначенным адресом в памяти контроллера. Теги всегда определяются с типом данных (Bool, Integer, и.т.д.): • PLC теги • Теги в блоках данных • Целые блоки данных
2.	Значение тега	Значения тега хранится в самом теге (например, 15 это значение целочисленного тега).
3.	Фактический параметр	Фактические параметры это теги, которые связаны с интерфейсом или инструкциями, функций или функциональных блоков.
4.	Формальный параметр (для передачи данных, парметр блока)	Формальные параметры это интерфейсные параметры функций и функциональных блоков (Входные, Выходные, проходные, и Ret_Val).

Примечание

Вы можете найти дополнительную информацию по следующим ссылкам:

Сколько информации доступно в интернете для миграции STEP 7 (TIA Portal) и WinCC (TIA Portal)?
https://support.industry.siemens.com/cs/ww/en/view/56314851

Какие требования должны быть выполнены для миграции проекта STEP 7 V5.x в STEP 7 Professional (TIA Portal)?
https://support.industry.siemens.com/cs/ww/en/view/62100731

Миграция PLC для S7-1500 со STEP 7 (TIA Portal)
https://support.industry.siemens.com/cs/ww/en/view/67858106

Насколько рационально и эффективно Вы программируете для S7-1200/S7-1500 в STEP 7 (TIA Portal)?
https://support.industry.siemens.com/cs/ww/en/view/67582299

Почему невозможна паредача различных регистров и явная передача параметров для S7-1500 в STEP 7 (TIA Portal)?
Помимо этого, миграция программ STL для S7-1500 описана по этой ссылке.
https://support.industry.siemens.com/cs/ww/en/view/67655405

2.3 Языки программирования

Для создания пользовательских программ имеется несколько языков программирования. У каждого языка есть свои преимущества, которые могут быть в дальнейшем использованы, в зависимости от программы. Каждый блок в пользовательской программе может быть создан на любом языке программирования.

Таблица 2-2: Языки программирования

Языки программирования	S7-1200	S7-1500
Ladder (LAD)	✓	✓
Function block diagram (FBD)	✓	✓
Structured control language (SCL)	✓	✓
Graph	✗	✓
Statement list (STL)	✗	✓

Примечание

Вы можете найти дополнительную информацию по следующим вопросам:

Сравнение языков программирования SIMATIC S7-1200 / S7-1500
https://support.industry.siemens.com/cs/ww/en/view/86630375

Какие требования необходимо учесть при миграции S7-SCL программ в STEP 7 (TIA Portal)?
https://support.industry.siemens.com/cs/ww/en/view/59784005

Какие инструкции не поддерживаются в STEP 7 (TIA Portal) программе на SCL ? https://support.industry.siemens.com/cs/ww/en/view/58002709

Каким образом определяются константы в STEP 7 (TIA Portal) в S7-SCL программе?
https://support.industry.siemens.com/cs/ww/en/view/52258437

2.4 Оптимизированный машинный код

TIA Portal и S7-1200/1500 позволяют получить оптимизированную производительность режима исполнения на всех языках программирования. Все языки одинаково компилируются сразу в машинный код.

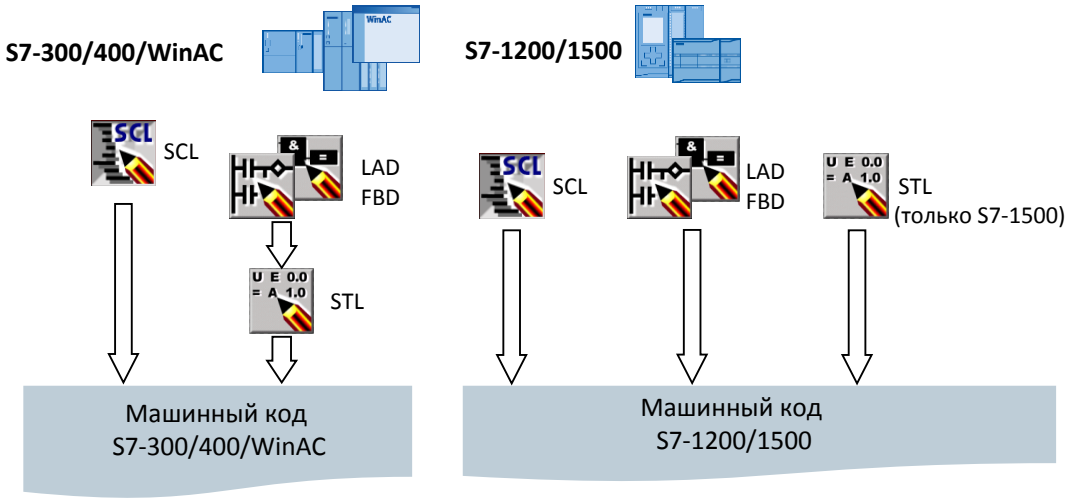
Преимущества

- У всех языков программирования одинаково высокий уровень производительности (при одинаковом типе доступа)
- Производительность не понижается при компиляции через промежуточный шаг на язык STL

Свойства

На следующем рисунке показаны отличия при компиляции S7 программ в машинный код.

Рисунок 2-3: Генерация машинного кода для S7-300/400/WinAC и S7-1200/1500

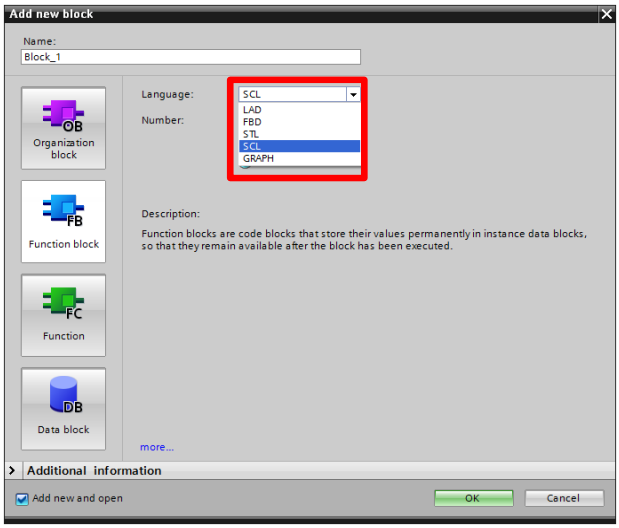


- Для контроллеров S7-300/400/WinAC программы на языках LAD и FBD сначала компилируются в STL, а потом уже в машинный код.
- Для контроллеров S7-1200/1500 все языки программирования компилируются сразу в машинный код.

2.5 Создание блоков

Все блоки, такие как, OB, FB и FC могут быть запрограммированы на описанных ранее языках программирования. Таким образом, исходный текст для программирования на SCL не создается. Вы можете выбрать язык программирования SCL, когда создаете блок. После этого, блок может быть сразу запрограммирован.

Рисунок 2-4: Окно “Add new block” (Создание нового блока)



2.6 Оптимизированные блоки

У контроллеров S7-1200/1500 имеется возможность оптимизированного хранения данных. В оптимизированных блоках, все теги автоматически сортируются по их типу данных. Данный метод позволяет минимизировать промежутки между тегами, таким образом такие теги оптимизированы по времени доступа для процессора.

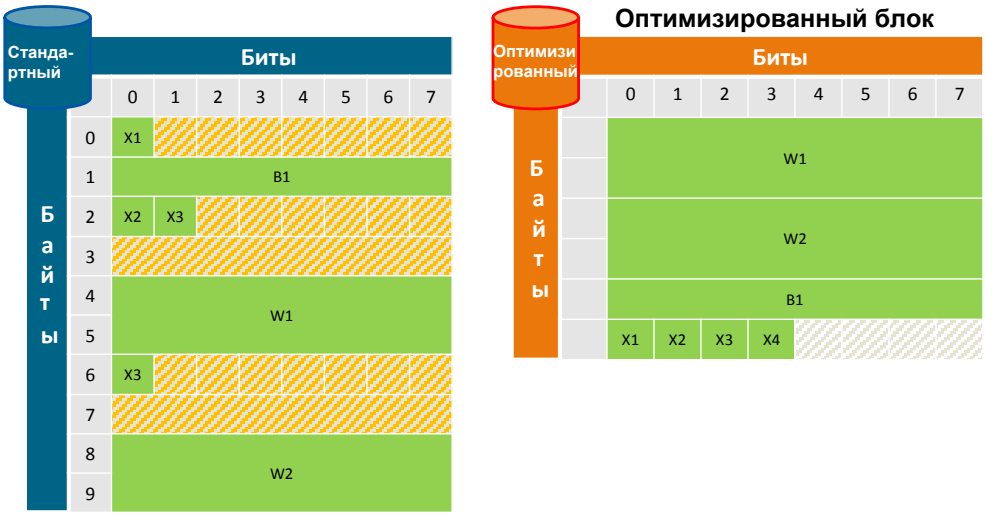
Неоптимизированные блоки существуют в целях совместимости с S7-1200/1500.

Преимущества

- Доступ всегда выполняется быстро, так как база в которой хранится информация оптимизирована системой и независима от ее описания.
- Отсутствует возможность ошибки при обращении к данным при использовании, абсолютной адресации, которую заменила символьная адресация.
- Изменения в описании переменных не повлекут ошибки доступа, поскольку, например, HMI получает доступ символьно.
- Отдельные теги могут быть описаны как “retain” (сохраняемые).
- Никаких настроек в экземплярном блоке данных не требуется. Все задается в FB (включая сохраняемость).
- Резерв памяти в блоке данных позволяет изменять фактические значения без потерь данных (см. главу 3.2.7 Загрузка без повторной инициализации)

2.6.1 S7-1200: Оптимизированный блок

Рисунок 2-5: Оптимизированный блок S7-1200



Свойства

- Промежутки между тегам отсутствуют, так как теги с наибольшим типом данных наибольшего размера располагаются в начале блока, а с наименьшим в конце.
- Для оптимизированных блоков возможен только символьный доступ.

2.6.2 S7-1500: Оптимизированный блок

Рисунок 2-6: Оптимизированный блок S7-1500

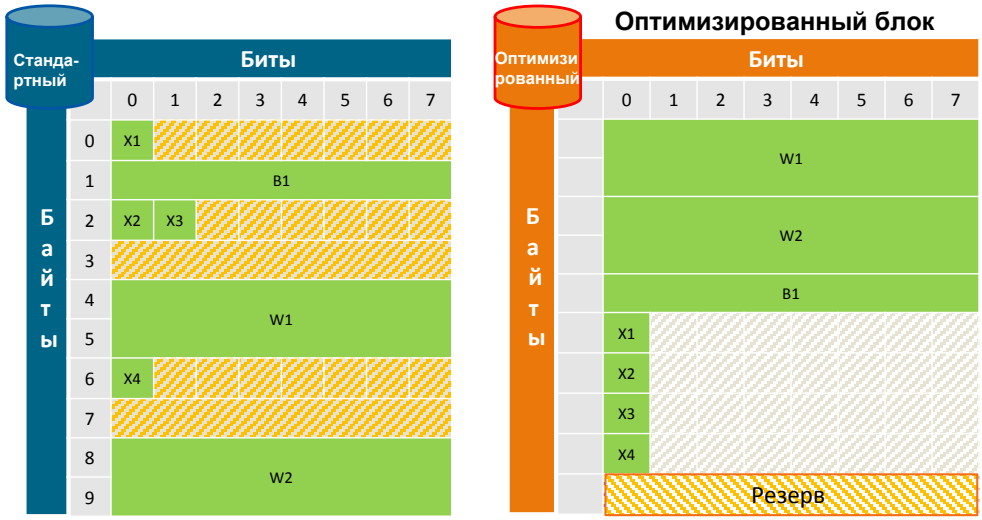
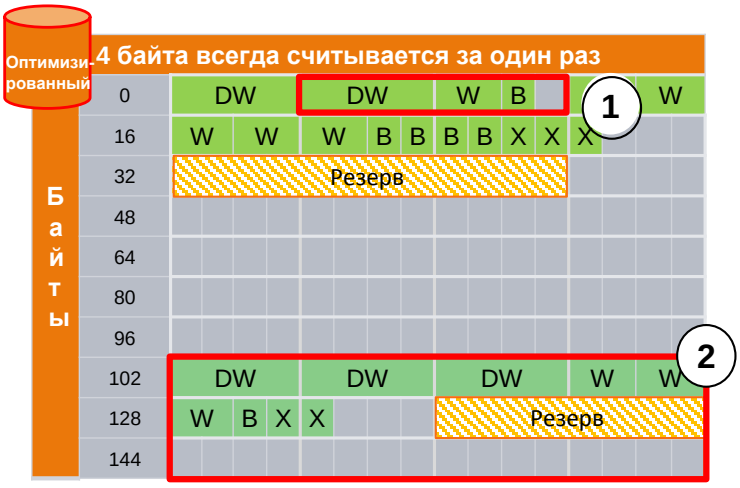


Рисунок 2-7: Распределение памяти в оптимизированных блоках данных



1. Структуры хранятся раздельно и могут быть скопированы как один блок.
2. Сохраняемые данные записываются в отдельной области и могут быть скопированы как один блок. В случае сбоя по питанию, эта информация сохраняется во внутреннюю память CPU. "MRES" выполнит сброс этой информации на начальные значения, хранящиеся в загрузочной памяти.

Свойства

- Промежутки между данными отсутствуют, так как теги с наибольшим размером типа данных располагаются в начале, а с наименьшим в конце.
- Быстрый доступ, благодаря такой структуре хранения в памяти в процессоре (Все теги записываются таким образом, чтобы S7-1500 мог напрямую считывать и записывать теги одной машинной командой).
- Теги с логическим типом (Bool) сохраняются как байт для более быстрого доступа. По этой причине контроллеру не нужна маска доступа.

- У оптимизированных блоков имеется резрвная память для перезагрузки в процессе работы (см. главу 3.2.7 Загрузка без повторной инициализации).
- Для оптимизированных блоков возможен только символьный доступ.

2.6.3 Наилучший возможный вариант хранения данных в S7-1500

В целях совместимости с первыми контроллерами SIMATIC, принцип хранения данных “Big-Endian” был применен в контроллерах S7-300/400 .

Новое поколение контроллеров S7-1500 всегда получает доступ к 4 байтам (32 бита) в последовательности “Little-Endian”, в силу изменения архитектуры процессера. Это дает в некоторых случаях свои преимущества.

Рисунок 2-8: Доступ к данным контроллера S7-1500

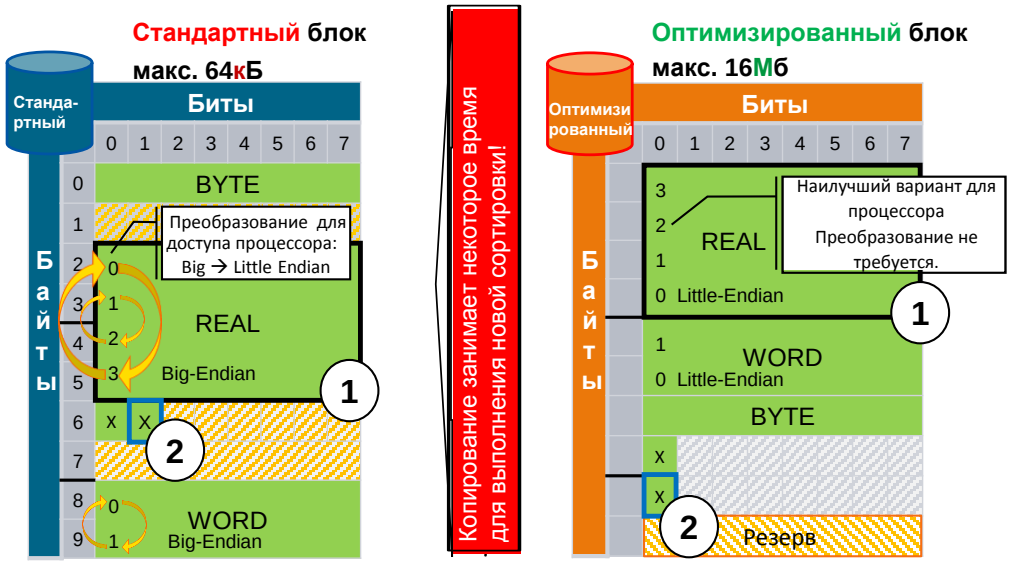


Таблица 2-3: Доступ к данным в контроллере S7-1500

	Стандартный блок	Оптимизированный блок
1.	Контроллеру необходимо получить доступ к 2x16 битам для получения доступа к 4 байтному значению (например, типа REAL). При этом, последовательность байт необходимо изменить.	Контроллер сохраняет теги, доступ оптимизирован. Выполняется 32 битный (REAL) доступ. Изменение последовательности байтов не требуется.
2.	Считывается весь байт и накладывается маска Байт блокируется для любого другого доступа.	Каждому биту присваивается байт. При получении доступа, контроллер не накладывает маску на байт.
3.	Максимальный размер блока 64Кб.	Максимальный размер блока 16Мб.

Рекомендация

- Всегда используйте только оптимизированные блоки.
  - Они не требуют абсолютной адресации, доступ к ним может быть получен при помощи символьной адресации. Косвенная адресация также возможна при помощи символьных данных (см. главу 3.6.2 Тип данных ARRAY и косвенный доступ к элементам).
  - Обработка оптимизированных блоков в контроллере гораздо быстрее чем стандартных блоков.
- Избегайте копирования данных между оптимизированными и неоптимизированными блоками. Необходимые преобразования между исходным форматом и необходимым могут занять много процессного времени.

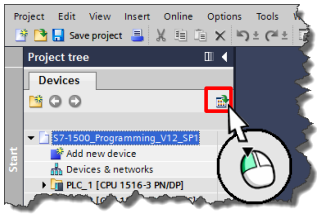
Пример: Установка оптимизированного доступа к блоку

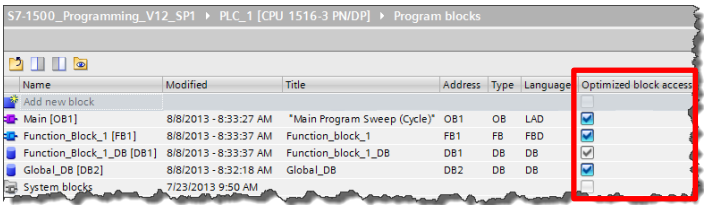
Оптимизированный доступ к блоку по умолчанию активирован для всех созданных блоков на S7-1200/1500. Доступ к блоку может быть установлен для OB, FB и глобальных DB. Для экземплярных DB, настройка зависит от соответствующего FB.

При миграции блока с контроллера S7-300/400 на S7-1200/1500, доступ к блоку не сбрасывается автоматически. Вы можете изменить тип доступа позже да “optimized block access” (оптимизированный доступ к блоку). Вам необходимо будет скомпилировать программу после изменения доступа к блоку. Если Вы измените FB на “optimized block access” (оптимизированный доступ к блоку), то назначенные экземплярные блоки данных будут автоматически обновлены.

Следуйте инструкциям, которые описаны ниже, для задания оптимизированного доступа к блоку.

Таблица 2-4: Задание оптимизированного доступа к блоку

Шаг	Инструкция
1.	Нажмите кнопку “Maximizes/minimizes the Overview” (Развернуть/свернуть отображение) в навигаторе проекта. 
2.	Перейдите в папку “Program blocks“ (Программные блоки).

Шаг	Инструкция
3.	В данном окне Вы увидите все блоки в программе и настройку их оптимизированности. В данном окне параметр “Optimized block access” (Оптимизированный доступ к блоку) может быть также изменен.  Примечание: Экземплярные блоки данных (здесь “Function_block_1_DB”) наследуют состояние “оптимизированный” от соответствующего FB. Именно поэтому свойство “оптимизированный” задается в FB. После компиляции проекта DB будет присвоено свойство, которое назначено для FB.

Отображение оптимизированных и неоптимизированных блоков в TIA Portal

На двух следующих изображениях можно увидеть разницу между оптимизированным и неоптимизированным DB.

К глобальным DB относятся те же самые отличия.

Рисунок 2-9: Оптимизированный блок данных (без смещения)

Function_Block_1_DB

	Name	Data type	Start value	Retain	Visible in ...	Setpoint
1	Input					
2	Input_bool_1	Bool	false	☑	☑	
3	Input_byte_1	Byte	16#0		☑	
4	Input_bool_2	Bool	false	☑	☑	
5	Input_word	Word	16#0		☑	
6	Input_byte_2	Byte	16#0		☑	
7	Output					
8	Output_bool_1	Bool	false		☑	
9	InOut					
10	Static					

Рисунок 2-10: Нептимизированный блок данных (со смещением)

Function_Block_1_DB

	Name	Data type	Offset	Start value	Retain	Visible in ...	Setpoint
1	Input						
2	Input_bool_1	Bool	0.0	false		☑	
3	Input_byte_1	Byte	1.0	16#0		☑	
4	Input_bool_2	Bool	2.0	false		☑	
5	Input_word	Word	4.0	16#0		☑	
6	Input_byte_2	Byte	6.0	16#0		☑	
7	Output						
8	Output_bool_1	Bool	8.0	false		☑	
9	InOut						
10	Static						

Таблица 2-5: Различия: оптимизированный и неоптимизированный блок данных

Оптимизированный блок данных	Неоптимизированный блок данных
Оптимизированные блоки данных адресуются символьно. “Смещение” **не** отображается.	В неоптимизированных блоках “смещение” отображается и может быть использовано для адресации.
В оптимизированный блоках **любой отдельный** тег может описываться как “Retain” (Сохраняемый).	В неоптимизированных блоках только **все или ни один** тег могут иметь свойство “Retain” (Сохраняемый).

Сохраняемость тегов глобального DB задается в самом DB. По умолчанию он сохраняемый.

Сохраняемость тегов экземплярного блока определяется в функциональном блоке (не в экземплярном DB). Данная настройка далее будет применена ко всем экземплярным блокам FB.

Типы доступа для оптимизированных и неоптимизированных блоков

В следующей таблица показаны все типы доступа к блокам.

Таблица 2-6: Типы доступа

Тип доступа	Оптимизированный блок	Неоптмизированный блок
Символьный	✓	✓
Индексированный (к элементам)	✓	✓
Выборочный (Slice) доступ	✓	✓
AT инструкция	× (Альтернатива: выборочный доступ)	✓
Абсолютный	× (Альтернатива: массив с индексом)	✓
Косвенный (через указатель)	× (Альтернатива: VARIANT / массив с индексом)	✓
Загрузка без повторной инициализации	✓	×

Примечание

Вы можете найти дополнительную информацию по следующим вопросам:

Какие различия необходимо знать между хранением данных при оптимизированном и стандартном доступе в STEP 7 (TIA Portal)?
https://support.industry.siemens.com/cs/ww/en/view/67655611

На какие свойства Вам необходимо обратить внимание в STEP 7 (TIA Portal) для инструкций "READ_DBL" и "WRIT_DBL", при использовании DB с оптимизированным доступом?
https://support.industry.siemens.com/cs/ww/en/view/51434747

2.6.4 Преобразование между оптимизированными неоптимизированными тегами

Главная рекомендация - это работать с оптимизированными тегами. Тем не менее, если Вы хотите в каких-то случаях придерживаться старого стиля программирования, будет смесь оптимизированных и неоптмизированных блоков в программе.

Система определяет внутренее место записи каждого тега, независимо от того сложного (с типом данных определенным пользователем) или элементарного типа данных (INT, LREAL, и.т.д.) он был описан.

В случае взаимодействия между двумя тегами с различным принципом хранения, система автоматически выполнит преобразование. В случае со структурированными тегами, данное преобразование требует требует повышенной производительности и по возможности, такой ситуации лучше избегать.

2.6.5 Передача параметров между блоками с оптимизированным и стандартным типом доступа

Если при вызове блока, структуры передаются в вызывающий блок в качестве проходного (InOut) параметра, они будут переданы по ссылке (см. главу 3.3.2 Передача по ссылке через проходные параметры).

Тем не менее это не относится к ситуации, если один блок со свойством “Optimized access” (Оптимизированный доступ), а другой блок имеет свойство “Standard access” (Стандартный доступ). Принципиально, все параметры передаются как копии (см. главу 3.3.1 Передача по значению через входные параметры).

В этом случае, вызываемый блок всегда работает с копиями данных. В процессе обработки блока, данные значения могут быть изменены и, после этого, быть скопированы обратно в исходный операнд.

Данный механизм может быть некорректен в некоторых случаях, если исходный операнд изменяется асинхронным процессом, например, доступом с HMI или OB обработки ошибок. Если, после обработки блока, копии будут переданы в исходный операнд, то данные асинхронных изменений будут в нем перезаписаны.

Примечание

Вы можете найти дополнительную информацию по следующему вопросу:

Почему данные HMI системы или Web сервера перезаписываются в S7-1500?
https://support.industry.siemens.com/cs/ww/en/view/109478253

Рекомендация:

- Всегда устанавливайте одинаковый тип доступа для обоих блоков, которые взаимодействуют друг с другом.

2.6.6 Коммуникация с оптимизированными данными

Интерфейс (CPU, CM) передает данные в таком порядке в каком они расположены (не важно оптимизированный или нет).

Рисунок 2-11: Коммуникация CPU-CPU

Отправляемая информация может быть:

- оптимизированной
- неоптимизированной
- тегом (любой тип)
- буфером (массив байтов)

Получаемая информация может быть:

- оптимизированной
- неоптимизированной
- тегом (любой тип)
- буфером (массив байтов)

Пример

- Тег с типом данных PLC (запись данных) передается в CPU.
- В отправляющем CPU, тег задан как фактический параметр коммуникационного блока (TSEND_C).
- В принимающем CPU, получаемая информация передается в тег того же типа
- В данном случае, с полученной информацие можно продолжать работать с символьной адресацией

Примечание

Любые теги или блоки данных (производные от типов данных PLC data types) могут быть использованы в качестве записей данных.

Примечание

Также возможно, что отправляемые и получаемые данные не будут идентичны:

Отправляемые данные	**Получаемые данные**
оптимизированные ->	неоптимизированные
неоптимизированные ->	оптимизированные

Контроллер автоматически производит корректную передачу данных и их сохранение.

2.7 Свойства блока

2.7.1 Размеры блока

Для контроллеров S7-1200/1500 максимальный размер блоков был значительно увеличен.

Таблица 2-7: Размеры блоков

Максимальный размер и количество (зависит от размера основной памяти)		**S7 -300/400**	**S7-1200**	**S7-1500**
DB	Макс. размер	64 кб	64 кб	64 кб (неоптимизированный) **10 Мб** (оптимизированный CPU1518)
	Макс. кол-во	16.000	65.535	65.535
FC/FB	Макс. размер	64 кб	64 кб	512 кб **3 Мб** (оптимизированный CPU1518)
	Макс. кол-во	7.999	65.535	65.535
FC / FB / DB	Макс. кол-во	4.096 (CPU319) 6.000 (CPU412)	1.024	10.000 (CPU1518)

Рекомендация

- Используйте DB для контроллеров S7-1500 в качестве места хранения данных больших объемов.
- Данные с размером > 64 кб, могут быть сохранены в оптимизированном DB (максимальный размер 10 Мб) в контроллерах S7-1500.

2.7.2 Количество организационных блоков (OB)

OB могут быть использованы для создания иерархии пользовательской программы. Для этой цели доступны различные OB.

Таблица 2-8: Количество организационных блоков

Тип организационного блока	**S7-1200**	**S7-1500**	**Цель**
Циклические и стартовые OB	100	100	Модуляризация программы
Аппаратное прерывание	50	50	OB обработки отдельных событий
Прерывание с задержкой времени	4*	20	Модуляризация программы
Циклическое прерывание		20	Модуляризация программы
Время дня	×	20	Модуляризация программы

* начиная с версии операционной системы V4 доступны прерывания с задержкой и 4 "watchdog" прерывания .

Рекомендация

- Используйте OB для создания иерархии структурированных пользовательских программ.
- Для более подробной информации по OB, обратитесь к главе 3.2.1 Организационные блоки (OB).

2.8 Новые типы данных для S7-1200/1500

Контроллеры S7-1200/1500 поддерживают новые типы данных, что позволяет сделать программирование более эффективным. При использовании новых 64 битных типов данных, возможно использование переменных с большим диапазоном значений и также увеличить точность вычислений.

Примечание

Вы можете найти дополнительную информацию по следующему вопросу:

Как выполнить преобразование типов данных в TIA Portal для S7-1200/1500?
https://support.industry.siemens.com/cs/ww/en/view/48711306

2.8.1 Элементарные типы данных

Таблица 2-9: Целочисленные типы данных

Тип	Размер	Value range
USint	8 бит	0 .. 255
SInt	8 бит	-128 .. 127
UInt	16 бит	0 .. 65535
UDInt	32 бита	0 .. 4.3 миллиона
ULInt*	64 бита	0 .. 18,4 Триллиона (10^{18})
LInt*	64 бита	-9,2 Триллиона .. 9,2 Триллиона
LWord	64 бита	16#0000 0000 0000 0000 до 16# FFFF FFFF FFFF FFFF

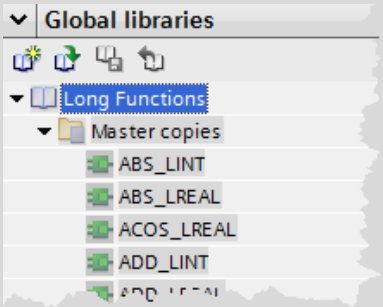
* только для S7-1500

Таблица 2-10: Десятичные типы данных с плавающей точкой

Тип	Размер	Диапазон значений
Real	32 бита (1 знаковый бит, 8 бит экспонента, 23 бита мантисса), с точностью до 7 знаков после запятой	-3.40e+38 .. 3.40e+38
LReal	64 бита (1 знаковый бит, 11 бит экспонента, 52 бита мантисса), с точностью до 15 знаков после запятой	-1.79e+308 .. 1.79e+308

Примечание

В TIA Portal имеется глобальная библиотека “Long Functions” с большим количеством инструкций для больших типов данных.



Примечание

Вы можете найти дополнительную информацию по следующему вопросу:

Почему в STEP 7 (TIA Portal), результат сложения типов DInt на SCL отображается некорректно?
https://support.industry.siemens.com/cs/ww/en/view/98278626

2.8.2 Тип данных Date_Time_Long

Таблица 2-11: Структура DTL (Date_Time_Long)

Год	Месяц	День	День недели	Час	Минута	Секунда	Наносекунда

DTL всегда считывает текущее системное время. Доступ к отдельным значениям выполняется символьно (например, `My_Timestamp.Hour`)

Преимущества

- К каждому полю (например, Year, Month, …) можно выполнить символьный доступ.

Рекомендация

Используйте новый тип данных DTL вместо LDT с символьным доступом (например, `My_Timestamp.Hour`).

Примечание

Вы можете найти дополнительную информацию по следующим вопросам:

Как Вы можете задать, считать и редактировать в STEP 7 (TIA Portal) дату и время для модулей CPU S7-300/S7-400/S7-1200/S7-1500?
https://support.industry.siemens.com/cs/ww/en/view/43566349

Какие функции доступны в STEP 7 V5.5 и TIA Portal для обработки типов данных DT и DTL?
https://support.industry.siemens.com/cs/ww/en/view/63900229

2.8.3 Вспомогательные типы данных для времени

Таблица 2-12: Типы данных для времени (только S7-1500)

Тип	Размер	Диапазон значений
LTime	64 бита	LT#-106751d23h47m16s854ms775us808ns до LT#+106751d23h47m16s854ms775us807ns
LTIME_OF_DAY	64 бита	LTOD#00:00:00.000000000 до LTOD#23:59:59.999999999

2.8.4 Типы данных для работы с Юникодом

Типы данных WCHAR и WSTRING могут быть использованы при работе с символами в формате Юникод.

Таблица 2-13: Типы данных для работы с Юникодом (только S7-1500)

Тип	Размер	Диапазон значений
WCHAR	2 байта	-
WSTRING	(4 + 2*n) байт	Предустановленное значение: 0 ..254 символов Макс. значение: 0 ..16382

Свойства

n = длина последовательности символов

- Например, обработка символов на Латинском Китайском или других языках.
- Разрывы строк, прокрутка страницы, символ табуляции, символ пробела
- Специальные символы: знак Доллара, кавычки

Пример

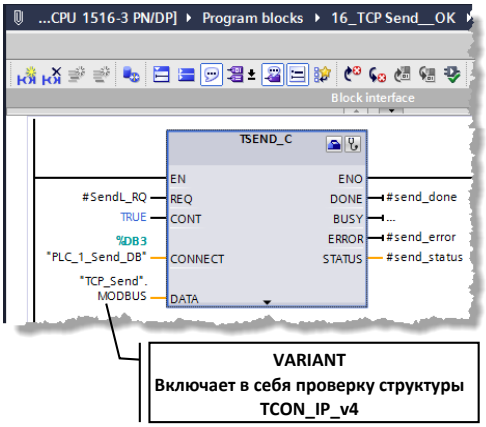
- `WCHAR#'a'`
- `WSTRING#'Hello World!'`

2.8.5 Тип данных VARIANT (S7-1500 и S7-1200 с версии 4.1)

Параметр типа VARIANT это указатель, который может ссылаться на теги различных типов данных. В отличие от указателя ANY, VARIANT это указатель с проверкой типа. Исходная и целевая структуры на выходе и на входе проверяются во время режима исполнения и они должны быть идентичными.

VARIANT используется, например, в качестве входного параметра для коммуникационных блоков (TSEND_C).

Рисунок 2-12: Тип данных VARIANT в качестве входного параметра для инструкции TSEND_C



Преимущества

- Встроенная проверка типа предотвращает ошибку доступа.
- Благодаря символьной адресации к тегам данного типа, код читается легче.
- Написание кода производится эффективнее и быстрее.
- Указатели интуитивно Variant понятны, чем указатели ANY.
- Теги Variant могут быть использованы напрямую в системных функциях.
- Возможна более гибкая и эффективная передача различных структурированных данных.

Свойства

Сравнение ANY и Variant позволяет увидеть отличие свойств данных типов.

Таблица 2-14: Сравнение ANY и Variant

ANY	Variant
Требует 10 Кбайт памяти с определенной структурой	Не требует пользовательской памяти
Инициализация либо через назначение области данных или с помощью заполнения структуры ANY	Инициализация при помощи назначения области данных системной инструкцией
Нетипизированный тип связанной структуры не может быть определен	Типизированный связанный тип может быть определен с длинной массива
Частично типизирован – Может быть определен размер массива	VARIANT может быть также обработан и создан при помощи системных инструкций

Рекомендации

- Используйте только тип данных VARIANT при работе с косвенной адресацией, если типы данных неизвестны до начала работы программы.
- Проверьте для чего Вы используете указатель ANY. Во многих случаях, его использование излишне (см. таблицу ниже).
- Используйте только тип данных VARIANT при работе с косвенной адресацией, если типы данных неизвестны до начала работы программы.
  - Используйте тип данных VARIANT в качестве InOut параметра для создания блоков, которые должны быть независимы от типа данных фактических параметров (см. пример в данной главе).
  - Используйте тип данных VARIANT вместо указателя ANY. Благодаря встроенной проверке типа, ошибки выявляются заранее. Благодаря символьной адресации, программный код становится более понятным.
  - Используйте инструкцию Variant, например, для определения типа (см. следующий пример и главу 2.9.3 VARIANT инструкции)
- Используйте индексированные массивы (ARRAY) вместо указателя ANY на адрес элементов массивов (см. главу 3.6.2 Тип данных ARRAY и косвенный доступ к элементам).

Таблица 2-15: Сравнение указателя ANY и его замена

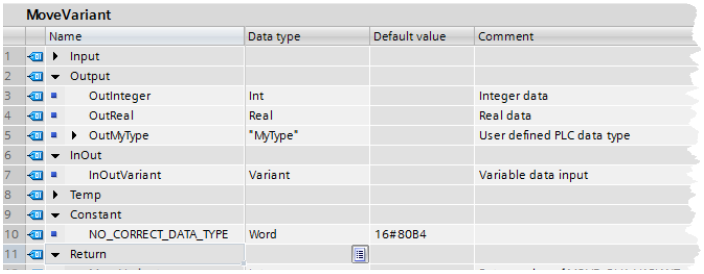
Где используется указатель ANY ?		**Замена в S7-1200/1500**
Функции программирования, в которых могут использоваться различные типы данных	→	Функции с указателем Variant в качестве InOut параметра для блоков (см. следующие примеры)
Обработка массивов • например, чтение, инициализация, копирование элементов одного типа	→	Стандартные функции для массивов • Чтение и запись с помощью #myArray[#index] (см. главу 3.6.2 Тип данных ARRAY и косвенный доступ к элементам) • Копирование с помощью MOVE_BLK (см. главу 2.9.2 Инструкции MOVE)
Передача и обработка структур • например, передача структуры, определенной пользователем, с помощью указателей ANY в функции	→	Передача структур в качестве InOut параметров • см. главу 3.3.2 Передача по ссылке через InOut параметр

Пример

При использовании типа данных VARIANT в пользовательской программе возможно определить тип данных и произвести соответствующую обработку. В следующей функции “MoveVariant” показан данный подход.

- Формальный параметр InOut “InVar” (тип данных VARIANT) используется, чтобы показать независимость тега от типа данных.
- Тип данных фактического параметра определяется с помощью инструкции “Type_Of” .
- При помощи инструкции “MOVE_BLK_VARIANT”, значение тега копируется в другие выходные формальные параметры, в зависимости от типа данных.

Рисунок 2-13: Формальные параметры функции “MoveVariant”



MoveVariant

	Name	Data type	Default value	Comment
1	Input			
2	Output			
3	OutInteger	Int		Integer data
4	OutReal	Real		Real data
5	OutMyType	"MyType"		User defined PLC data type
6	InOut			
7	InOutVariant	Variant		Variable data input
8	Temp			
9	Constant			
10	NO_CORRECT_DATA_TYPE	Word	16#80B4	
11	Return			

```
CASE TypeOf(#InOutVariant) OF // Check datatypes
Int:  // Передача целого типа (Integer)
    #MoveVariant := MOVE_BLK_VARIANT(SRC := #InOutVariant,
                                     COUNT := 1,
                                     SRC_INDEX := 0,
                                     DEST_INDEX := 0,
                                     DEST => #OutInteger);
Real:  // Передача вещественного типа (Real)
    #MoveVariant := MOVE_BLK_VARIANT(SRC := #InOutVariant,
                                     COUNT := 1,
                                     SRC_INDEX := 0,
                                     DEST_INDEX := 0,
                                     DEST => #OutReal);
MyType: // Передача собственного типа (MyType)
     #MoveVariant := MOVE_BLK_VARIANT(SRC := #InOutVariant,
                                     COUNT := 1,
                                     SRC_INDEX := 0,
                                     DEST_INDEX := 0,
                                     DEST => #OutMyType);
ELSE  // Ошибка, тип данных не определен
     #MoveVariant := WORD_TO_INT(#NO_CORRECT_DATA_TYPE);
     // 80B4: Код ошибки MOVE_BLK_VARIANT: Тип данных не соответствует
END_CASE;
```

Примечание

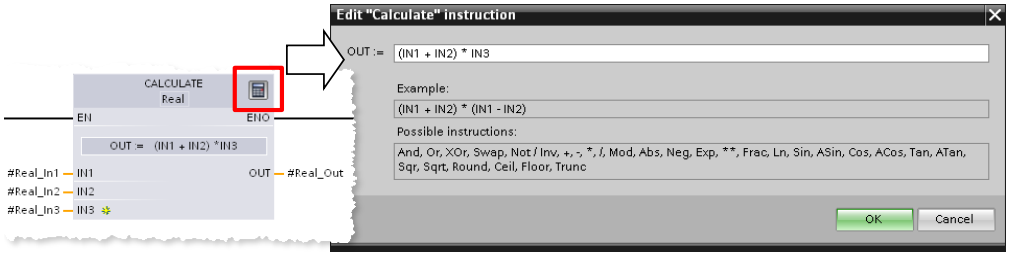
Если Вы хотите скопировать значения неструктурированных тегов VARIANT, Вы также можете использовать VariantGet вместо MOVE_BLK_VARIANT (см. главу 2.9.3 VARIANT инструкции).

2.9 Инструкции

2.9.1 CALCULATE

С помощью инструкции CALCULATE, Вы можете выполнить математические вычисления (например, (IN1 + IN2) * IN3), которые не зависят от типа данных. Математическая формула записывается в редакторе инструкции.

Рисунок 2-14: Инструкция CALCULATE с редактором формулы



Примечания

Для получения более подробной информации, обратитесь к Online справке TIA Portal по инструкции “CALCULATE”.

Преимущества

- Математическая формула использует только одну инструкцию
- Экономия времени благодаря простой настройке

Свойства

- Поддерживает последовательности битов, целые, вещественные числа
- Поддерживает различные математические функции (все основные арифметические операции, тригонометрические функции, округление, логарифмические функции, и.т.д.)
- Изменяемое количество входных параметров

Рекомендация

- Всегда используйте инструкцию CALCULATE для математических вычислений вместо множественного вызова таких инструкций, как ADD, SUB, и.т.д.

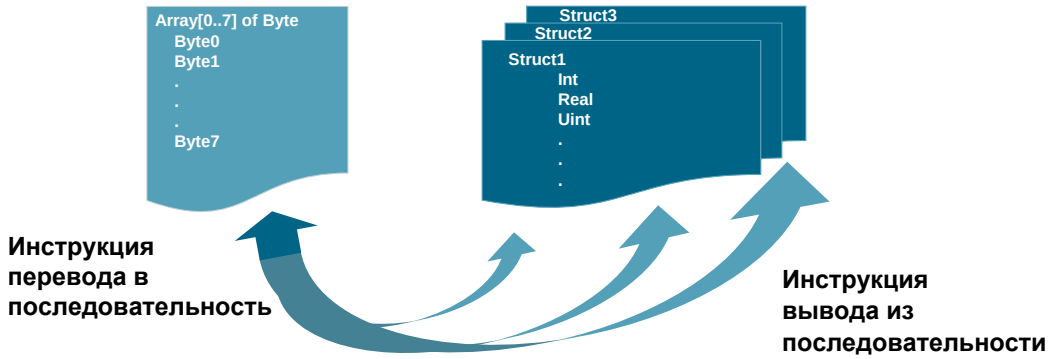
2.9.2 Инструкции MOVE

В STEP 7 (TIA) имеются следующие MOVE инструкции. Инструкция MOVE_BLK_VARIANT для S7-1200/1500 является новой.

Таблица 2-16: Инструкции перемещения

Инструкция	Использование	Свойства
MOVE	Копирование значения Копирование массива	• Копирование содержимого входного параметра IN в выходной параметр OUT. • Входные и выходные параметры должны быть одного типа. • Параметры могут быть также структурированными (PLC data types). • Копирование всего массива и структур.
MOVE_BLK	Копирование нескольких областей	• Копирование содержимого массива в другой массив. • Типы данных должны совпадать у исходного и целевого массива. • Копирование всего массива и структур. • Копирование нескольких элементов массива со структурами, начиная с определенного элемента.
UMOVE_BLK	Копирование массива без прерывания	• Консистентное копирование содержимого массива без риска прерывания копирования вызова OB обработки прерывания. • Типы данных должны совпадать у исходного и целевого массива.
MOVE_BLK_VARIANT (S7-1500 и S7-1200, начиная с FW4.1)	Копирование массива	• Копирование одного или нескольких структурированных тегов (PLC data types). • Определение типов данных в процессе работы • Детальная информация об ошибке • Также поддерживается отдельно для элементарных и структурированных типов, PLC data types, массивы, и массивы DB .
Переход в последовательность (S7-1500 и S7-1200, начиная с FW4.1)	Копирование структурированных данных в байтовый массив	• Несколько записей данных могут быть записаны в один байтовый массив и отправлены на другие устройства как фрейм. • Входные и выходные параметры могут быть переданы как тип Variant.
Вывод из последовательности (S7-1500 и S7-1200, начиная с FW4.1)	Копирование байтового массива в одну или несколько структур	• Применение для I-Device: I-Device получает некоторые записи во входную область, которые будут скопированы в различные структуры. • Несколько записей могут быть скомбинированы различные массив байтов. Вывод из последовательности позволяет скопировать их в различные структуры.

Рисунок 2-15: Инструкции перевода в последовательность и вывода из последовательности (S7-1500 и S7-1200, начиная с FW4.1)



Рекомендация

- Вы должны понимать разницу между MOVE, MOVE_BLK и MOVE_BLK_VARIANT
  - Используйте инструкцию MOVE для копирования всей структуры.
  - Используйте инструкцию MOVE_BLK для копирования частей массива (ARRAY) с известным типом данных.
  - Используйте инструкцию MOVE_BLK_VARIANT только, если Вы хотите копировать часть массива (ARRAY), тип данных которого будет известен только в процессе работы программы.

Примечание

UMOVE_BLK: Процесс копирования не может быть прерван другой задачей операционной системы. Поэтому, время реакции CPU может быть увеличено при работающей инструкции "Copy array without interruption" (Копирование массива без прерывания).

Для получения более подробной информации об инструкциях MOVE, обратитесь к Online помощи TIA Portal.

Примечание

Вы можете найти дополнительную информацию по следующим вопросам:

Как выполнять копирование областей памяти и структурированных данных из одного блока данных в другой в STEP 7 (TIA Portal) ?
https://support.industry.siemens.com/cs/ww/en/view/42603881

2.9.3 VARIANT инструкции (S7-1500 и S7-1200, начиная с FW4.1)

Таблица 2-17: Инструкции для типа данных Variant

Инструкция	Применение	Свойства
Инструкции MOVE		
VariantGet	Чтение значения	Данная инструкция позволяет Вам считать значение тега, ссылающегося на VARIANT.
VariantPut	Запись значения	Данная инструкция позволяет Вам записать значение тега, ссылающегося на VARIANT.
Список		
CountOfElements	Подсчет элементов	С помощью данной инструкции, Вы можете получить количество элементов в массиве из переменной типа VARIANT.
Инструкции сравнения		
TypeOf() (только SCL)	Определение типа данных	Используйте данную инструкцию для определения типа данных из переменной типа VARIANT.
TypeOfElements() (только SCL)	Определение типа данных массива	Используйте данную инструкцию для определения типа данных элементов массива из переменной типа VARIANT.
Инструкции преобразования		
VARIANT_TO_DB_ANY (только SCL)	Определение номера блока данных	С помощью данной инструкции, Вы можете получить номер экземплярного блока данных PLC data type, системного типа данных или массива DB.
DB_ANY_TO_VARIANT (только SCL)	Создание Variant тега из блока данных.	С помощью данной инструкции, Вы можете создать тег Variant из экземплярного блока данных с типом PLC data type, системного типа данных или массива DB.

Примечание

Для получения более подробной информации по инструкциям VARIANT, обратитесь к online помощи TIA Portal.

2.9.4 RUNTIME

При помощи инструкции "RUNTIME", Вы можете оценить режим исполнения всей программы, отдельных блоков или последовательности команд. Вы можете вызвать данную инструкцию на SCL (S7-1200/S7-1500) и на STL (S7-1500).

Примечание

Вы можете найти дополнительную информацию по следующим вопросам:

Как определить время работы программы или отдельных ее блоков в S7-1200/S7-1500 ?
https://support.industry.siemens.com/cs/ww/en/view/87668055

2.10 Символика и комментарии

2.10.1 Редактор программы

Преимущества

При использовании символики и комментариев в Вашей программе, код будет легким для понимания.
Вся символика вместе с программным кодом сохраняется в процессе загрузки программы в контроллер, что позволяет выполнять обслуживание системы, при отсутствующем offline проекте.

Рекомендация

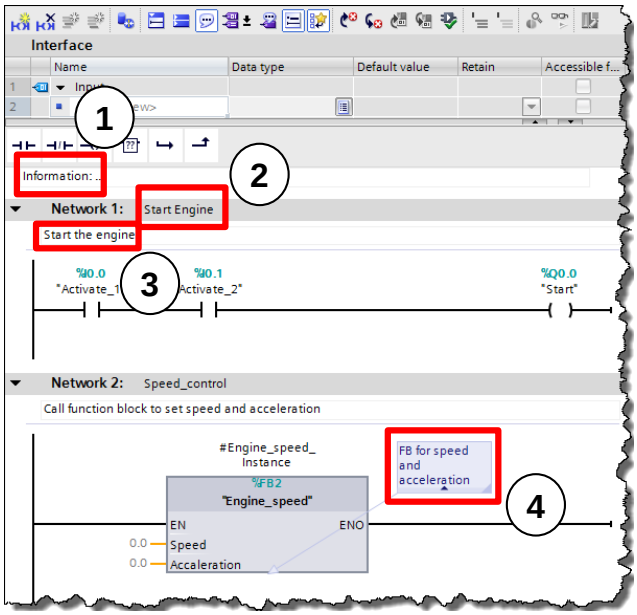
- Используйте комментарии в программах для улучшения читаемости кода программы. Заголовки сегментов видны даже при свернутом сегменте.
- Разрабатывайте программы таким образом, чтобы они были понятны для Ваших коллег.

В следующем примере, Вы можете увидеть вспомогательные инструменты для коментирования программ в редакторе.

Пример

На следующем изображении, Вы можете увидеть варианты создания комметариев в редакторе LAD (также, как для FDB).

Рисунок 2-16: Комментарии в пользовательской программы (LAD)



Возможные варианты комментариев:

1. Комментарий к блоку
2. Заголовок сегмента
3. Комментарий сегмента
4. Комментарий инструкций, блоков и функций (открытие, закрытие, и.т.д.)

На языках программирования SCL и STL, с помощью //, Вы можете закомментировать одну строку.

Пример

```
Filling level := Radius * Radius * PI * height;
// Вычисление уровня заполнения
```

Примечание

Вы можете найти дополнительную информацию по следующим вопросам:

Почему в STEP 7 (TIA Portal), отображаемые тексты, заголовки и комментарии не отображаются после открытия проекта в редакторе?
https://support.industry.siemens.com/cs/ww/en/view/41995518

2.10.2 Комментарии в таблице наблюдений

Преимущества

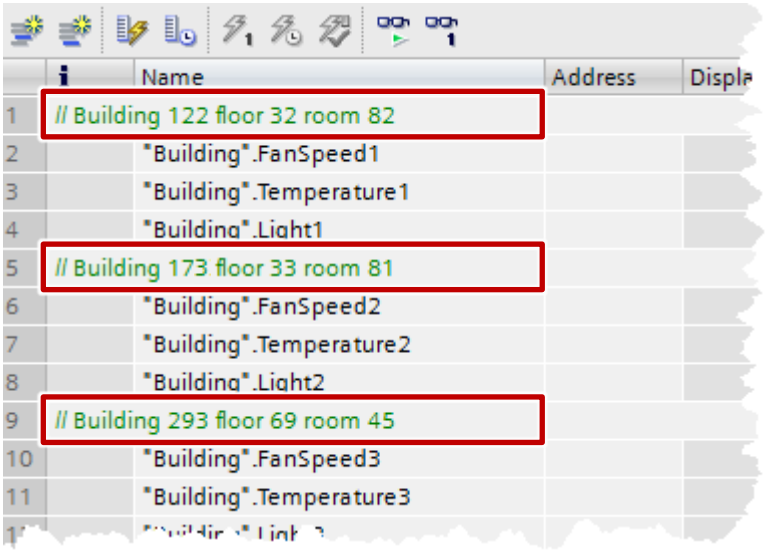
- В таблице наблюдений, возможно также создание комментариев для получения более структурированного вида.

Рекомендация

- Всегда используйте символы комментариев для структурирования Вашей таблицы наблюдения.
- По возможности, давайте комментарии каждому тегу

Пример

Figure 2-17: Watch table with comment lines



2.11 Системные константы

В контроллерах S7-300/400 идентификация аппаратных и программных компонентов выполняется при назначении логического или диагностического адресов.

В S7-1200/1500 идентификация выполняется при помощи системных констант. У всех аппаратных и программных компонентов (например, интерфейсы, модули, OB, ...) контроллеров S7-1200/1500 имеются свои системные константы. Системные константы автоматически создаются при проектировании конфигурации устройства для центральной и распределенной периферии.

Преимущества

- Вы можете получить доступ через имена модуля вместо аппаратной идентификации.

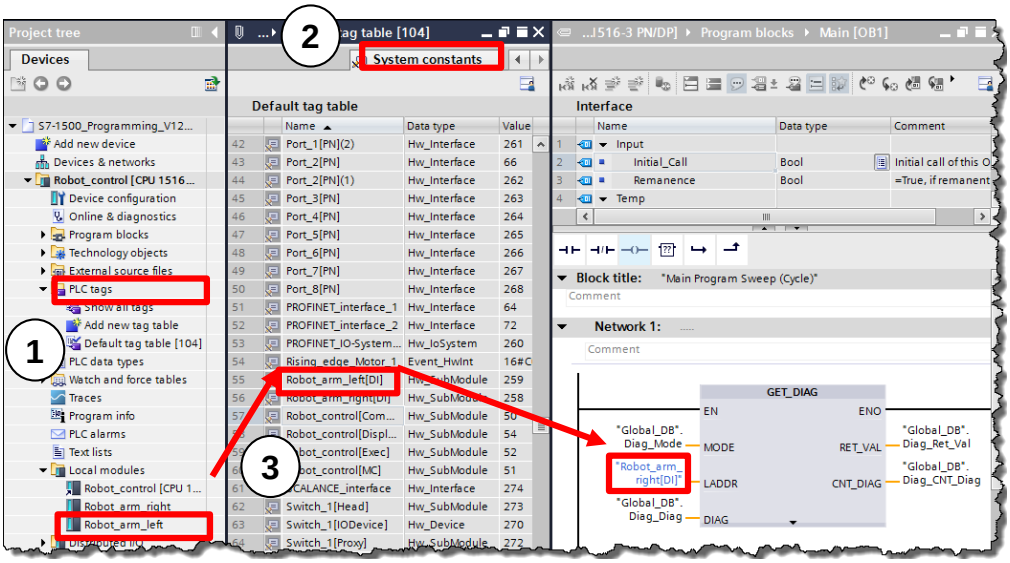
Рекомендация

- Назначьте имя модулю согласно его предназначению, для облегчения создания программы.

Пример

В следующем примере, Вы можете увидеть, как используются системные константы в пользовательской программе.

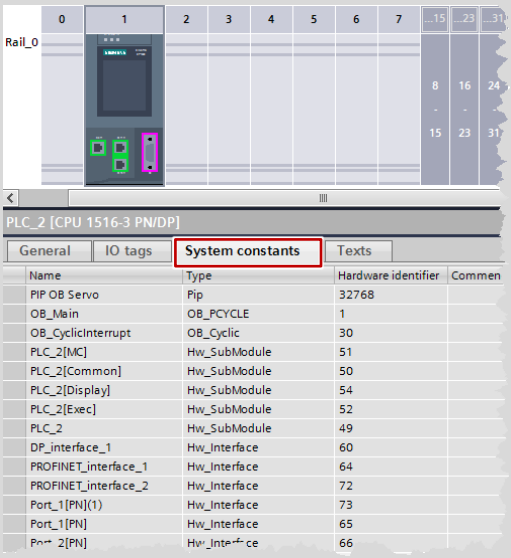
Рисунок 2-18: “Системные константы” в пользовательской программе



1. Системные константы контроллера можно найти в каталоге “PLC tags –Default tag table” (Теги PLC - Таблица тегов по умолчанию).
2. Системные константы находятся в отдельной вкладке в “Default tag table” (Таблица тегов по умолчанию).
3. В данном примере, символьное имя “Robot_arm_left” было назначено для модуля DI.
   Вы также можете найти модуль в таблице системных констант. В пользовательской программе “Robot_arm_left” взаимосвязан с входом диагностического блока “GET_DIAG” .

Примечание

Откройте редактор “Device configuration” (Конфигурация устройства), для быстрого поиска системных констант для каждого устройства.



Name	Type	Hardware identifier	Commen
PIP OB Servo	Pip	32768	
OB_Main	OB_PCYCLE	1	
OB_CyclicInterrupt	OB_Cyclic	30	
PLC_2[MC]	Hw_SubModule	51	
PLC_2[Common]	Hw_SubModule	50	
PLC_2[Display]	Hw_SubModule	54	
PLC_2[Exec]	Hw_SubModule	52	
PLC_2	Hw_SubModule	49	
DP_interface_1	Hw_Interface	60	
PROFINET_interface_1	Hw_Interface	64	
PROFINET_interface_2	Hw_Interface	72	
Port_1[PN](1)	Hw_Interface	73	
Port_1[PN]	Hw_Interface	65	
Port_2[PN]	Hw_Interface	66	

Примечание

Вы можете найти дополнительную информацию по следующим вопросам:

Для чего нужны системные константы в S7-1200/1500 в STEP 7 (TIA Portal)?
https://support.industry.siemens.com/cs/ww/en/view/78782835

2.12 Пользовательские константы

С помощью пользовательских констант, Вы можете создавать свои постоянные значения. В основном, в пользовательской программе используются локальные константы для OB, FC, FB и также глобальные константы для всей программы контроллера.

Преимущества

- Пользовательские константы могут использоваться для изменения постоянных значений для всей программы глобально или локально.
- С помощью пользовательских констант, программа может быть написана в более читаемом виде.

Свойства

- Локальные пользовательские константы объявляются в интерфейсе блока.
- Глобальные пользовательские константы объявляются в “PLC tags” (Теги PLC).
- Пользовательская программа может только считывать пользовательские константы.
- Для защищенных блоков (со свойством know-how protected), пользовательские константы скрыты.

Рекомендация

- Используйте пользовательские константы для улучшения читаемости и гибкости программы …
  - кодов ошибок,
  - CASE инструкций,
  - коэффициентов преобразования,
  - натуральных констант ...

Пример

Рисунок 2-19: Локальные пользовательские константы блока для инструкций CASE

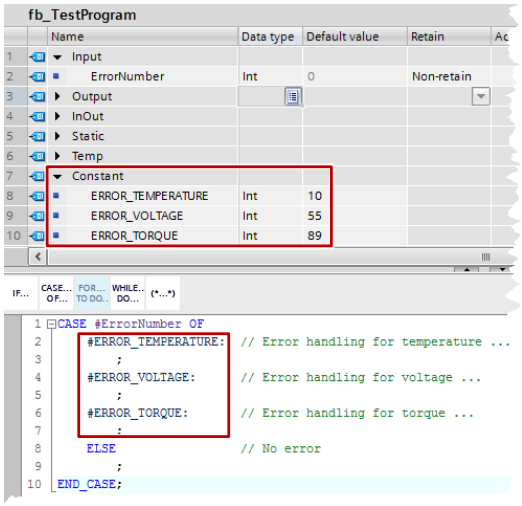
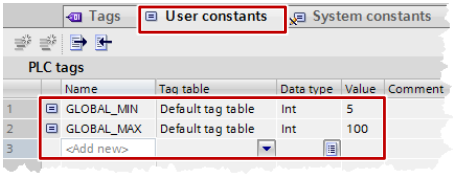


Рисунок 2-20: Глобальные пользовательские константы контроллера



Примечание

Более подробная информация по использованию пользовательских констант описана в FAQ:
Каким образом можно преобразовать тег в STEP 7 (TIA Portal)?
https://support.industry.siemens.com/cs/ww/en/view/61928891

2.13 Внутренний ссылочный ID для тегов контроллера и тегов HMI

STEP 7, WinCC, Startdrive, Safety и другие - интегрированы в общую базу среды разработки TIA Portal . Изменение данных автоматически отразится на других областях пользовательской программы, независимо от того, где было выполнено изменение: контроллер, панель или привод. Этим обеспечивается целостность данных.

При создании тега, TIA Portal автоматически создает уникальный ссылочный ID. Вы не можете увидеть или запрограммировать данный ID. Данная процедура является внутренней. При изменении тегов (адрес), ссылочный ID остается неизменным.

2.13 Внутренний ссылочный ID для тегов контроллера и тегов HMI

На изображении, ниже схематично показано соответствие ссылочному ID некоторой информации.

Рисунок 2-21: Внутренний ссылочный ID для PLC и HMI

PLC_1 **HMI_1**

Символьное имя PLC	Абсолютный адрес	Внутренний ID PLC	Внутренний ID HMI	Символьное имя HMI	Тип доступа	Соединение с PLC
Motor_1	I0.0	000123	009876	Motor_1	<symbolic access>	PLC_1
Valve_2	Q0.3	000138	000578	Valve_2	<symbolic access>	PLC_1

Примечание

ID будет изменен, если …

- будет изменено имя
- будет изменен тип
- будет удален тег

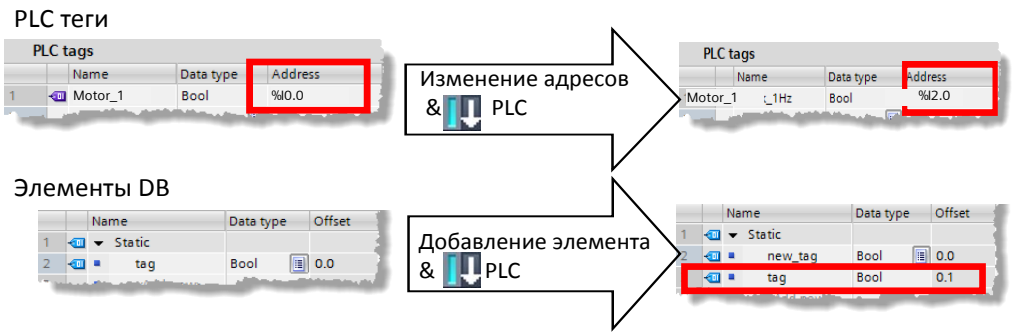
Преимущества

- Вы можете переназначить теги без изменения внутренних связей. Коммуникация между контроллером, HMI и приводом останется неизменной.
- Длина символьного имени не влияет на коммуникационную нагрузку между контроллером и HMI.

Свойства

Если Вы измените адреса тегов PLC, Вам необходимо будет перезагрузить контроллер. Нет необходимости в перезагрузке устройств HMI, благодаря внутренним системным адресам со ссылочными ID (см. Рисунок 2-22: Изменение адресов или добавление строки).

Рисунок 2-22: Изменение адресов или добавление строки



2.14 Режим STOP в случае возникновения ошибок

По сравнению с S7-300/400, S7-1200/1500 переходят в режим “STOP” лишь в нескольких случаях.

Благодаря консистентной проверке в TIA Portal, переход в режим “STOP” для контроллеров S7-1200/1500 во многих случаях может быть исключен. Проверка программных блоков на констистентность выполняется при компиляции в TIA Portal. Данный механизм исключает остановку контроллеров S7-1200/1500 при возникновении некоторых ошибок, в отличие от их предшественников.

Преимущества

Контроллеры S7-1200/1500 переходят в режим STOP только в трех случаях. Что упрощает программирование обработки ошибок.

Свойства

Таблица 2-18: Реакция на ошибки S7-1200/1500

	Ошибка	S7-1200	S7-1500
1.	Однократное превышение времени цикла	RUN	**STOP**, если OB80 отсутствует
2.	Двукратное превышение времени цикла	**STOP**	**STOP**
3.	Ошибка программирования	RUN	**STOP**, если OB121 отсутствует

OB обработки ошибок:

- OB80 “Time error interrupt” (Превышение времени цикла) вызывается операционной системой, когда было превышено максимальное время цикла.
- OB121 “Programming error” (Ошибка программирования) вызывается операционной системой, при возникновении ошибки в программе.

При каждом возникновении ошибки, автоматически производится запись в диагностический буфер.

Примечание

В контроллерах S7-1200/1500 имеются также другие OB обработки ошибок (диагностическое событие, отказ стойки, и.т.д.).

Более подробную информацию по реакции на ошибки в S7-1200/1500, Вы можете найти в online помощи TIA Portal в разделе “Events and OB” (События и OB).

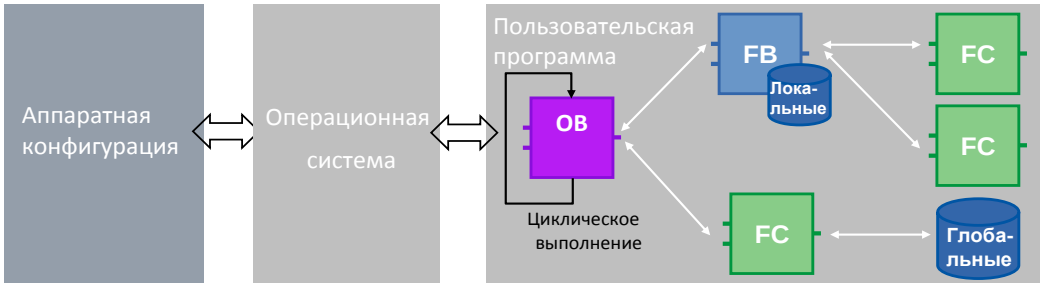
3 Введение в программирование

3.1 Операционная система и пользовательская программа

Контроллеры SIMATIC содержат операционную систему и программу пользователя.

- Операционная система управляет всеми функциями и процессами в контроллере, которые не связаны с определенной задачей управления (например, обработка рестарта, обновление образа процесса, вызов пользовательской программы, обработка ошибок, управление памятью, и.т.д). Операционная система - встроенная часть контроллера.
- Пользовательская программа состоит из блоков, которые необходимы для реализации задачи автоматизации. Пользовательская программа состоит из блоков и в дальнейшем загружается в контроллер.

Рисунок 3-1: Операционная система и пользовательская программа



Для контроллеров SIMATIC, пользовательская программа всегда выполняется циклически. Циклический OB (“Main”) уже имеется в папке “Program blocks” (Программные блоки) после добавления контроллера в STEP 7. Блок обрабатывается и вызывается контроллером в бесконечном цикле .

3.2 Программные блоки

В STEP 7 (TIA Portal) типы блоков остались от предыдущих версий STEP 7:

- Организационные блоки
- Функциональные блоки
- Функции
- Блоки данных

Опытные пользователи STEP 7 наверняка уже с ними знакомы, а новичкам будет легко их осваивать.

Преимущества

- Вы можете структурировать свою программу различными типами блоков.
- При структурном программирования, Вы получаете возможность работы с блоками многократного использования внутри или вне данного проекта. Такие блоки отличаются только набором параметров (см. главу 3.2.8 Повторное использование блоков).
- Ваш проект или установка становятся более прозрачными. Ошибки в станции можно легко обнаружить, проанализировать и устранить. Сопровождение и корректировка Вашей программы становится проще.

Рекомендации

- Структурируйте Вашу задачу автоматизации.
- Разбейте общий функционал линии на отдельные функциональные узлы

  Разделяйте данные функциональные узлы на еще меньшие процедуры до тех пор, пока не сможете реализовать функцию с многократным вызовом с разными параметрами.
- Определите интерфейсы между функциональными узлами. Определяйте уникальные интерфейсы для функционала, который поставляется “сторонними компаниями”.

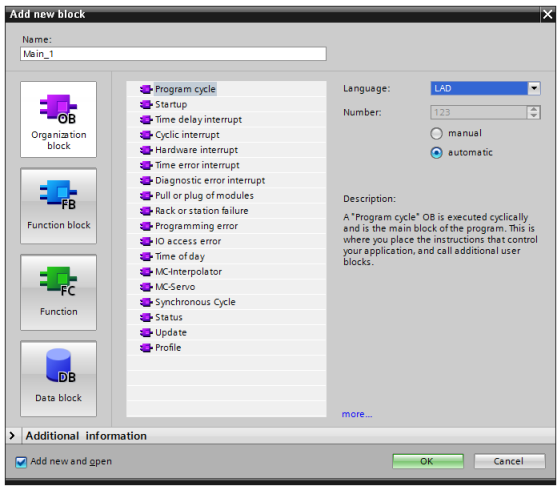
Все организационные блоки, функциональные блоки и функции могут быть запрограммированы на следующих языках:

Таблица 3-1: Языки программирования

Язык программирования	S7-1200	S7-1500
Ladder (LAD)	✓	✓
Function block diagram (FBD)	✓	✓
Structured control language (SCL)	✓	✓
Graph	✗	✓
Statement list (STL)	✗	✓

3.2.1 Организационные блоки (OB)

Рисунок 3-2: Окно “Add new block” (Добавление нового блока) (OB)



Организационные блоки (OB) являются интерфейсом между операционной системой и программой пользователя. Они вызываются операционной системой и управляют следующими процессами:

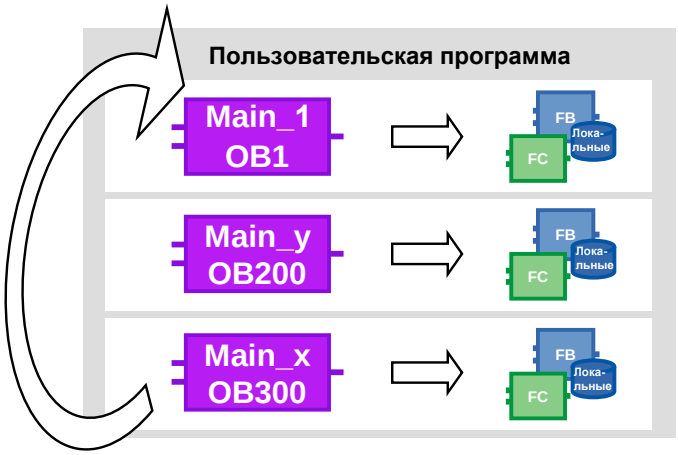
- Поведение при запуске контроллера
- Циклическая обработка программы
- Обработка прерываний в программе
- Обработка ошибок

В зависимости от типа контроллера доступно различное количество типов OB.

Свойства

- OB вызываются операционной системой контроллера.
- В программе могут быть созданы несколько главных OB (Main). Такие OB последовательно обрабатываются в порядке возрастания их номеров OB.

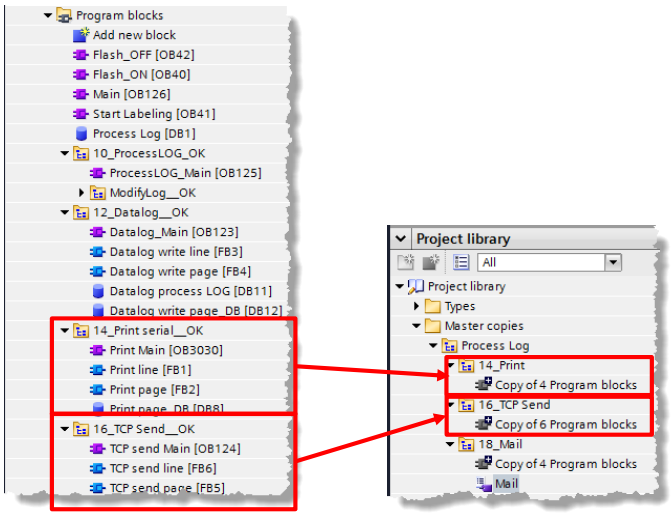
Рисунок 3-3: Использование нескольких главных OB (Main)



Рекомендации

- Разнесите вызовы различных частей подпрограмм, которые могут переноситься с одного контроллера на другой на несколько главных OB (Main).
- Избегайте связей между несколькими главными OB (Main). Они должны быть использованы независимо друг от друга. Если Вы все же производите обмен данными между главными OB, используйте для этого глобальные DB (см. главу 4.2 Переход от меркерной области к глобальным блокам данных).
- Разделяйте все части программы, которые взаимодействуют друг с другом на отдельные папки и сохраняйте их для повторного использования в проектной или глобальной библиотеке.

Рисунок 3-4: Сохранение частей программы в библиотеке проекта



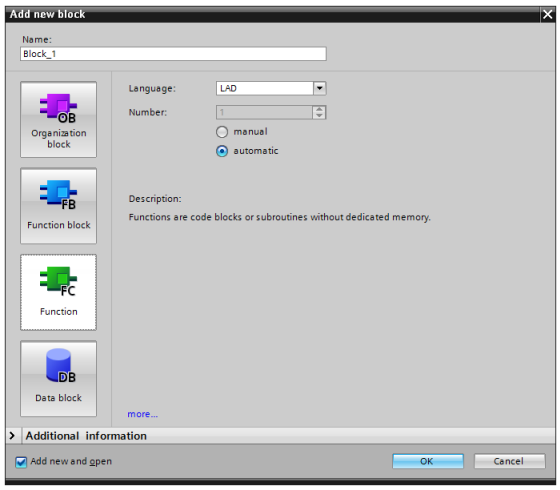
Для получения более подробной информации, обратитесь к главе 3.7 Библиотеки.

Примечание

Вы можете найти дополнительную информацию по следующим вопросам:
Какие организационные блоки могут быть использованы в STEP 7 (TIA Portal) ?
https://support.industry.siemens.com/cs/ww/en/view/40654862

3.2.2 Функции (FC)

Рисунок 3-5: Окно “Add new block” (Добавление нового блока) (FC)



Функции (FC) это блоки без памяти. Именно поэтому, значения параметров блока сохранены до следующего вызова блока.

Свойства

- FC это блоки циклического сохранения.
- Временные теги не определены, при вызове неоптимизированных блоков. В оптимизированных блоках, теги всегда получают значения по умолчанию (S7-1500 и S7-1200, начиная с Firmware V4). Таким образом, такое поведение предотвращает возникновение ошибок.
- Для сохранения данных в FC, возможно использование глобальных блоков данных.
- У FC могут быть несколько выходов .
- Значение функции может быть повторно использовано на SCL в качестве формулы.

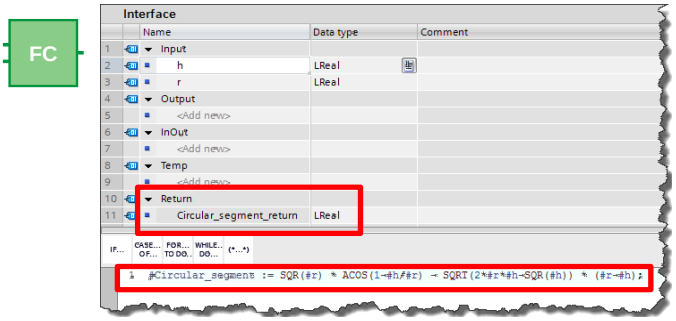
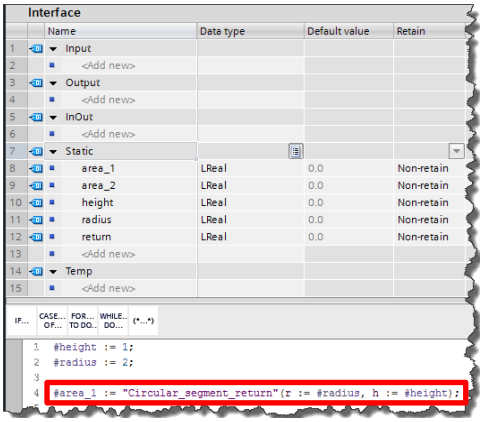
Рекомендация

- Используйте функции для приложений, которые несколько раз вызываются в разных частях программы.
- Для повторного использования значения функции, используйте SCL.
  `<Операнд> := <имя FC> (список параметров);`

Пример

В следующем примере, в FC запрограммировано математическое выражение. Результат вычисления описан как возвращаемое значение, которое в дальнейшем может быть использовано повторно.

Таблица 3-2: Повторное использование значения в функции

Шаг	Инструкция
4.	Создайте FC с математическим выражением (circular сегмент) и определите значение “Return”, как результат выражения. 
5.	Выполните вызов FC для циклического вычисления в сегменте любого блока (SCL). `<Операнд> := <имя FC> (список параметров);` 

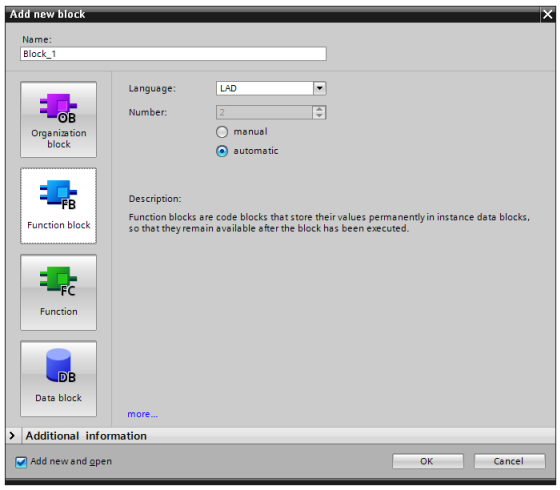
Примечание

Вы можете найти дополнительную информацию по следующим вопросам:

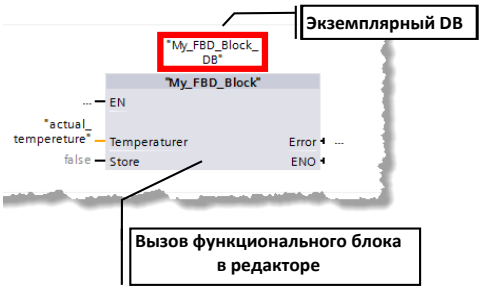
Какое максимальное количество параметров возможно определить для функции в STEP 7 (TIA Portal) в S7-1200/S7-1500 CPU?
https://support.industry.siemens.com/cs/ww/en/view/99412890

3.2.3 Функциональные блоки (FB)

Рисунок 3-6: Окно “Add new block” (Добавление нового блока) (FB)



Функциональные блоки (FB) это блоки с циклическим сохранением данных, значения которых будут доступны при следующем вызове блока. Хранение данных реализуется в экземплярном блоке данных.



Свойства

- FB это блоки с циклическим сохранением данных.
- Временные теги неопределены, при вызове неоптимизированных блоков. В оптимизированных блоках, теги всегда получают значения по умолчанию (S7-1500 и S7-1200, начиная с Firmware V4). Таким образом, такое поведение предотвращает возникновение ошибок.
- Статические переменные сохраняют свои значения до следующего вызова блока.

Рекомендация

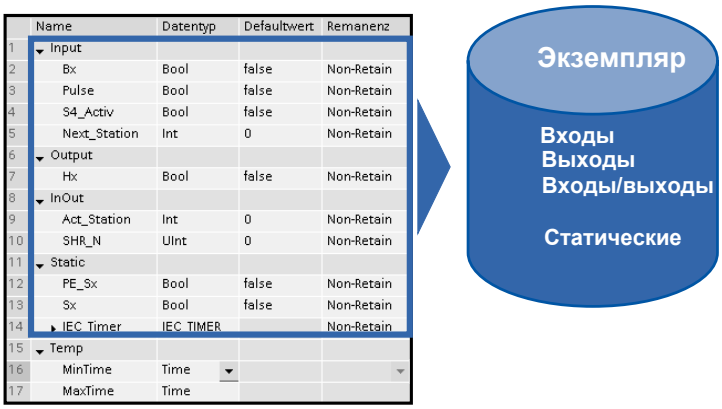
- Используйте функциональные блоки для создания структурированной пользовательской программы. Функциональный блок также может быть вызван несколько раз в различных частях пользовательской программы. Данный механизм упрощает программирование повторно вызываемых элементов программы.
- Если функциональные блоки вызываются несколько раз в программе, используйте отдельные экземплярные блоки или лучше мультиэкземпляры.

3.2.4 Экземпляры

При вызове функционального блока необходимо указать экземплярный блок данных. Данные, которые обрабатывает функциональный блок сохраняются в экземплярном DB.

Экземплярные DB всегда создаются в соответствии с описанным интерфейсом FB и отдельные переменные не могут быть изменены в самом экземплярном DB.

Рисунок 3-8: Структура интерфейса FB



В экземплярном DB хранятся входы, выходы, входы/выходы и статические переменные. Временные переменные храняться в L стеке. L стек актуален только для текущего процесса. Т.е. временные переменные должны инициализировать значение в каждом цикле.

Свойства

- При вызове FB всегда необходимо назначать экземплярные DB .
- Экземплярные DB создаются не вручную в TIA Portal, а автоматически при вызове FB.
- Структура экземплярного DB определяется согласно интерфейсу FB и может быть изменена только через него.

Рекомандация

- Создавайте программы таким образом, чтобы данные экземплярного DB могли изменяться только соответствующим FB. Таким образом, Вы гарантируете, что блок будет использован конкретно для решения своей задачи в любых проектах.

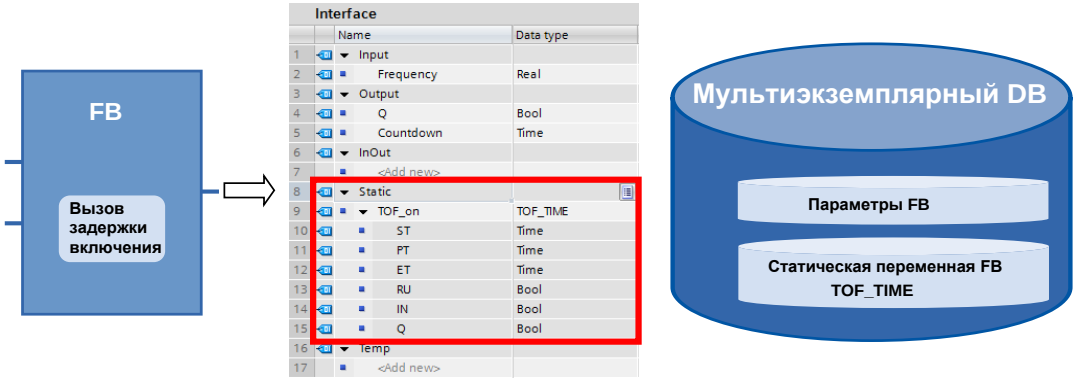
Для получения более подробной информации, обратитесь к главе 3.4 Интерфейс блока для обмена данными.

3.2.5 Мультиэкземпляры

При помощи мультиэкземпляров, вызываемые функциональные блоки могут сохранять свою информацию в экземплярном блоке вызывающего функционального блока. Т.е. если один функциональный блок вызывается в другом функциональном блоке, то вся его информация будет сохранена в экземплярном блоке данных FB более высокого уровня. Функционал вызываемого блока остается неизменным.

На следующем изображении показано, как в одном FB используется другой FB (“IEC Timer”). Все данные сохраняются в мультиэкземплярном DB. Таким образом, можно создать блок с независимым поведением по времени, например, тактовый генератор.

Рисунок 3-9: Мультиэкземпляры



Преимущества

- Возможность повторного использования
- Многократный вызов
- Более читаемая программа с меньшим количеством DB
- Простое копирование программ
- Хорошие возможности структурирования программы

Свойства

- Мультиэкземпляры это области памяти внутри экземплярных DB.

Рекомендация

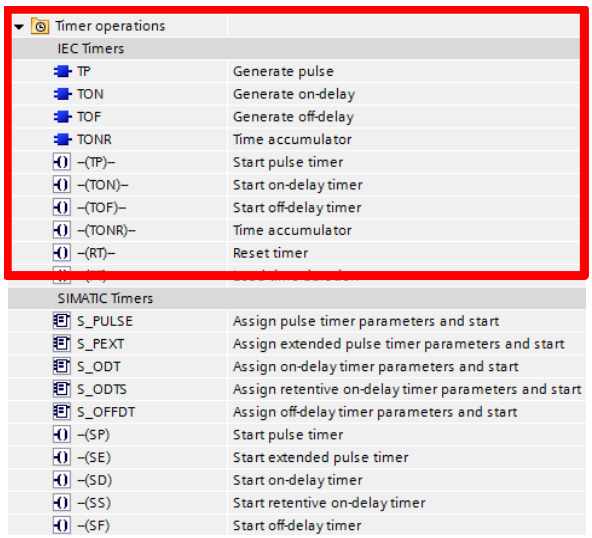
Используйте мультиэкземпляры для …

- уменьшения количества экземплярных DB.
- создания легко читаемых программ с возможностью повторного использования.
- программирования локальных функций, например, таймер, счетчик, выделение фронта.

Пример

Если Вам требуются функции таймеров или счетчиков, используйте блоки “IEC Timer” и “IEC Counter” вместо SIMATIC таймеров и счетчиков с абсолютной адресацией. Если имеется возможность, рекомендуется использовать мультиэкземпляры. Данный механизм позволяет сократить количество блоков данных в пользовательской программе до минимума.

Рисунок 3-10: Библиотека IEC таймеров



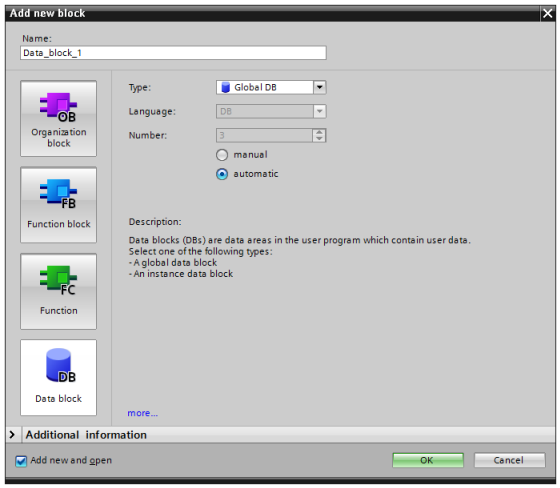
Примечание

Вы можете найти дополнительную информацию по следующим вопросам:

Каким образом объявить таймеры и счетчики для S7-1500 в STEP 7 (TIA Portal) ?
https://support.industry.siemens.com/cs/ww/en/view/67585220

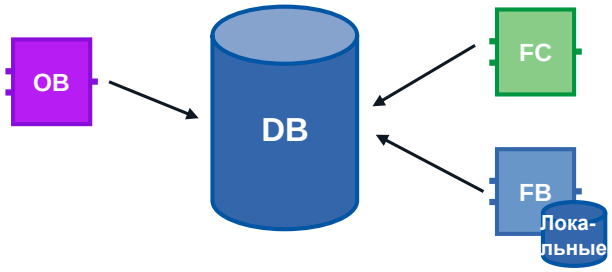
3.2.6 Глобальные блоки данных (DB)

Рисунок 3-11: Окно “Add new block” (Добавление нового блока) (DB)



В блоках данных может находиться различная пользовательская информация, которая может быть использована во всей программе.

Рисунок 3-12: Глобальный DB в качестве основного места хранения данных



Преимущества

- Структурированная область памяти
- Высокая скорость доступа

Свойства

- Все блоки в пользовательской программе могут получить доступ к глобальным DB.
- Структура глобальных DB может состоять из произвольных типов данных.
- Глобальные DB создаются с помощью программного редактора или в соответствии ранее создаваемым “user-defined PLC data type" (Тип данных определенный пользователем) (см. главу Тип данных STRUCT и PLC data types).

Рекомендации

- Используйте глобальные DB, когда данные должны быть обработаны в другой части программы.

Примечание

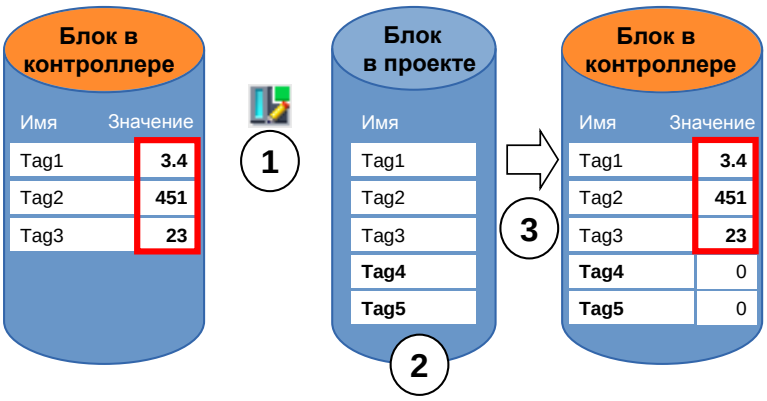
Вы можете найти дополнительную информацию по следующим вопросам:

Какие типы доступа, значения столбцов и режимы работы доступны для глобальных блоков данных в STEP 7?
https://support.industry.siemens.com/cs/ww/en/view/68015630

3.2.7 Загрузка без повторной инициализации

Для того, чтобы изменить пользовательскую программу при уже работающем контроллере, у контроллеров S7-1200 (с версии V4.0) и S7-1500 есть возможность расширения интерфейсов оптимизированных функций или блоков данных в процессе работы. Вы можете загрузить изменные блоки без перехода CPU в режим STOP, при этом фактические значения ранее загруженных переменных не будут изменены.

Рисунок 3-13: Загрузка без повторной инициализации



Если контроллер находится в режиме RUN, выполните следующую процедуру, .

1. Активируйте “Downloading without reinitialization” (Загрузка без повторной инициализации)
2. Добавьте новые переменные в блок данных
3. Загрузите блок в контроллер

Преимущества

- Загрузка новых переменных без прерывания рабочего процесса. Контроллер остается в режиме “RUN”.

Свойства

- Загрузка без повторной инициализации возможна только в оптимизированных блоках.
- Новые переменные будут инициализированы. Значения остальных переменных останутся неизменными.
- Блоку с резервом памяти требуется больше памяти в контроллере.
- Резерв памяти зависит от рабочей памяти контроллера; тем не менее, не более 2 МБ.
- Предполагается, что для блока определен резерв памяти
- По умолчанию, резерв памяти установлен на 100 байт.
- Резерв памяти задается индивидуально для каждого блока.
- Блоки могут быть расширены.

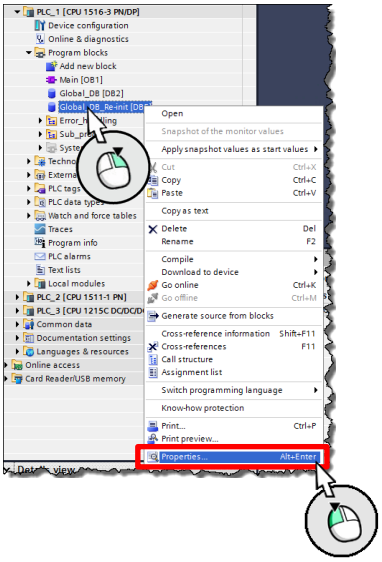
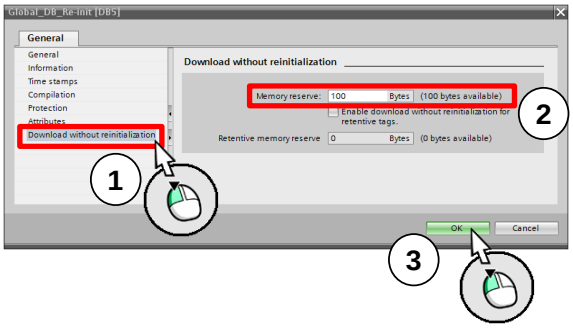
Рекомендация

- Определяйте резерв памяти для блоков, которые будут расширены в дальнейшем в процессе ввода в эксплуатацию (например, тестовые блоки). Процесс ввода в эксплуатацию не будет прерван при загрузке новых переменных. Текущие значения уже созданных переменных не будут изменены.

Пример: Установка резерва памяти для блока

В следующей таблице показан способ назначения резерва памяти для дальнейшей загрузки без повторной инициализации.

Таблица 3-3: Назначение резерва памяти

Шаг	Инструкция
1.	Правой кнопкой мыши выделите любой оптимизированный блок в дереве проекта, далее выберите “Properties” (Свойства). 
2.	 1. Выделите раздел “Download without reinitialization” (Загрузка без повторной инициализации). 2. Назначьте необходимый резерв памяти в “Memory reserve”. 3. Подтвердите, нажав "OK".

Примечание

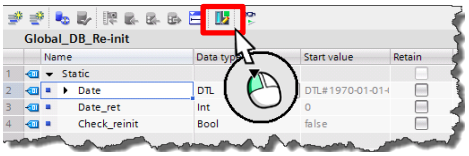
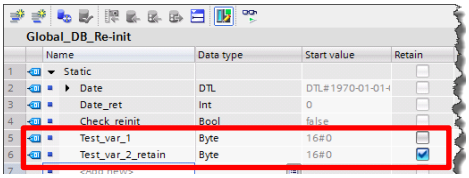
Вы также можете установить значение по умолчанию для размера резерва памяти новых блоков в TIA portal.

В меню, перейдите на "Options – Settings", затем "PLC programming – General – Download without reinitialization“.

Пример: Загрузка без повторной инициализации

На следующем примере показано, как выполнять загрузку без повторной инициализации.

Таблица 3-4 Загрузка без повторной инициализации

Шаг	Инструкция
1.	Требование: должен быть установлен резерв памяти (см. выше)
2.	Откройте оптимизированный глобальный блок данных DB.
3.	Нажмите на кнопку “Download without reinitialization” (Загрузка без повторной инициализации) и подтвердите, нажав “OK” 
4.	Добавьте новую переменную (можно сохраняемую). 
5.	Загрузите блок в контроллер.
6.	Результат: • Фактические значения переменных блока остались прежними

Примечание

Более подробную информацию, Вы можете найти в online помощи TIA Portal в “Loading block extensions without reinitialization” (Загрузка расширений блока без повторной инициализации).

Вы можете найти дополнительную информацию по следующим вопросам:

Какие способы загрузки есть в S7-1500 в режиме RUN?
https://support.industry.siemens.com/cs/ww/en/view/68015630

3.2.8 Возможность повторного использования блоков

Благодаря концепции создания блоков, Вы получаете возможность создавать структурированные и эффективные программы.

Преимущества

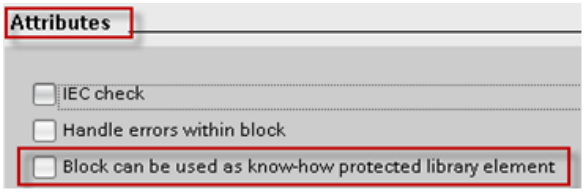
- Блоки могут быть использованы в любом месте программы.
- Блоки могут быть универсально использованы в другом проекте.
- В случае, если каждый блок решает свою независимую задачу, то такой подход является основой структурного программирования.
- Уменьшается количество ошибок.
- Возможна простая диагностика ошибок.

Рекомендация

Если Вы хотите использовать блок повторно, придерживайтесь следующих рекомендаций:

- Воспринимайте блок, как инкапсулированную функцию. т.е, каждый блок представляет из себя, решение той или иной задачи, пользовательской программы.
- Используйте несколько главных OB (Main), для группировки частей системы.
- Всегда выполняйте обмен данными между блоками через собственные интерфейсы, а не через экземпляры (см. главу 3.4.1 Интерфейс блока для обмена данными).
- При работе с блоком не используйте проектных специфичных данных, а также следующие компоненты:
  - Доступ к глобальным DB и использование конкретных экземплярных DB
  - Доступ к тегам
  - Доступ к глобальным константам
- Для блоков с возможностью повторного использования имеется набор требований, так же как к защищенным (know-how-protected) блокам в библиотеках. Поэтому, Вам необходимо проверить наличие свойства у блоков с возможностью повторного вызова“Block can be used as know-how protected library element” (Блок может быть использован, как элемент библиотеки с защитой know-how protected). До проверки скомпилируйте блок .

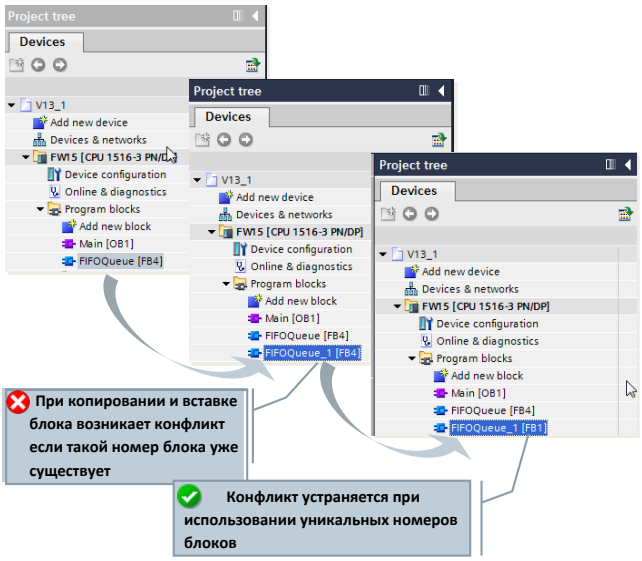
Рисунок 3-14: Атрибуты блока



3.2.9 Автоматическое назначение номеров блокам

Для внутренней обработки, необходимые номера блоков назначаются автоматически системой (настройка в свойствах блока).

Рисунок 3-15: Автоматическое назначение номеров блокам



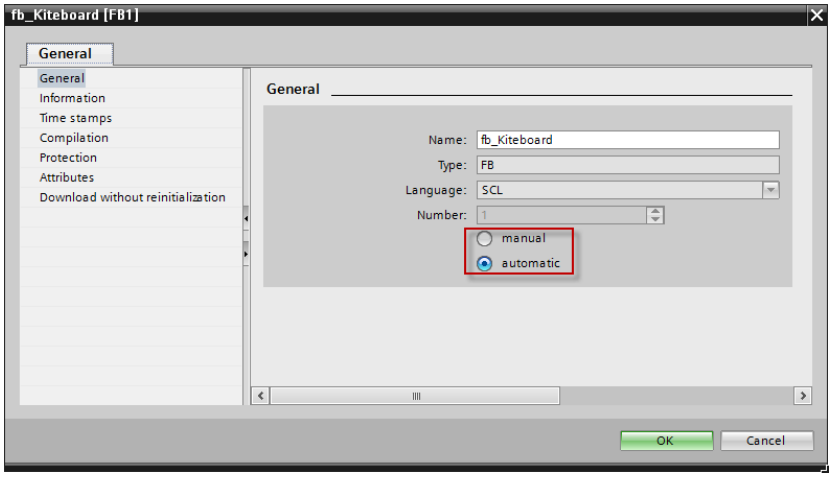
Преимущества

- Конфликт с номерами блоков, например, после копирования, автоматически устраняется в процессе компиляции TIA Portal.

Рекомендация

- Не меняйте текущую настройку “automatic” (автоматически).

Рисунок 3-16: Настройка в свойствах блока



3.3 Типы интерфейса блока

У FB и FC имеется три различных типа в интерфейсе: In (Входной), InOut (Проходной) и Out (Выходной). С помощью данных типов интерфейса, блоки получают параметры. Параметры обрабатываются и результаты выводятся в вызывающий блок. Параметры InOut используются как для передачи данных в вызывающий блок, так и для возврата результата обратно. Имеется два способа передачи для таких параметров.

3.3.1 Задание фактического значения на входной параметр

При вызове блока, значение фактического параметра копируется во входной параметр блока с типом In. Для этого в блоке выделяется дополнительная область памяти.

Рисунок 3-17: Копирование значения во входной параметр

Свойства

- Каждый блок производит одинаковую обработку заданных параметров
- Значения копируются при вызове блока

3.3.2 Задание фактического значения на проходной параметр

При вызове блока, адрес фактического параметра задается на проходной параметр. При этом, дополнительное выделение памяти не требуется.

Рисунок 3-18: Передача на значения по ссылке (указатель на место хранения параметра)

Свойства

- Каждый блок производит одинаковую обработку заданных параметров
- Фактические параметры передаются по ссылке в вызывающий блок

Рекомендация

- Для рационального использования памяти для структурированных переменных (например, ARRAY, STRUCT, STRING,…), используйте проходные параметры .

3.3.3 Варианты передачи параметров

В следующий таблице показано, как в S7-1200/1500 параметры блока могут быть использованы для передачи переменных элементарного и сложного типа.

Таблица 3-5: Варианты передачи параметров

Тип блока / формальный параметр		Элементарный тип данных	Сложный тип данных
FC	Входной	копия	**по ссылке**
	Выходной	копия	**по ссылке**
	Проходной	копия	**по ссылке**
FB	Входной	копия	копия
	Выходной	копия	копия
	Проходной	копия	**по ссылке**

Примечание

Если при вызове блока, выполняется передача оптимизированных данных со свойством “standard access” (стадартный доступ), то такие данные передаются в качестве копии. Если блок содержит много параметров сложного типа, то может возникнуть переполнение области временных переменных (локальный стек) .

Это можно предотвратить, создав один и тот тип доступа для обоих блоков (см. главу 2.6.5 Передача параметров между блоками с оптимизированным и стандартным доступом).

3.4 Принцип хранения

В STEP 7 имеется различие между глобальной и локальной областью памяти. Глобальная область памяти доступна для любого блока в пользовательской программе. Локальная область памяти доступна только в соответствующем блоке.

3.4.1 Интерфейсы блоков для обмена данными

Если Вы "инкапсулируете" функции и программы, то обмен данными между блоками должен быть выполнен только через их интерфейсы, что дает Вам преимущества в их использовании.

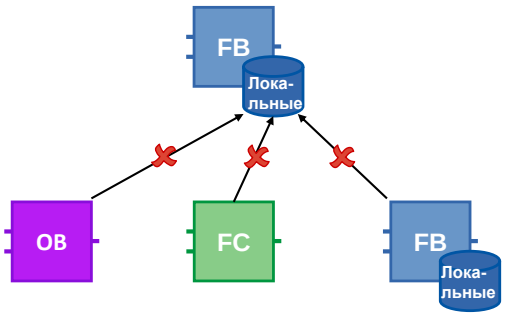
Преимущества

- Программа может состоять из отдельных блоков, каждый из которых решает свою задачу.
- Программа может быть легко расширена и введена в работу.
- Программный код легко читается, так как отсутствует скрытый перекрестный доступ

Рекомендация

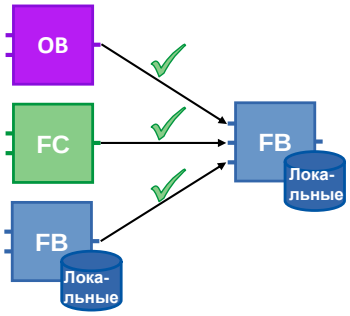
- По возможности, используйте только локальные переменные. В данном случае блок может быть универсальным и использоваться многократно.
- Выполняйте обмен данными через интерфейс блоков (In, Out, InOut), для возможности многократного использования их.
- В качестве локальной памяти используйте только экземплярные блоки для соотвествущих функциональных блоков. Остальные блоки не должны записывать что-либо в экземплярные блоки данных.

Рисунок 3-19: Предотвращение доступа к экземплярным блокам данных



Если для обмена данными используется только интерфейс блока, то это гарантирует, что все блоки могут функционировать независимо друг от друга.

Рисунок 3-20: Интерфейсы блока для обмена данными



3.4.2 Глобальная область памяти

Область памяти называется глобальной если любая часть пользовательской программы может получить к ней доступ. Имеются аппаратно-зависимые области памяти (например, меркерная память, таймеры, счетчики, и.т.д.) и глобальные DB. При работе с аппаратно-зависимой областью памяти следует помнить, что программа может быть неработоспособной на других контроллерах, так как некоторые области памяти могут быть уже заняты. Поэтому, Вы должны использовать глобальные DB вместо аппаратно-зависимых областей памяти.

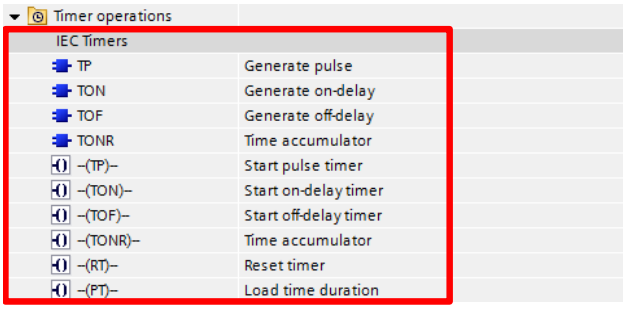
Преимущества

- Пользовательские программы могут использованы универсально и независимо от аппаратной конфигурации.
- Пользовательская программа может быть структурирована без разделения областей меркерной памяти для различных пользователей.
- Оптимизированные глобальные DB гораздо эффективнее чем меркерная область памяти, которая не оптимзирована по причине необходимости её совместимости.

Рекомендация

- Не используйте меркерную память, работайте с глобальными DB.
- Не работайте с аппаратно-зависимой памятью, например, с синхробайтом или счетчиком. Используйте IEC счетчики и таймеры с мультиэкземплярами (см. главу 3.2.5 Мультиэкземпляры). IEC таймеры можно найти в “Instructions – Basic Instructions – Timer operations” (Инструкции - Основные инструкции -Таймеры).

Рисунок 3-21: IEC таймеры



3.4.3 Локальная область памяти

- Статические переменные
- Временные переменные

Рекомендация

- Используйте статические переменные для значений, которые потребуются в следующем цикле.
- Используйте временные переменные в качестве кэш памяти для текущего цикла. Время доступа к временным переменным меньше, чем к статическим переменным.

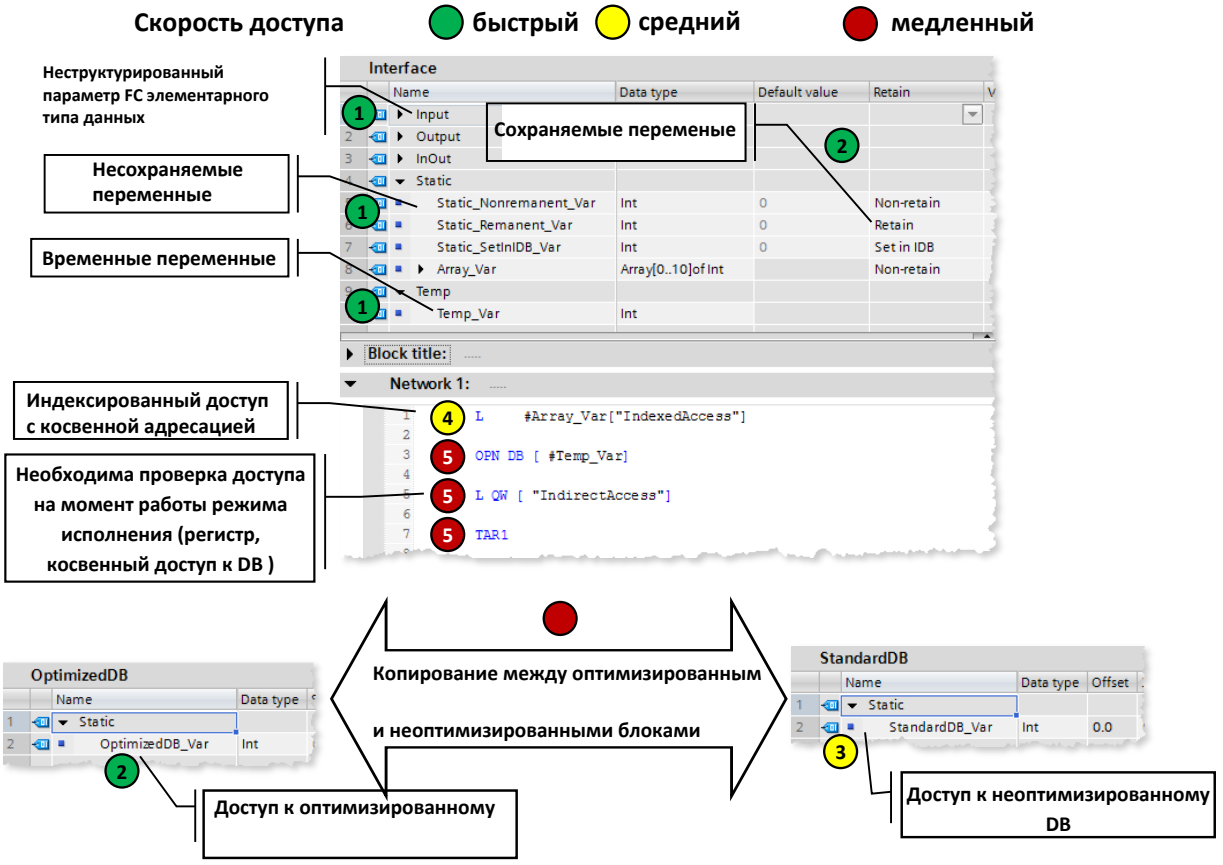
Примечание

Оптимизированные блоки: Временные переменные инициализируются при каждом вызове блока “значением по умолчанию” (S7-1500 и S7-1200, начиная с V4).
Неоптимизированные блоки: Временные переменные имеют неопределенные значения при каждом вызове блока.

3.4.4 Скорость доступа к областям памяти

В STEP 7 имеется несколько вариантов доступа к памяти. По системным причинам, один тип доступа может быть быстрым, другой медленным.

Рисунок 3-22: Различные способы доступа к памяти



Наиболее быстрый тип доступа в S7-1200/1500 в порядке возрастания времени доступа:

1. Оптимизированные блоки: Временные переменные, параметры FC и FB, несохраняемые статические переменные
2. Оптимизированные блоки с доступом к:
   - Сохраняемым переменным FB
   - Оптимизированным глобальным DB
3. Доступ к неоптимизированным блокам
4. Индексированный доступ, с индексом, рассчитываемым во время режима исполнения (например, `Motor [i]`)
5. Доступ с проверкой в режиме исполнения
   - Доступ к DB, которые создаются в процессе работы режима исполнения или открываются с косвенной адресацией (например, OPN DB[i])
   - Доступ к регистрам или косвенный доступ к памяти
6. Копирование структур между оптимизированными и неоптимизированными блоками (кроме байтового массива)

3.5 Сохраняемость

В случае сбоя питания, контроллер с помощью его резервной энергии копирует сохраняемые данные из рабочей памяти в энергонезависимую память. После рестарта контроллера, обработка программы продолжается с сохраняемыми данными. В зависимости от контроллера, доступны различные обьемы сохраняемой памяти.

Таблица 3-6: Сохраняемая память в S7-1200/1500

Контроллер	**Используемая сохраняемая память для меркеров, таймеров, счетчиков, DB и технологических обьектов**
CPU 1211C,1212C, 1214C, 1215C, 1217C	10 Кбайт
CPU 1511-1 PN	88 Кбайт
CPU 1513-1 PN	88 Кбайт
CPU 1515-2 PN, 1516-3 PN/DP	472 Кбайт
CPU 1518-4 PN/DP	768 Кбайт

Таблица 3-7: Различия S7-1200 и S7-1500

S7-1200	**S7-1500**
Сохраняемость может быть установлена **только** для меркеров	Сохраняемость может быть установлена для меркеров, счетчиков и таймеров

Преимущества

- Сохраняемые данные запоминают свои значения при переходе контроллера в режим STOP и обратно в RUN или в случае сбоя питания и рестарта контроллера.

Свойства

Для переменных с элементарным типом данных оптимизированного DB, сохраняемость может быть настроена индивидуально для каждой переменной. Неоптимизированные блоки данных могут быть или полностью сохраняемыми или полностью несохраняемыми.

Сохраняемые данные могут быть удалены с помощью "memory reset" (сброс памяти) или "Reset to factory settings" (Сброс на заводские установки):

- Переключение режима на контроллере (MRES)
- Дисплей контроллера
- Online с помощью STEP 7 (TIA Portal)

Рекомендация

- Не используйте свойство “Set in IDB” (Назначается в экземплярном блоке данных). Всегда назначайте сохраняемость в функциональном блоке, а не в экземплярном блоке данных.
  Свойство “Set in IDB” (Назначается в экземплярном блоке данных) увеличивает время обработки программы. Для интерфейсов FB, всегда выбирайте
  “Non-retain” (Несохранямый) или “Retain” (Сохраняемый).

Рисунок 3-23: Программный редактор (Интерфейс функционального блока)

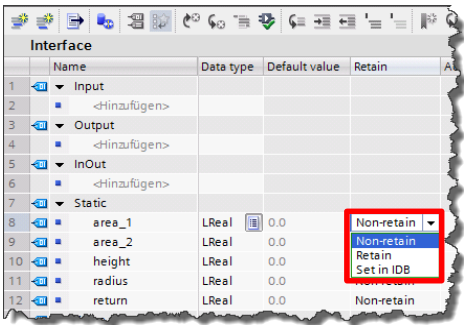
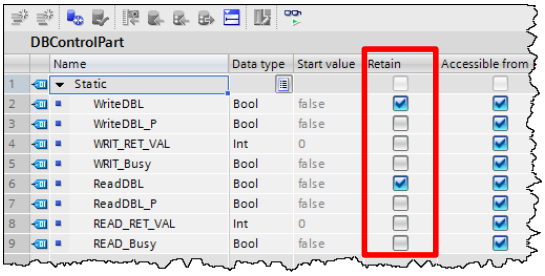


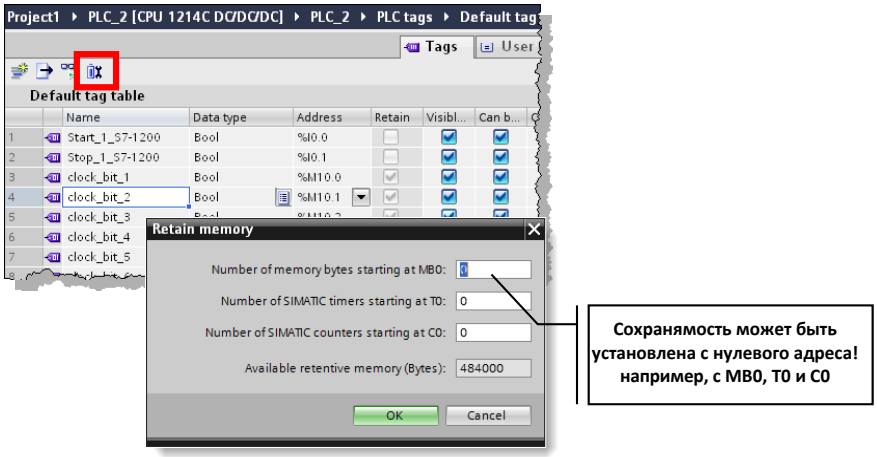
Рисунок 3-24: Программный редактор (Блок данных)



Пример: Сохраняемость тегов PLC

Настройка сохранямости данных выполняется в таблицах тегов PLC, функциональных блоках или блоках данных.

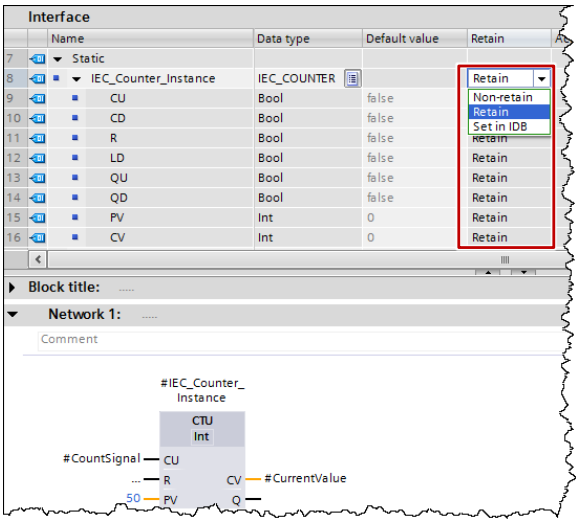
Рисунок 3-25: Настройка сохранямости переменных в таблице тегов PLC



Пример: Сохраняемый счетчик

Вы также можете обьявить экземпляры или функции (таймер, счетчик, и.т.д.), как сохраняемые. Как было ранее описано в главе 3.2.5 Мультиэкземпляры

Рисунок 3-26: Сохраняемый счетчик в качестве мультиэкземпляра



Примечание

Если недостаточно сохраняемой памяти PLC, то имеет смысл сохранять данные в виде блоков данных, которые находятся только в загрузочной памяти PLC. Данный способ описан в качестве примера для S7-1200. Также работает на S7-1500.

Вы можете найти дополнительную информацию по следующему вопросу:

Как сконфигурировать блок данных с атрибутом "Only store in load memory" (Сохраняется только в загрузочной памяти) в STEP 7 (TIA Portal) для S7-1200?
https://support.industry.siemens.com/cs/ww/en/view/53034113

3.6 Символьная адресация

3.6.1 Символьная адресация вместо абсолютной адресации

TIA Portal оптимизирован под использование символьной адресации. При этом, Вы получаете множество преимуществ. При работе с символьной адресацией, Вы можете создавать программы, не обращая внимания на внутреннюю структуру блоков. Контроллер сам определяет оптимальный вариант для хранения данных. Таким образом, Вы можете полностью сосредоточиться на поставленной задаче.

Преимущества

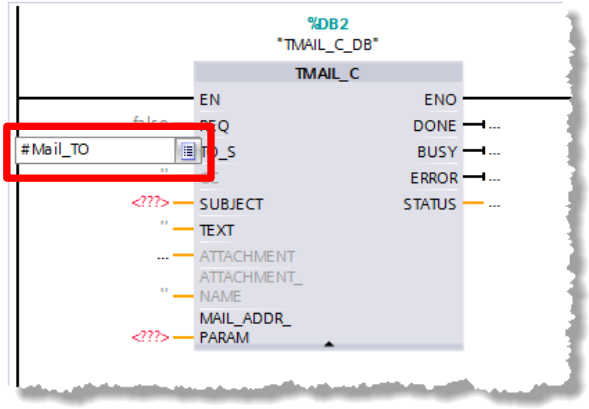
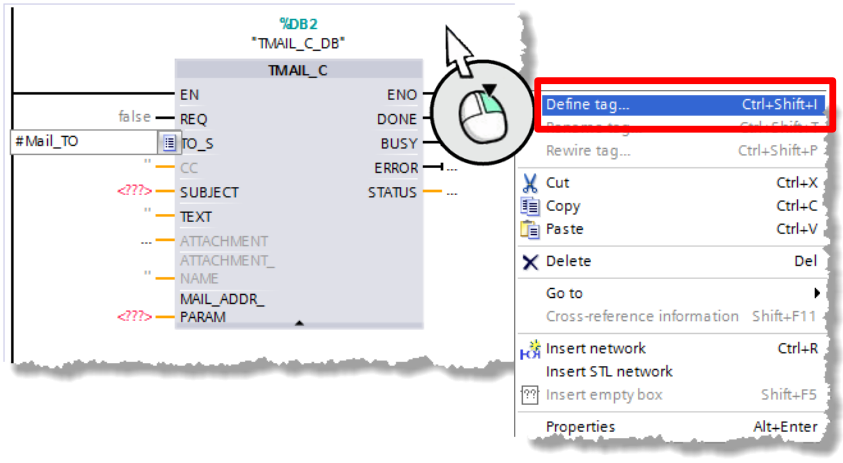
- Читаемость программ, благодаря символьным именам переменных
- Автоматическое обновление имен переменных во всех местах программы
- Управление хранением данных в программе не требует управления вручную (абсолютная адресация)
- Мощный механизм доступа к данным
- Не требуется ручная оптимизация для повышения быстродействия или уменьшения размера программы
- IntelliSense помогает выполнять быстрый ввод переменных
- Меньшее количество программных ошибок, благодаря проверке типа (проверка типов данных выполняется для всех типов доступа)

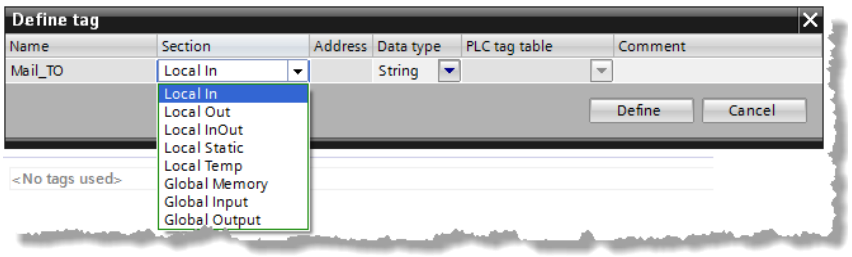
Рекомендация

- “Не требуется знания структуры хранения данных“
- “Думайте” символьно. Определяйте “соответствующее” имя для каждой функции, переменной, например, Pump_boiler_1, heater_room_4, и.т.д. Таким образом, программа будет читаема даже при отсутствии комментариев.
- Назначайте всем используемым переменным символьное имя при помощи правой кнопки мыши.

Пример

Таблица 3-8: Пример создания символьных переменных

Шаг	Инструкция
1.	Откройте программный редактор и откройте любой блок.
2.	Введите символьное имя непосредственно на вход инструкции. 
3.	Нажмите правой кнопкой мыши на блоке и выберите “Define tag…” (Определить тег). 

Шаг	Инструкция
4.	Определите тег. 

Есть очень эффективный способ, который сохранит Вам время, если Вы хотите определить несколько переменных в сегменте. В первую очередь назначьте всем переменным имена. После этого, определите все переменные одновременно, с помощью шага 4.

Примечание

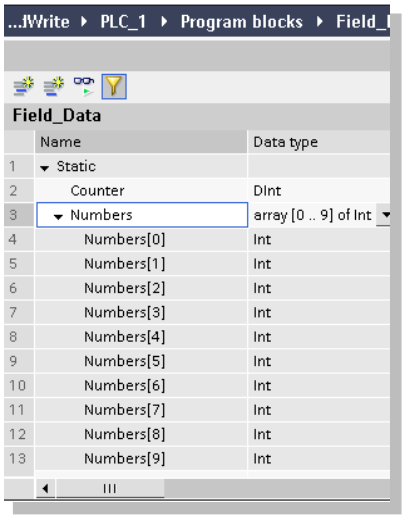
Вы можете найти дополнительную информацию по следующему вопросу:

Почему присвоение и использование символьных имен в STEP 7 (TIA Portal) обязательно для S7-1500?
https://support.industry.siemens.com/cs/ww/en/view/67598995

3.6.2 Тип данных ARRAY и косвенный доступ к элементам

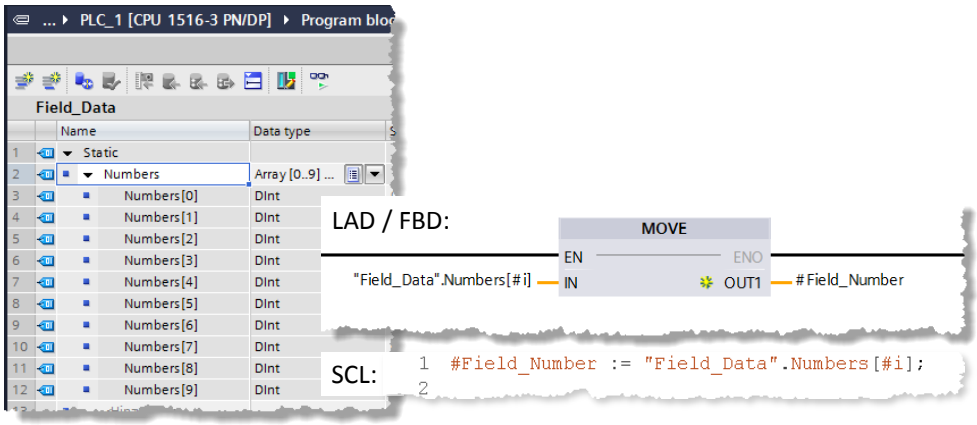
Тип данных ARRAY (Массив) представляет из себя структуру данных, которая состоит из нескольких элементов одного типа. Тип данных ARRAY подходит, например, для хранения рецептов, отслеживания перемещения материалов, циклическая последовательность обработки, протоколы, и.т.д.

Рисунок 3-27: Массив с 10 элементами целочисленного (INT) типа данных



вы можете косвенно получить доступ к отдельным элементам в массиве с помощью переменной (`array ["index"]`).

Рисунок 3-28: Косвенный доступ к элементу



Преимущества

- Простой доступ, так как тип даных всех элементов ARRAY одинаковый.
- Не требуется создания сложного указателя
- Возможно быстрое создание и расширение
- Используется на всех языках программирования

Свойства

- Структурированный тип данных
- Структура данных состоит из определенного числа элементов одного типа данных
- Возможно создание многомерных массивов
- Возможен косвенный доступ с помощью переменной, динамически изменяющейся во время работы программы

Рекомендация

- Используйте массив ARRAY для получения индексированного доступа вместо указателя (например, указатель ANY). Программа становится более читаемой, так как восприятие массива гораздо понятнее, благодаря символьному имени, в отличие от указателя.
- Для хранения индекса ипользуйте тип данных DINT в качестве временной переменной для получения наиболее быстрого доступа.
- Используйте инструкцию “MOVE_BLK”, чтобы скопировать часть одного массива в другой.
- Используйте инструкцию “GET_ERR_ID”, чтобы получить идентификатор ошибки при доступе к массиву.

Примечание

Вы можете найти дополнительную информацию по следующим вопросам:

Каким образом можно выполнить доступ к массиву с переменным индексом в S7-1500? https://support.industry.siemens.com/cs/ww/en/view/67598676

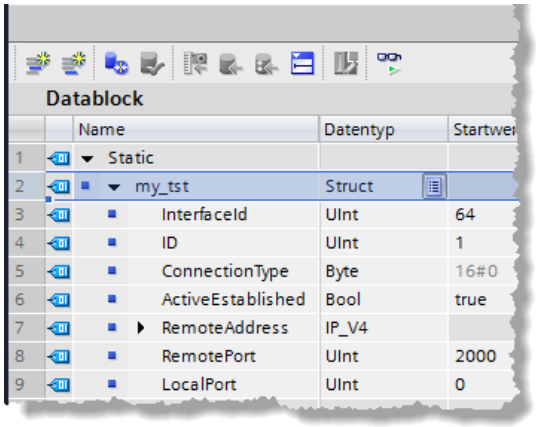
Каким способом можно безопасно использовать косвенную адресацию в STEP 7 (TIA Portal)? https://support.industry.siemens.com/cs/ww/en/view/97552147

Как, в STEP 7 (TIA Portal), для S7-1500 можно выполнить передачу данных между двумя переменными с типами данных “Array of Bool” и “Word”? https://support.industry.siemens.com/cs/ww/en/view/108999241

3.6.3 Тип данных STRUCT и PLC data type

Тип данных STRUCT представляет из себя структуру данных, которая состоит из элементов различных типов. Описание структуры производится в соответствующем блоке.

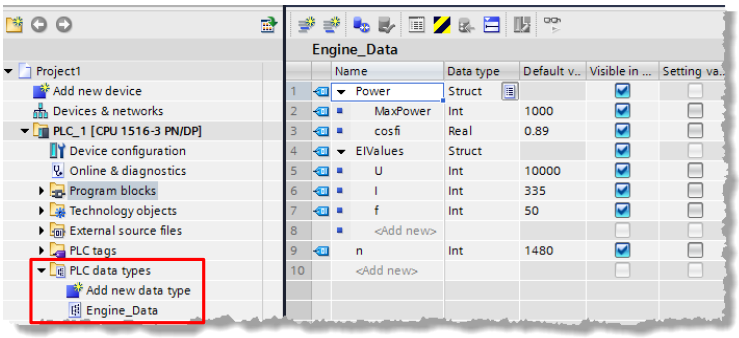
Рисунок 3-29: Структура с элементами различного типа данных



В сравнении со структурами, шаблон PLC data type определяется вне контроллера в TIA Portal и может быть централизованно изменен. Все места, где используется такой тип будут автоматически обновлены.

PLC data type описываются в разделе “PLC data types” в навигаторе проекта до начала использования.

Рисунок 3-30: PLC data types



Преимущества

- Изменение в PLC data type автоматически вступает в силу во всех местах использования в программе.
- Простой обмен данными между блоками через их интерфейсы

Свойства

- PLC data types всегда оканчивается на границе слова (см. Рисунок ниже).
- Учитывайте свойства системы, когда ...
  - используете различные области ввода/вывода (см. главу 3.6.4 Доступ к областям ввода/вывода с помощью PLC data types).
  - используете фреймы при коммуникации с PLC data types
  - используете записи параметров с PLC data types для периферии.
  - используется абсолютная адресация в неоптимизированных блоках.

Рисунок 3-31: PLC data types всегда оканчивается на границе слова

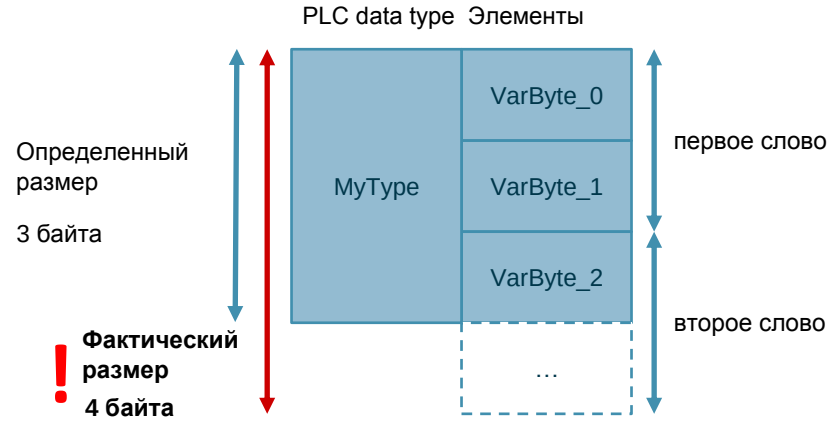
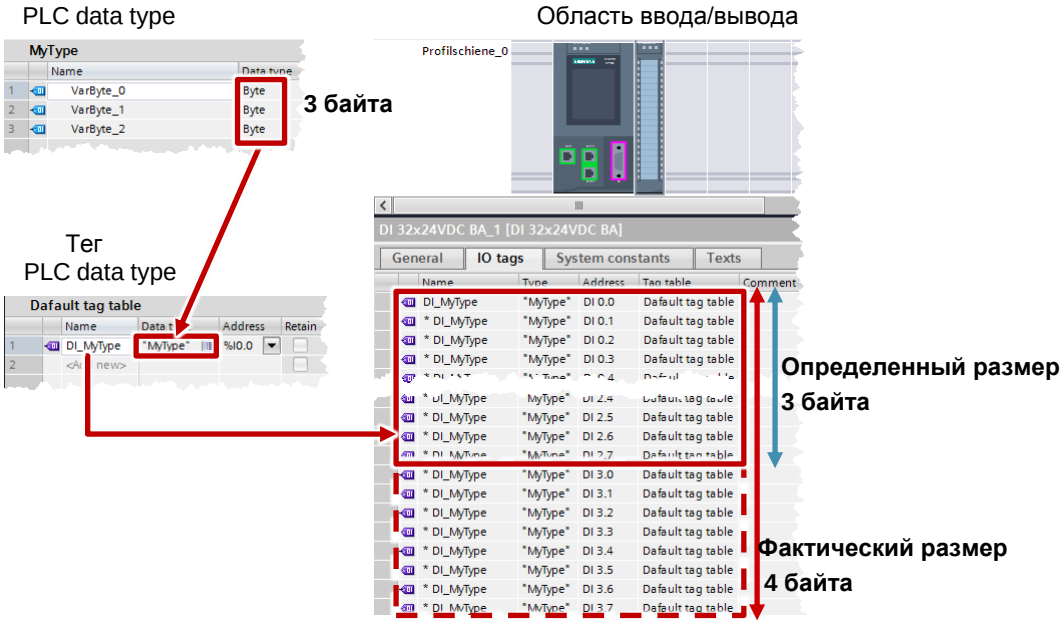


Рисунок 3-32: PLC data type при работе с областями ввода/вывода



Рекомендация

- Используйте шаблоны PLC data types для обьединения нескольких значений, например, фреймы или данные двигателя (уставка скорости, направление вращения, температура, и.т.д.)
- Всегда используйте шаблоны PLC data types вместо структур для многократного использования в программе.
- Используйте шаблоны PLC data types для структурирования в блоках данных.
- Используйте шаблоны PLC data types для назначения структуры блоку данных. PLC data type может быть использован для любого количества DB. Вы легко можете создать необходимое количество DB с одной структурой и в дальнейшем настраивать их централизованно по единому шаблону PLC data type.

Примечание

Вы можете получить подробную информацию по следующим вопросам:

Каким образом инициализируются структуры в оптимизированных областях памяти для S7-1500 STEP 7 (TIA Portal)?
https://support.industry.siemens.com/cs/ww/en/view/78678760

Как создать шаблон PLC data type для контроллера S7-1500 ?
https://support.industry.siemens.com/cs/ww/en/view/67599090

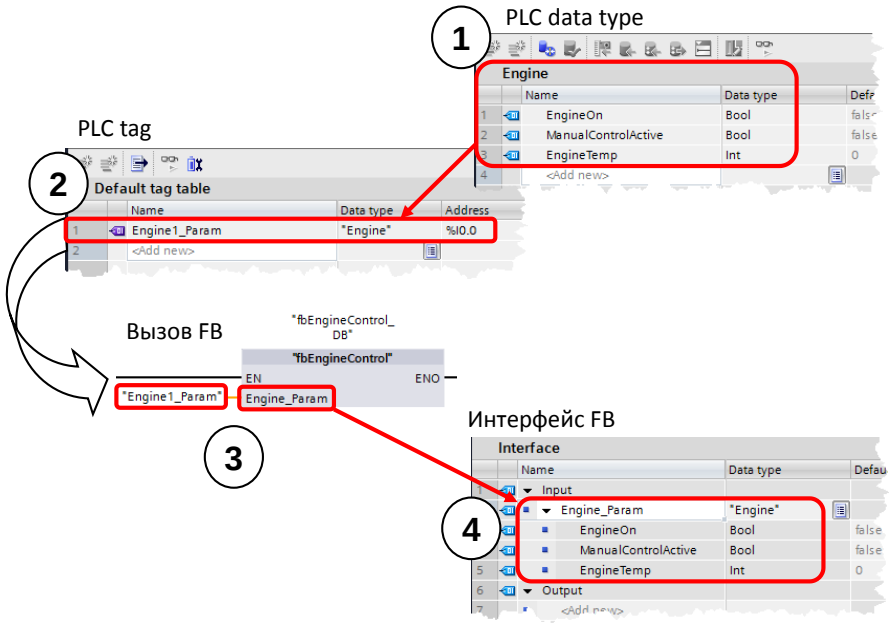
Каким образом применять пользовательские типы данных (UDT) в STEP 7 (TIA Portal)? https://support.industry.siemens.com/cs/ww/en/view/67582844

Почему, при вызове блока для S7-1500 должны передаваться целые структуры, вместо одиночных элементов ?
https://support.industry.siemens.com/cs/ww/de/view/67585079

3.6.4 Доступ к областям ввода/вывода с помощью PLC data types

В контроллерах S7-1500, Вы можете создать шаблоны PLC data types и использовать их для структурированного и символьного доступа ко входам и выходам.

Рисунок 3-33: Доступ к областям ввода/вывода с помощью PLC data types



1. Создание PLC data type с необходимой структурой данных
2. Создание тега PLC по шаблону PLC data type и начальный адрес области входов/выходов (%Ix.0 или %Qx.0, например, %I0.0, %Q12.0, …)
3. Передача тега PLC в качестве фактического параметра в функциональный блок
4. Входной параметр функционального блока с типом, созданного шаблона PLC data type

Преимущества

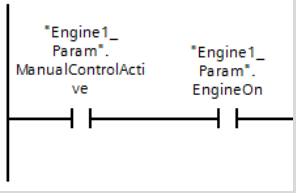
- Высокая эффективность при программировании
- Простое многократное использование, благодаря шаблону PLC data types

Рекомендация

- Используйте PLC data types для получения доступа к входам/выходам, например, для символьного получения и отправки телеграмм.

Примечание

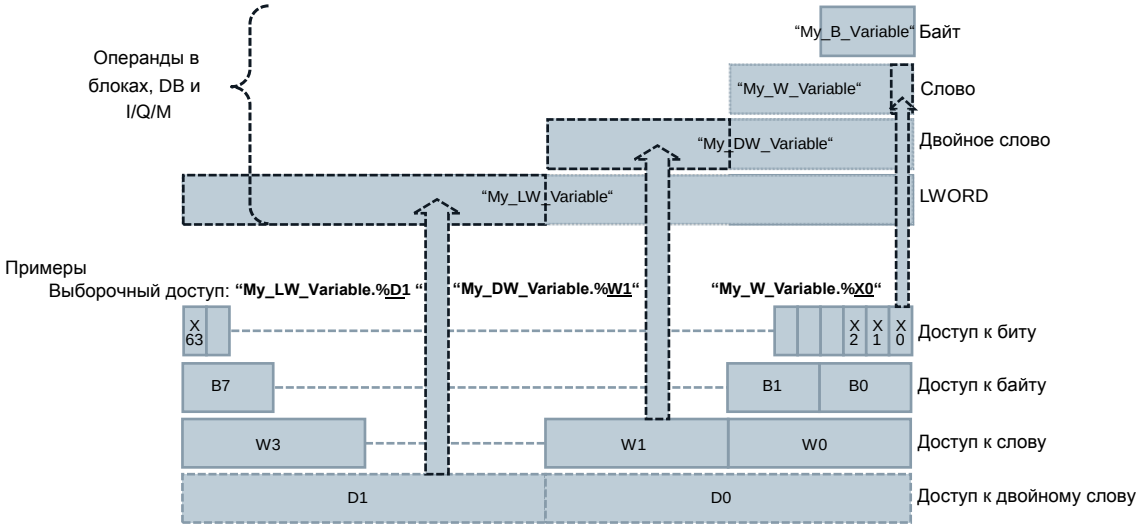
Доступ к отдельным элементам шаблона PLC data type может быть выполнен напрямую в пользовательской програме:



3.6.5 Выборочный доступ

Для контроллеров S7-1200/1500, Вы можете получить доступ к тегам с типами данных Byte, Word, DWord или LWord. Разделение области памяти (например, байт или слово) на области памяти меньшего размера (например, Bool) также называется выборкой. На Рисунке, приведенном ниже, показан доступ к переменным с типом данных бит, байт и слово.

Рисунок 3-34: Выборочный доступ



Преимущества

- Высокая эффективность при программировании
- Не требуется дополнительных структурирований при обьявлении переменной
- Простой доступ (например, биты управления)

Рекомендация

- Используйте выборочный доступ вместо AT отображения при доступе к конкретным областям данных в составе переменных.

Примечание

Вы можете получить подробную информацию по следующим вопросам:

Как в STEP 7 (TIA Portal), Вы можете получить символьный доступ к неструктурированным типам данных: побитно, побайтно, пословно или символьно? https://support.industry.siemens.com/cs/ww/en/view/57374718

3.7 Бибилотеки

В TIA Portal, Вы можете создавать независимые библиотеки из различных компонентов проекта, которые в дальнейшем могут быть использованы повторно.

Преимущества

- Простое хранение сконфигурированных компонентов в TIA Portal:
  - Устройства (контроллер, HMI, привод, и.т.д.)
  - Программы, блоки, переменные, таблицы наблюдения
  - Образ HMI, HMI теги, скрипты, и.т.д.
- Межпроектный обмен через библиотеки
- Функция централизованного обновления элементов библиотеки
- Управление версиями компонентов библиотеки
- Уменьшение количества возможных источников ошибок

Рекомендации

- Создавайте мастер копии для упрощения повторного использования блоков, аппаратных конфигураций, изображений HMI, и.т.д.
- Создавайте типы для поддерживаемой системы возможности повторного использования компонентов библиотеки:
  - Управление версиями блоков
  - Функция централизованного обновления всех элементов программы
- Используйте глобальную библиотеку для обмена с другими пользователями или в качестве центральной базы для одновременной работы нескольких пользователей.
- Сконфигурируйте место сохранения Вашей глобальной библиотеки таким образом, чтобы она автоматически открывалась при запуске TIA Portal. Более подробная информация доступна по ссылке: https://support.industry.siemens.com/cs/ww/en/view/100451450

Примечание

Вы можете получить подробную информацию по следующим вопросам:

Какие элементы STEP 7 (TIA Portal) и WinCC (TIA Portal), Вы можете сохранять в библиотеке в качестве типа или мастер копии?
https://support.industry.siemens.com/cs/ww/en/view/109476862

Каким образом открыть глобальную библиотеку с правом доступа на запись в STEP 7 (TIA Portal)?
https://support.industry.siemens.com/cs/ww/en/view/37364723

3.7.1 Типы библиотек и элементы библиотек

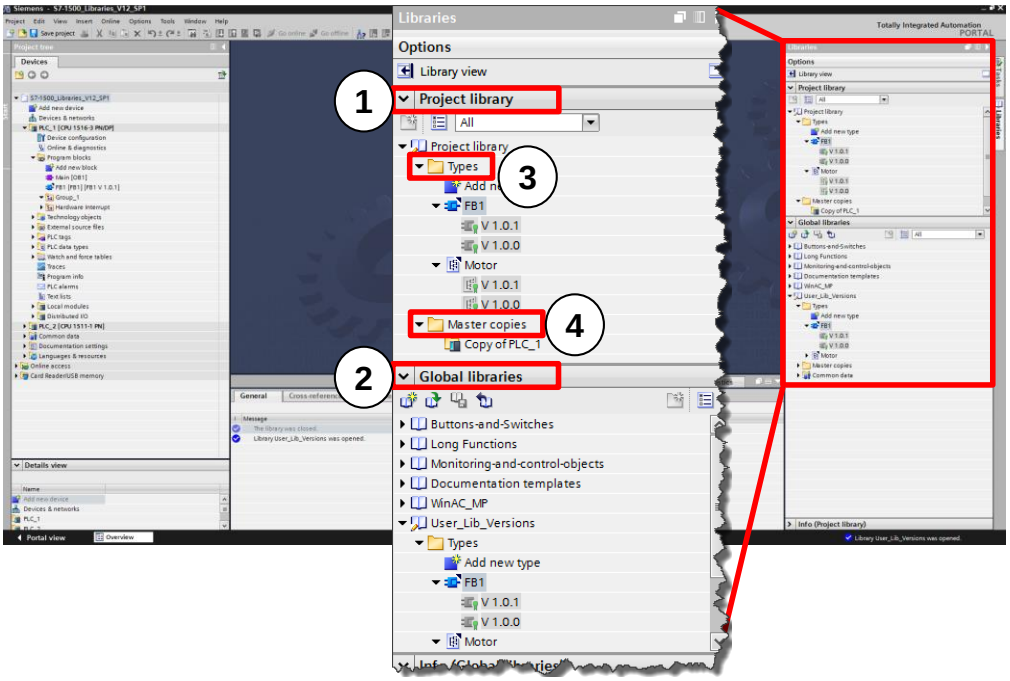
Существует два типа библиотек:

- "Project library" (Проектная библиотека)
- "Global library" (Глобальная библиотека).

Каждая состоит из:

- "Types" (Типов)
- "Master Copies" (Мастер копий)

Рисунок 3-35: Библиотеки в TIA Portal



(1) "Project library" (Проектная библиотека)

- Встроена и управляется в проекте
- Позволяет повторно использовать компоненты в проекте

(2) "Global library" (Глобальная библиотека)

- Независимая библиотека
- Возможно использование в разных проектах

Библиотека состоит из двух типов библиотечных элементов:

(3) "Master copies" (Мастер копии)

- Копии конфигурационных элементов в библиотеке (например, блоки, аппаратная конфигурация, таблицы переменных PLC, и.т.д.)
- Копии не связаны с элементами в проекте.
- Мастер копии могут также состоять из нескольких конфигурационных элементов.

(4) "Types" (Типы)

- Типы связаны с местами использования компонентов Вашего проекта. При изменении типов, все места их использования в проекте могут быть автоматически обновлены.

- Поддерживаемые типы: блоки (FC, FB), PLC data types, образы HMI, лицевые панели HMI, HMI UDT, скрипты).
- Подчиненные элементы автоматически типизируются.
- Каждый тип получает свою версию: Изменения могут быть сделаны только при создании новой версии.
- В контроллере может быть использована только одна версия типа.

3.7.2 Типовая концепция

Данная концепция позволяет создавать стандартизованные функции автоматизации, которые Вы можете использовать в нескольких установках. Типовая концепция позволяет Вам создавать новые версии и обновления для функций.

Вы можете использовать типы из библиотеки в пользовательской программе. При этом, Вы получаете следующие преимущества:

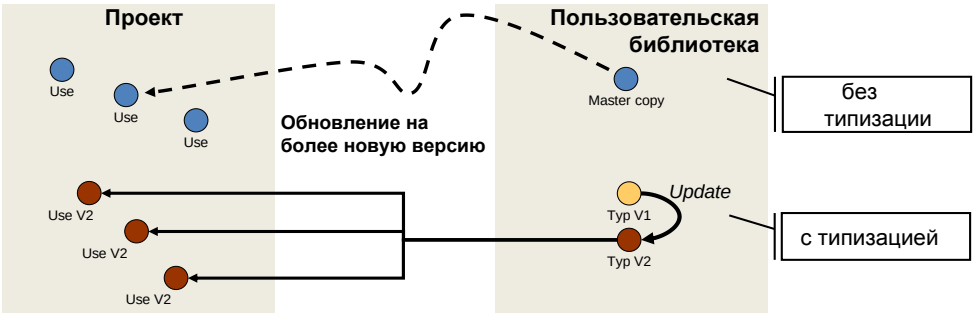
Преимущества

- Централизованное обновление всех компонентов в проекте
- Невозможны нежелательные изменения в местах использования типов.
- Система гарантирует консистентность типов, затрудняя возможность удаления нежелательных операций.
- При удалении типа, он будет удален из всех мест, где был использован до этого в программе.

Свойства

При использовании типа, Вы можете изменять данный элемент централизованно и эти изменения будут применены во всей программе проекта.

Рисунок 3-36: Типизация с помощью пользовательских библиотек



- Типы всегда помечаются в проекте, для лучшей идентификации

3.7.3 Различия между типизированными объектами для CPU и HMI

Имеются системные отличия между типизированными обьектами для контроллеров и HMI:

Таблица 3-9: Отличия типов для контроллера и HMI

Контроллер	HMI
Подчиненные элементы управления типизируются.	Подчиненные элементы HMI **не** типизируются.
Подчиненные элементы управления используют экземпляры.	Подчиненные элементы HMI **не** используют экземпляры.
Элементы управления редактируются в **тестовом режиме**	HMI изображения и скрипты редактируются в среде отладки. Лицевые панели и HMI - UDT редактируются прямо в библиотеке **без тестового режима**.

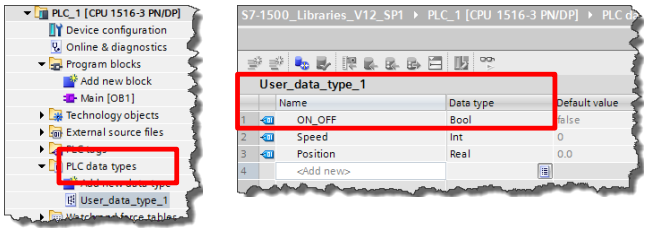
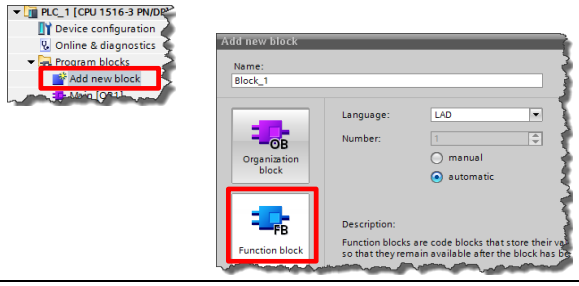
Более подоробную информацию по работе с библиотекам, Вы можете найти в следующих примерах.

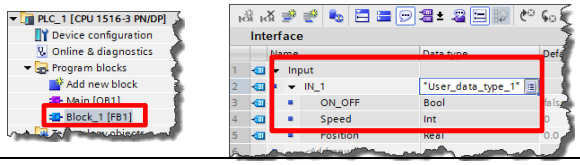
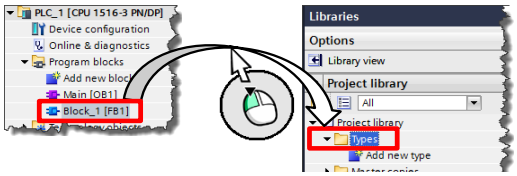
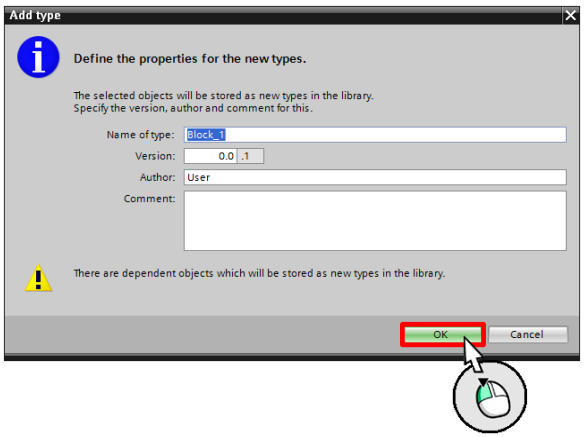
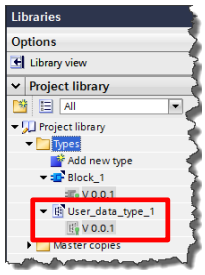
3.7.4 Создание версий блока

Пример: Создание типа

В следующем примере показано использование основных функций при использовании типов библиотек.

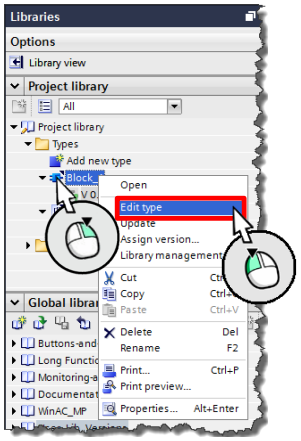
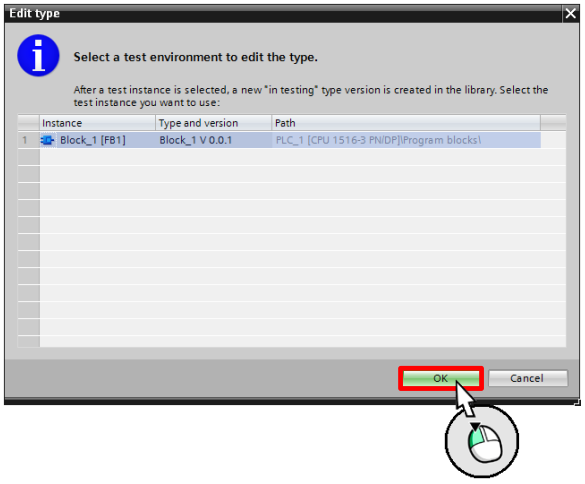
Таблица 3-10: Создание типа

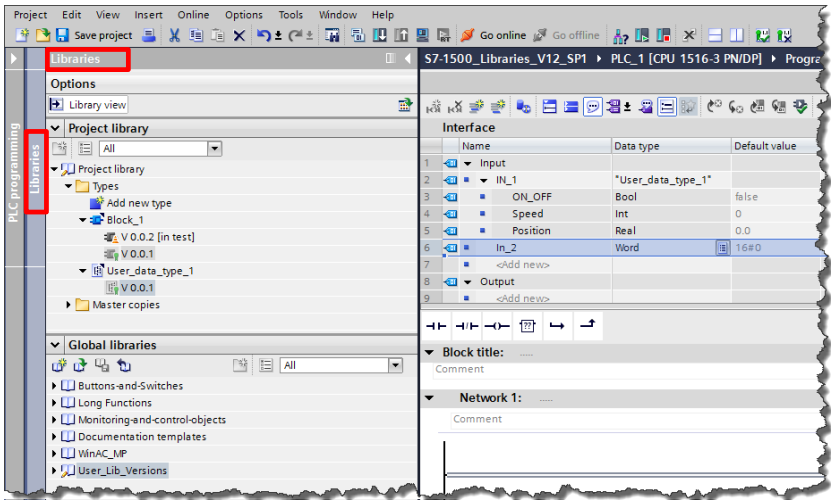
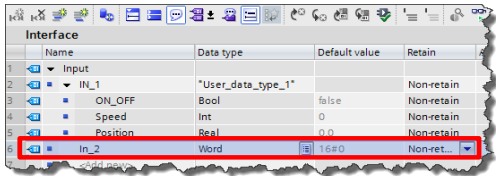
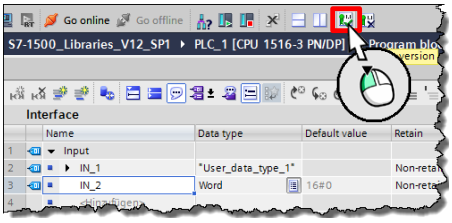
Шаг	Инструкция
1.	Создайте новый шаблон PLC data type с помощью “Add new data type” (Добавить новый тип данных) и создайте несколько переменных. Далее, данный тип будет использован, как прообраз. 
2.	Создайте новый функциональный блок с помощью “Add new Block” (Добавить новый блок). Данный тип будет более высокого уровня. 

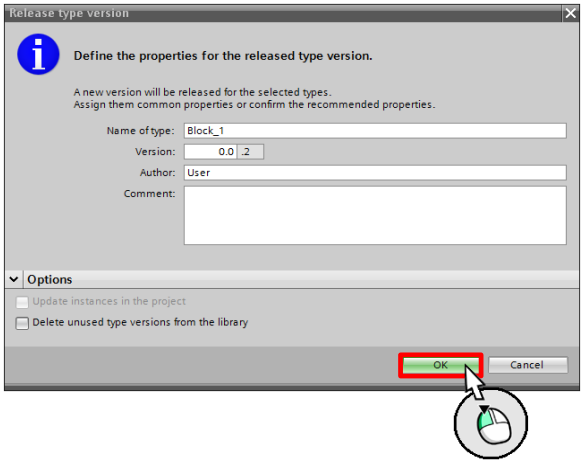
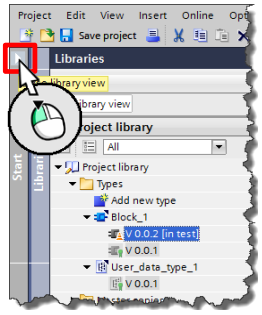
Шаг	Инструкция
3.	Определите входную переменную, с созданным типом данных. PLC data type при этом будет использован в качестве подчиненного функциональному блоку. 
4.	Перенесите функциональный блок, с помощью drag & drop, в раздел “Types” (Типы) проектной библиотеки. 
5.	При необходимости, назначьте: Имя типа, версию, автора и комментарий, после чего нажмите “OK”. 
6.	Использованный шаблон PLC data type также будет автоматически сохранен в библиотеке. 

Пример: Изменение типа

Таблица 3-11: Изменение типа

Шаг	Инструкция
1.	Нажмите правой кнопкой мыши на блоке в “Project library” (Проектная библиотека) и выберите “Edit type” (Редактировать типа) 
2.	Выберите какой контроллер будет использован в качестве тестового оборудования для проверки функционирования и нажмите “OK”.  Если в проекте несколько контроллеров используют выбранный блок, то необходимо выбрать нужный контроллер в качестве тестового.

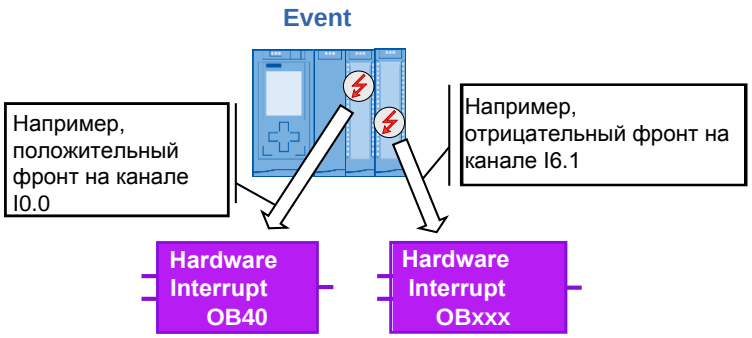
Шаг	Инструкция
3.	При открытии окна библиотеки, будет создана новая версия блока и она будет помечена “in test” (проверяется). 
4.	Добавьте еще одну входную переменную.  На данном этапе, Вы можете проверить изменения в блоке, выполнив загрузку проекта в контроллер. Когда Вы завершите тестирование блока, переходите к следующим шагам.
5.	Нажмите кнопку “Release version” (Выпустить версию блока). 

Шаг	Инструкция
6.	Откроется диалоговое окно. Здесь, Вы можете написать комментарий к текущей версии . Нажмите “OK”.  Если блок используется в нескольких местах программы в различных контроллерах проекта, Вы можете одновременно обновить все блоки: “Update instances in the project” (Обновить экземпляры в проекте). Если старые версии элемента более не требуются, Вы можете удалить их, нажав “Delete unused type versions from library” (Удалить неиспользуемые версии типа из библиотеки).
7.	Закройте отображение библиотеки “Close library view” (Закрыть отображение библиотеки). 

3.8 Повышение производительности при помощи аппаратных прерываний

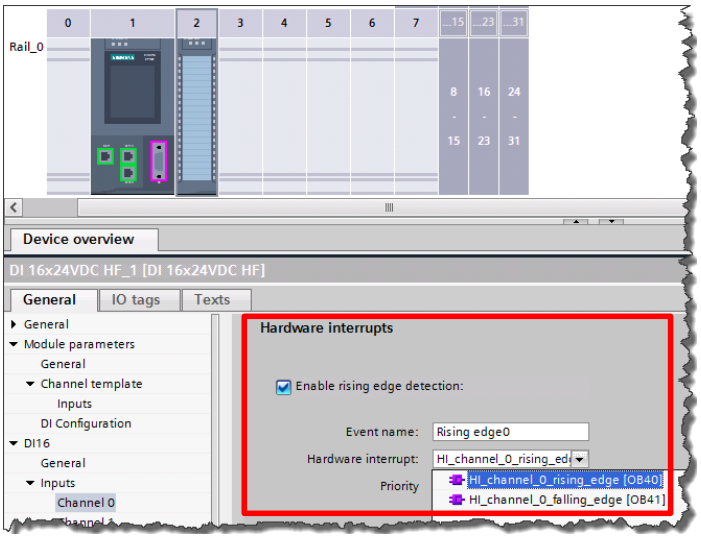
Выполнение пользовательской программы могут прервать некоторые события, например аппаратные прерывания. Когда Вам необходим быстрый отклик процессора на аппаратное прерывание (например, появление положительного фронта сигнала на входном дискретном канале), его необходимо сконфигурировать. Для каждого такого прерывания, может быть запрограммирован свой OB. Данный OB будет вызван операционной системой контроллера при появлении такого события. При этом, рабочий цикл контроллера останавливается и продолжается после обработки аппаратного прерывания.

Рисунок 3-37: При появлении прерывания вызывается OB



На следующем изображении, Вы можете увидеть конфигурацию “аппаратного прерывания” в редакторе аппаратной конфигурации для цифрового входного модуля.

Рисунок 3-38: Конфигурация аппаратного прерывания



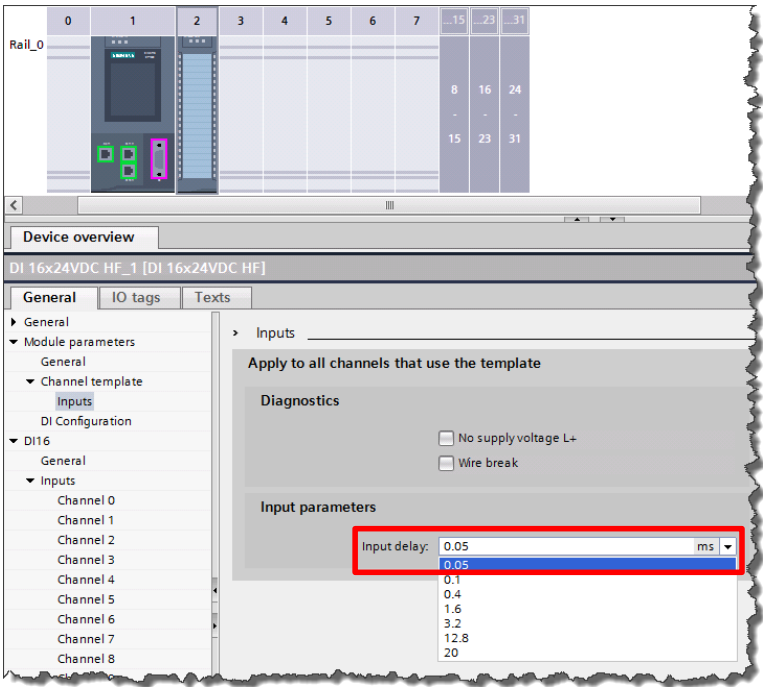
Преимущества

- Быстрая реакция системы на событие (выход за верхнюю, нижнюю границы, появление фронта сигнала и.т.д.)
- Для каждого события может быть вызван свой OB.

Рекомендация

- Используйте процессные прерывания в программе для быстрой реакции на появление аппаратных событий.
- Если отклик системы недостаточно быстрый, несмотря на програмирование аппаратного прерывания, Вы также можете улучшить реакцию системы. Установите наименьшую задержку для модуля в “Input delay”. Отклик на событие будет выполнен только спустя заданное время задержки. Данный параметр используется для фильтрации входного сигнала, например, для устранения таких ошибок, как, дребезг контактов.

Рисунок 3-39: Установки задержки для входа



3.9 Дополнительные рекомендации по увеличению производительности

Здесь Вы можете найти несколько полезных рекомендаций, которые увеличат производительность Вашего контроллера.

Рекомендации

Для повышения производительности контроллеров S7-1200/1500, воспользуйтесь следующими советами:

- LAD/FBD: Отключите “generate ENO” для блоков. При активном режиме исполнения, будет деактивирована проверка.
- STL: Не используйте регистры, так как адресные регистры и регистры данных эмулируются в S7-1500, в целях совместимости.

Примечание

Вы можете найти дополнительную информацию по следующим вопросам:

Каким образом деактивировать управление выходом ENO у инструкции?https://support.industry.siemens.com/cs/ww/en/view/67797146
Каким образом можно увеличить производительность в STEP 7 (TIA Portal) для S7-1200/S7-1500 CPU?
https://support.industry.siemens.com/cs/ww/en/view/37571372

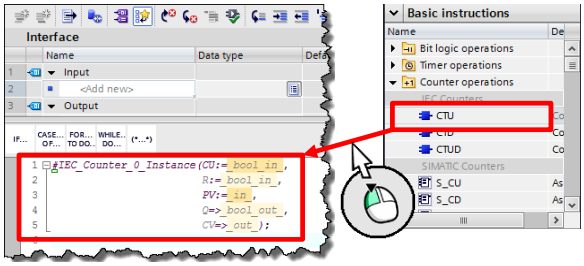
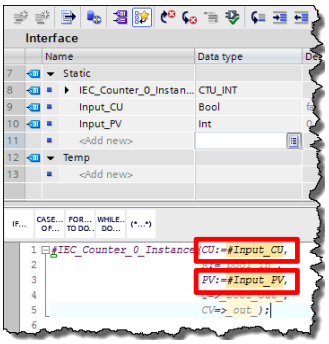
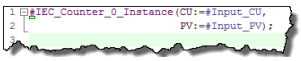
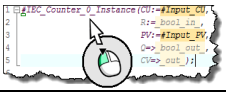
3.10 Язык программирования SCL: Советы и рекомендации

3.10.1 Использование шаблонов вызова

Многие инструкции языка программирования могут быть представлены в виде уже написанного шаблона вызова инструкции с формальными параметрами.

Пример

Таблица 3-12: Простое расширение шаблона

Шаг	Инструкция
1.	Перенесите инструкцию из библиотеки в программу SCL. В редакторе будет показан шаблон вызова данной инструкции. 
2.	Теперь необходимо заполнить необходимые параметры и после этого нажать кнопку “Return” (Возврат). 
3.	Редактор автоматически удалит лишнее из шаблона вызова 
4.	Если Вы хотите вернуть полный вызов инструкции, проделайте следующую процедуру. Поместите курсор мыши на вызов и нажмите “**CTRL+SHIFT+SPACE**”. Теперь, Вы находитесь в режиме Call Template. Редактор повторно расширит вызов. С помощью кнопки “TAB”, Вы можете выполнять переход между параметрами. 
5.	Примечание: в режиме “Call Template” ввод выполняется курсивом.

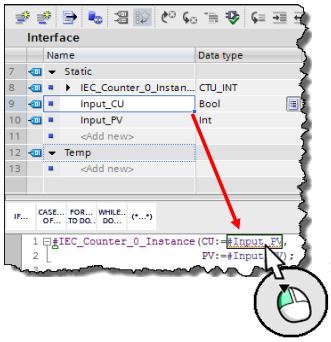
3.10.2 Какие параметры инструкции обязательны?

Если Вы расширили шаблон вызова, то по цвету параметров, Вы поймете какой формальный параметр обязателен к заданию фактического, а какой нет. Обязательные параметры выделяются темным цветом.

3.10.3 Перенос имен переменных

В редакторе SCL, Вы также можете воспользоваться функцией перетаскивания drag & drop, также могут быть перенесены имена переменных. Для замены одной переменной на другую, проделайте следующую процедуру.

Таблица 3-13: Перенос переменных в SCL

Шаг	Инструкция
1.	Перенесите одну переменную на другую с помощью drag & drop. Удерживайте переменную более чем на 1 секунду, а затем отпустите.  > удерживайте примерно 1 секунду Переменная будет заменена.

3.10.4 Применение циклов FOR, REPEAT и WHILE

При работе с циклами, имеется три различных инструкции. На следующих примерах показаны основные отличия.

Свойства: цикл FOR

Цикл FOR выполняется **определенное количество раз**. Сначала, счетчик устанавливается на начальное значение. Затем, в каждом проходе цикла, счетчик увеличивается с заданным шагом, до тех пор, пока не достигнет конечного значения.

В целях высокой производительности, начальное и конечные значения высчитываются один раз в начале цикла. .После этого, значение счетчика не влияет на код в теле цикла.

Синтаксис

```
FOR counter := start_count TO end_count DO
    // Тело цикла ;
END_FOR;
```

При помощи команды EXIT, цикл может быть прерван в любой момент времени.

Свойства: цикл WHILE

Цикл с предусловием WHILE, прекращает свою работу, как только условие цикла перестало выполняться. **Условие завершения цикла проверяется до** тела цикла. Таким образом, цикл не будет выполняться, если условие не удовлетворено. В теле цикла, каждая переменная может быть подготовлена для следующего цикла.

Синтаксис

```
WHILE condition DO
     // Тело цикла ;
END_WHILE;
```

Свойства: цикл REPEAT

Цикл с постусловием REPEAT, прекращает свою работу, как только условие цикла перестало выполняться. **Условие завершения цикла проверяется после** тела цикла. Таким образом, данный цикл **будет выполнен, как минимум один раз**. В теле цикла, каждая переменная может быть подготовлена для следующего цикла.

Синтаксис

```
REPEAT
    // Тело цикла ;
UNTIL condition
END_REPEAT;
```

Рекомендация

- Если значение тега счетчика заранее известно, используйте цикл FOR.
- Если тег счетчика или условие продолжения цикла может корректироваться в процессе выполнения циклов, используйте циклы WHILE или REPEAT.

3.10.5 Использование инструкции CASE

С помощью инструкции CASE на языке SCL, программа может выполнить переход к одному из участков программы при выполнении условия. После этого, выполнение инструкции CASE заканчивается. Данный механизм, например, позволит Вам часто проверять необходимые диапазоны значений.

Пример

```
CASE #myVar OF
        5:
                FC5(#myParam);
        10,12:
                FC10(#myParam);
        15:
                FC15(#myParam);
        0..20:
                FCGlobal(#myParam);
// FCGlobal никогда не вызовается для значений 5, 10, 12 или 15!
        ELSE
END_CASE;
```

Примечание

CASE инструкции также работают с типами данных CHAR, STRING и с их элементами (см. пример в главе 2.8.5 Тип данных VARIANT).

3.10.6 Поведение счетчика для циклов FOR

Циклы FOR на языке SCL работают исключительно при помощи счетчиков, т.о. количество итераций фиксировано. В цикле FOR, значение счетчика не может быть изменено. С помощью инструкции EXIT, цикл может быть прерван в любой момент времени.

Преимущества

- Оптимизиация программы компилятором выполняется эффективнее, когда значение счетчика заранее известно.

Пример

```
FOR #var := #lower TO #upper DO
  #var := #var + 1; // некорректно, Компилятор -> Предупреждение
END_FOR;
```

3.10.7 Цикл FOR с обратным направлением

На языке SCL, Вы также можете увеличить счетчик в обратном направлении с другим шагом. Для этого, используйте ключевое слово “BY” в условии цикла.

Пример

```
FOR #var := #upper TO #lower BY -2 DO

END_FOR;
```

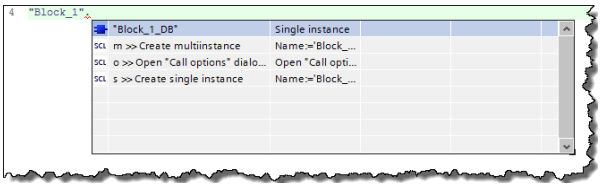
Если, Вы определили “BY” в качестве “-2”, как в показанном примере, то счетчик будет уменьшаться на 2 в каждом проходе цикла. Если, Вы не использовали ключевое слово “BY”, то по умолчанию “BY” будет задано 1.

3.10.8 Простое создание экземпляров для вызовов

Если, Вы предпочитаете работать с клавиатурой, то имеется простой способ создания экземпляров для блоков на SCL.

Пример

Таблица 3-14: Простое создание экземпляров

Шаг	Инструкция
1.	Задайте имя блока: после "." (точка). Вы увидите следующее. 
2.	Сверху показаны уже имеющиеся экземпляры. В дополнение к ним, Вы можете создать новый экземпляр или мультиэкземпляр. Используйте "s" или "m", чтобы перейти к соответствующим записям в окне компиляции.

3.10.9 Обработка переменных с типом данных Time (время)

Вы можете работать с переменными типа данных Time, на языке SCL, также как и с обыкновенными значениями т.е. Вам не потребуются дополнительные функции, такие как, например, T_COMBINE, но Вы можете использовать простые арифметические операции. Данный механизм называется “перегрузка операторов”. Компилятор SCL автоматически использует нужные функции. Вы можете использовать необходимые арифметические инструкции для типа данных Time (время), при этом эффективность программирования увеличивается.

Пример

```
timeDifference := timeStamp1 - timeStamp2;
```

В следующей таблице показано соответствие перегруженных операторов и функций, которые выполняются по факту:

Таблица 3-15: Перегруженные операнды в SCL

Перегруженный операнд	Операция
ltime + time	T_ADD LTime
ltime + time	T_SUB LTime
ltime + lint	T_ADD LTime
ltime + lint	T_SUB LTime
time + time	T_ADD Time
time + time	T_SUB Time
time + dint	T_ADD Time
time + dint	T_SUB Time
ldt + ltime	T_ADD LDT / LTime
ldt + ltime	T_ADD LDT / LTime
ldt + time	T_ADD LDT / Time
ldt + time	T_SUB LDT / Time
dtl + ltime	T_ADD DTL / LTime
dtl + ltime	T_SUB DTL / LTime
dtl + time	T_ADD DTL / Time
dtl + time	T_SUB DTL / Time
ltod + ltime	T_ADD LTOD / LTime
ltod + ltime	T_SUB LTOD / LTime
ltod + lint	T_ADD LTOD / LTime
ltod + lint	T_SUB LTOD / LTime
ltod + time	T_ADD LTOD / Time
ltod + time	T_SUB LTOD / Time
tod + time	T_ADD TOD / Time
tod + time	T_SUB TOD / Time
tod + dint	T_ADD TOD / Time
tod + dint	T_SUB TOD / Time
dt + time	T_ADD DT / Time
dt + time	T_SUB DT / Time
ldt – ldt	T_DIFF LDT
dtl – dtl	T_DIFF DTL
dt – dt	T_DIFF DT
date – date	T_DIFF DATE
ltod – ltod	T_DIFF LTOD
date + ltod	T_COMBINE DATE / LTOD
date + tod	T_COMBINE DATE / TOD

4 Аппаратно-независимое программирование

Чтобы убедиться, что блок может быть использован в любом контроллере, необходимо отказаться от использования аппаратно-зависимых функций и свойств.

4.1 Типы данных S7-300/400 и S7-1200/1500

Ниже показан список элементарных типов данных и групп данных.

Рекомендация

- Используйте только те типы данных, которые поддерживаются всеми контроллерами.

Таблица 4-1: Элементарные типы данных, в соответствии со стандартом EN 61131-3

	Описание	S7 - 300/400	S7-1200	S7-1500
Битовые типы данных	• BOOL • BYTE • WORD • DWORD	✓	✓	✓
	• LWORD	×	×	✓
Символьный тип	• CHAR (8 бит)	✓	✓	✓
Числовые типы данных	• INT (16 бит) • DINT (32 бита) • REAL (32 бита)	✓	✓	✓
	• SINT (8 бит) • USINT (8 бит) • UINT (16 бит) • UDINT (32 бит)a • LREAL (64 бита)	×	✓	✓
	• LINT (64 бита) • ULINT (64 бита)	×	×	✓
Типы данных, обозначающие время	• TIME • DATE • TIME_OF_DAY	✓	✓	✓
	• S5TIME	✓	×	✓
	• LTIME • L_TIME_OF_DAY	×	×	✓

Таблица 4-2: Группы данных, состоящие из других типов данных

	Описание	S7 - 300/400	S7-1200	S7-1500
Типы данных, обозначающие время	• DT (DATE_AND_TIME)	✓	×	✓
	• DTL	×	✓	✓
	• LDT (L_DATE_AND_TIME)	×	×	✓
Строка	• STRING	✓	✓	✓
Массив	• ARRAY	✓	✓	✓
Структура	• STRUCT	✓	✓	✓

Таблица 4-3: Типы формальных параметров, которые используются блоками

	Описание	S7 - 300/400	S7-1200	S7-1500
Указатель	• POINTER • ANY	✓	×	✓[1)]
	• VARIANT	×	✓	✓
Блоки	• TIMER • COUNTER	✓	✓[2)]	✓
	• BLOCK_FB • BLOCK_FC	✓	×	✓
	• BLOCK_DB • BLOCK_SDB	✓	×	×
	• VOID	✓	✓	✓
Шаблон пользователя	• PLC Data Type	✓	✓	✓

1) При оптимизированном доступе, возможна только символьная
2) адресация Для S7-1200/1500 тип данных TIMER и COUNTER представлен как IEC_TIMER и IEC_Counter.

4.2 Переход от меркеров к глобальным блокам данных

Преимущества

- Оптимизированные глобальные DB гораздо эффективнее, чем меркерная память, которая не является оптимизированной в целях совместимости.

Рекомендация

- Использование меркерной памяти (также системных битов и синхробайта) может быть причиной появления ошибок в программе, так как размер данной области памяти у каждого контроллера разный. При написании программ, не используйте меркерную память, вместо этого работайте с глобальным блоками данных. Таким образом программы будут гораздо универсальнее.

4.3 Программирование "синхробайта"

Рекомендация

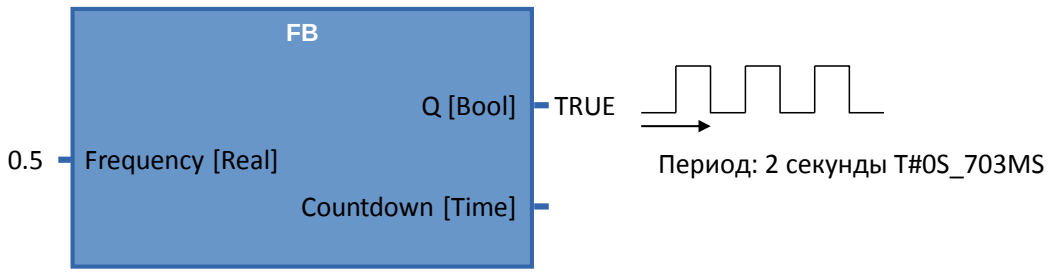
Для работы с синхробайтом, необходимо выполнить его активацию в аппаратной конфигурации контроллера, что налагает требования её соответствия в других контроллерах.

Используйте программный блок в качестве тактового генератора. Ниже, Вы можете найти пример, в котором запрограммирован тактовый генератор на языке программирования SCL.

Пример

У созданного блока имеется следующий функционал. Задается необходимая частота. Выход “Q” имеет тип данных Bool, который будет изменяться с заданной частотой. На выход “Countdown” будет выводиться оставшееся время текущего состояния “Q”.

Если заданная частота меньше или равна 0.0, тогда выход Q = FALSE и Countdown = 0.0.



Примечание

Данный пример, Вы можете скачать по следующей ссылке:

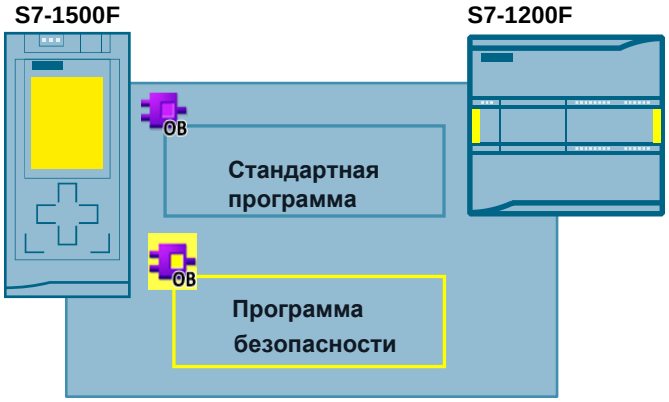
https://support.industry.siemens.com/cs/ww/en/view/87507915

5 STEP 7 Safety в TIA Portal

5.1 Введение

Контроллеры систем безопасности и противоаварийной защиты (Fail-safe) S7-1200F и S7-1500F CPU поддерживаются в TIA Portal V13 SP1. В таких контроллерах, помимо программирования программы безопасности, доступны также стандартные функции. Для создания программ безопасности, используется пакет SIMATIC STEP 7 Safety (TIA Portal).

Рисунок 5-1: Стандартная программа и программа безопасности



Преимущества

- Универсальный инструмент, как для создания стандартных, так и программ безопасности: TIA Portal
- Программирование на LAD и FBD
- Универсальные инструменты диагностики и online функции

Примечание

Программы безопасности не гарантируют отсутствие ошибок. Программист отвечает за правильную логику программирования.

Система безопасности и противоаварийной защиты - означает, что в контроллере обеспечивается правильная обработка пользовательской программы безопасности.

Примечание

Дополнительную информацию по теме безопасности, требований безопасности или принципы программ безопасности, доступны на:

TIA Portal - Обзор наиболее важных документов и ссылки
https://support.industry.siemens.com/cs/ww/en/view/90939626

Applications & Tools – Safety Integrated
https://support.industry.siemens.com/cs/ww/en/ps/14675/ae

STEP 7 Safety (TIA Portal) - Руководства
https://support.industry.siemens.com/cs/ww/en/ps/14675/man

5.2 Термины

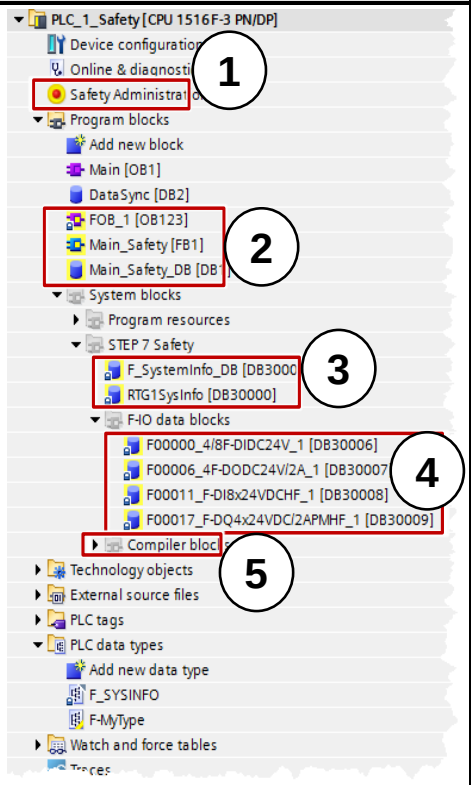
В данном документе используются следующие термины.

Таблица 5-1: Термины безопасности

Термин	Описание
Стандартная пользовательская программа	Стандартная пользовательская программа, в которой не используются функции F программы.
Программа безопасности (F программа, программа безопасности)	Пользовательская программа безопасности, обрабатывается отдельно, независимо на контроллере . Все блоки и инструкции программы безопасности помечены желтым цветом в пользовательском интерфейсе редактора (например, в навигаторе проекта), для более легкого различия стандартных блоков и блоков программы безопасности. Параметры безопасности F-CPU и F-I/O помечены желтым цветом в редакторе аппаратной конфигурации.

5.3 Элементы програмы безопасности

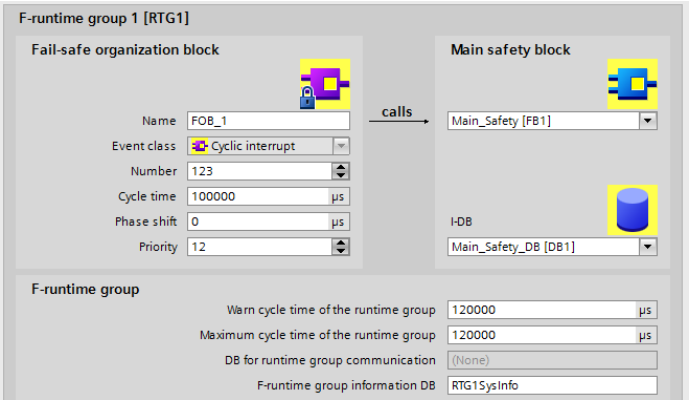
Программа безопасности всегда состоит из пользовательских блоков или сгенерированных системой F-блоков и редактора "Safety Integration" (Управление безопасностью)

Описание	Элементы
1. Редактор “Safety administration” - Состояние программы безопасности - F коллективная подпись - Состояние работы безопасности - Создание/огранизация групп F runtime - Информация об F блоках - Информация об F-шаблонах PLC data types - Обьявление/изменение доступа 2. F блоки, созданные пользователем 3. Сгенерированные системой F блоки Блоки в которых содержаться информация о состоянии F runtime группы. 4. Сгенерированные системой F-I/O блоки данных Блоки с тегами для работы с F модулями. 5. “Блоки компилятора” Сгенерированные системой блоки - Работают в фоновом режиме контроллера и необходимы для обработки программы безопасности. - Данные блоки не могут быть изменены пользователем.	

5.4 F runtime группа

Программа безопасности всегда выполняется в "F-runtime" группе с заданным циклом. "F runtime" группа состоит из “Fail-safe organization block” (Организационный блок безопасности), который называется “Main safety block” (Основный блок безопасности программы). Все пользовательские функции безопасности вызываются из “Main safety block”.

Рисунок 5-2: "F-runtime" группа в редакторе “Safety administration” (Управление безопасностью)



Преимущества

- Runtime группы могут быть созданы и сконфигурированы в “Safety Administrator” (Управление безопасностью).
- F-блоки в runtime группе создаются автоматически.

Свойства

- Может быть создано до 2 F runtime групп

5.5 F подпись (F signature)

У каждого F компонента (станция, периферия, блоки) имеется уникальная F подпись. При помощи F подписи можно легко найти F конфигурацию устройства, F блоки или всю станцию, соответствующую оригинальной станции или программе.

Преимущества

- Простое и быстрое сравнение F блоков и F конфигурации

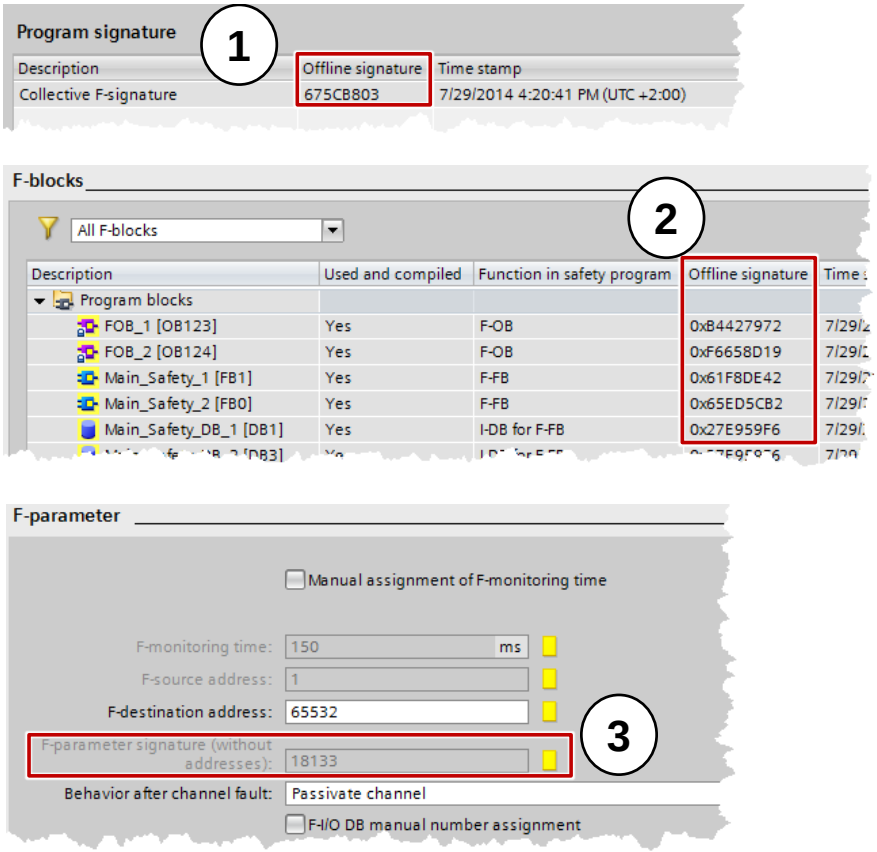
Свойства

- Подпись F параметров (без адресов F-периферии)…
  - изменения только через назначение параметров.
  - остается неизменной при изменении PROFIsafe адреса. Тем не менее, общая F подпись всей станции изменяется.
- Подпись F блока изменяется при изменении логики в F блоке.

- Подпись F блока остается неизменной при изменении
  - номера блока,
  - интерфейса блока,
  - версии блока.

Пример

Рисунок 5-3: Примеры F подписи



1. Коллективная F подпись станции в редакторе “Safety administration” (Управление безопасностью)
2. Подписи F блоков в редакторе “Safety Administration” (Управление безопасностью) (можно также узнать из свойств блока)
3. Подпись F параметров “Device view” (Отображение устройства) в “Devices & Networks”

Примечание

Для контроллеров S7-1500F можно получить коллективную F подпись прямо на дисплее или через Web сервер.

5.6 Назначение PROFIsafe адреса на F-I/O

У каждого F-I/O устройства имеется PROFIsafe адрес для идентификации и коммуникации с F контроллерами. При назначенении PROFIsafe адреса, возможны две различные конфигурации.

Таблица 5-3: Задание F адреса

ET 200M / ET 200S (Тип адреса PROFIsafe 1)	ET 200MP / ET 200SP (Тип адреса PROFIsafe 2)
Назначение PROFIsafe адреса непосредственно на модуле с помощью DIL переключателя В конфигурации устройства TIA Portal и на DIL переключателе на периферии, PROFIsafe адреса должны быть одинаковые.	Назначение PROFIsafe адресов исключительно через TIA Portal Сконфигурированный PROFIsafe адрес загружается в интеллекутальный модуль.

Преимущества

- Замена F модулей возможна без задания PROFIsafe адреса на ET 200MP и ET 200SP. Интеллектуальный кодовый модуль остается в базовом устройстве (BaseUnit), при замене модуля.
- Простая конфигурация, TIA Portal сообщит о неверном задании PROFIsafe адреса.
- PROFIsafe адреса всех F модулей может быть назначен одновременно с ET 200SP.

Примечание

Более подробная информация по назначению PROFIsafe адреса для F-I/O доступна в:

SIMATIC Industrial Software SIMATIC Safety – Конфигурация и программирование
https://support.industry.siemens.com/cs/ww/en/view/54110126

5.7 Оценка F-периферии

Все состояния соответствующего F-I/O сохраняются в F-I/O блоках. В программе безопасности, состояния могут быть проанализированы и обработаны. Существуют следующие различия между S7-1200/1500F и S7-300F/400F.

Таблица 5-4: Переменные в F-I/O DB с S7-300F/400F и S7-1500F

Переменные в F-I/O DB или состояние значения в PAE	F-I/O с S7-300/400F	F-I/O с S7-1200/1500F
ACK_NEC	✓	✓
QBAD	✓	✓
PASS_OUT	✓	✓
QBAD_I_xx *	✓	✗
QBAD_O_xx *	✓	✗
Состояние значения	✗	✓

* QBAD_I_xx и QBAD_O_xx сообщают Вам о действительности значения канала и ссылаются на **инвертированное** состояние значения в S7-1200/1500F (более подробная информация доступна в следующей главе).

5.8 Состояние значения (S7-1200F / S7-1500F)

В дополнение к диагностическим сообщениям, состоянием и отображением ошибок, F модуль предоставляет информацию о действительности каждого входного и выходного сигнала - состояния значения. Состояние значения хранится таким же образом, как входной сигнал - в области отображения:

Состояние значения говорит о корректности значения канала.

- 1: в канале находится корректное значение.
- 0: в канале находится некорректное значение.

Таблица 5-5: Различия между Q_BAD (S7-300F/400F) и состоянием значения (S7-1200/1500F)

Сценарий	QBAD (S7-300F/400F)	Состояние значения (S7-1200/1500F)
Корректное значение на F-I/O (ошибок нет)	FALSE	TRUE
Наличие ошибки на канале	TRUE	FALSE
Информация об уходе ошибки на канале (ACK_REQ)	TRUE	FALSE
Квитирование ошибки канала (ACK_REI)	FALSE	TRUE

Свойства

- Состояние значения записывается в область отображения входов и выходов.
- Доступ к значению канала и состоянию значения F-I/O должнен быть выполнен из одной F runtime группы.

Рекомендация

- Для более легкого чтения программы, добавляйте в конец переменной "`_VS`", например, "`Tag_In_1_VS`" в качестве символьного имени для состояния значения (Value Status).

Пример

Положение битов состояния значения в области отображения на примере F-DI 8x24VDC HF модуля.

Таблица 5-6: Биты состояния значения в области отображения на примере F-DI 8x24VDC HF

Байт в F-CPU	Назначение битов в F-CPU							
	7	6	5	4	3	2	1	0
x + 0	DI_7	DI_6	DI_5	DI_4	DI_3	DI_2	DI_1	DI_0
x +1	Состояние значения для DI_7	Состояние значения для DI_6	Состояние значения для DI_5	Состояние значения для DI_4	Состояние значения для DI_3	Состояние значения для DI_2	Состояние значения для DI_1	Состояние значения для DI_0

x = начальный адрес модуля

Примечание

Более подробная информация по состоянию значения всех модулей ET 200SP доступна в:

Руководства по F-CPU
https://support.industry.siemens.com/cs/ww/en/ps/13719/man

Руководства по периферийным модулям F I/O
https://support.industry.siemens.com/cs/ww/en/ps/14059/man

5.9 Типы данных

В программах безопасности могут быть использованы следующие типы данных для S7-1200/1500F.

Таблица: 5-7: Целочисленные типы данных

Тип	Размер	Диапазон значений
BOOL	1 бит	0 .. 1
INT	16 бит	-32.768 .. 32.767
WORD	16 бит	-32.768 .. 65.535
DINT	32 бита	-2.14 .. 2.14 миллиона
TIME	32 бита	T#-24d20h31m23s648ms до T#+24d20h31m23s647ms

5.10 Шаблоны PLC data type для F-программ

Для программ безопасности также доступны стркутурированные типы данных PLC data types.

Преимущества

- Изменения в PLC data type автоматически вступают в силу во всех местах программы пользователя.

Свойства

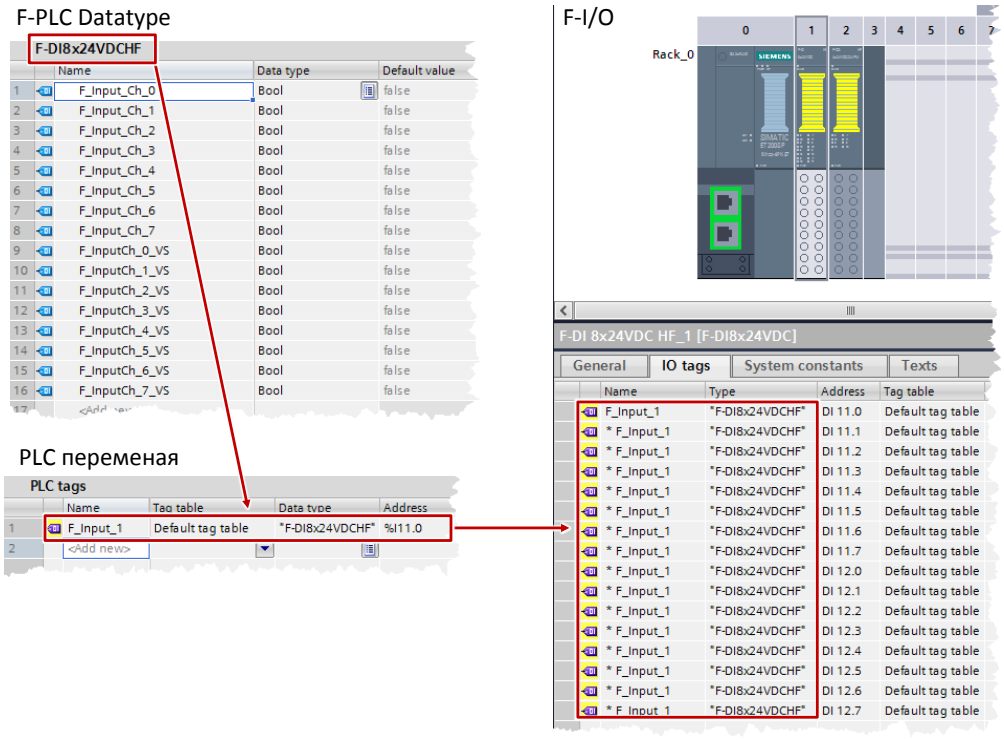
- Шаблоны F-PLC data types обьявляются и используются так же, как PLC data types.
- Шаблоны F-PLC data types могут работать со всеми типами данных, которые могут использоваться в программе безопасности.
- Недоступно использование F-PLC data types внутри других шаблонов F-PLC data types.
- Шаблоны F-PLC data types, стандартные пользовательские программы могут быть использованы в программе безопасности, так же как в стандартной программе.

Рекомендация

- Для доступа к областям ввода/вывода, необходимо использовать шаблоны F-PLC data types (как в главе 3.6.4 Доступ к областям ввода/вывода с помощью PLC data types)
- Должны быть соблюдены следующие правила:
  - Структура переменных шаблона F-PLC data type должна совпадать со структурой канала F-I/O.
  - Шаблон F-PLC data type для F-I/O с 8 каналами, например:
    - 8 переменных типа BOOL (значение канала)
    - 16 переменных типа BOOL (значение канала + состояние значения)
  - Доступ к F-I/O может быть выполнен только к активированным каналам. При конфигурировании 1oo2 (2v2) оценки, верхний канал всегда деактивируется.

Пример

Рисунок 5-4: Доступ к областям ввода/вывода с помощью F-PLC data types



5.11 TRUE / FALSE

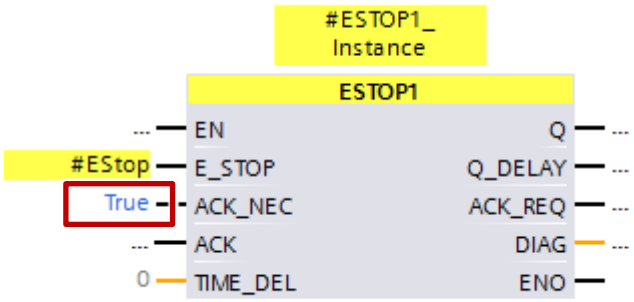
Если Вам необходимы сигналы “TRUE” и “FALSE” в программе безопасности, имеется два различных способа:

- в качестве фактических параметров блока
- в качестве операнда в инструкции

Фактический параметр блока

Для контроллеров S7-1200/1500F, Вы можете использовать битовые (Bool) константы “FALSE” для 0 и “TRUE” для 1, в качестве фактических параметров для назначения формальным параметрам, для дальнейшего их использования блоком в процессе вызова в программе безопасности. В формальный параметр записывается только ключевое слово “FALSE” или “TRUE”.

Рисунок 5-5: Сигналы “TRUE” и “FALSE” в качестве фактических параметров

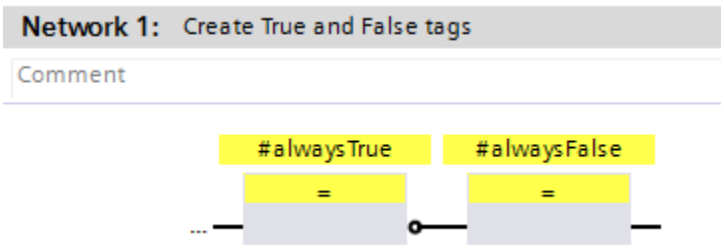


Использование в инструкциях

Если, Вам необходимо назначить сигналы “TRUE” или “FALSE” инструкциям, Вам необходимо сформировать их, как показано на рисунке ниже.

- Переключитесь на язык программирования FBD.
- Создайте две статических или временных переменных с типом bool: “alwaysTrue”, “alwaysFalse”.
- Создатйе сегмент 1, как на рисунке ниже.
- Переменные могут быть использованы в качестве “True” и “False” для всего блока.

Рисунок 5-6: Сигналы “TRUE” и “FALSE”



5.12 Оптимизированная компиляция и режим исполнения программы

В данной главе, Вы узнаете о том, как уменьшить время компиляции и работы пользовательской программы.

В зависимости от приложения, не всегда получится применить некоторые рекомендации. Тем не менее, обьясняется почему некоторые методы в программировании позволяют уменьшить время компиляции и работы программы.

Помимо пользовательских блоков, существуют также внутренние системные блоки, создаемые автоматически. Такие блоки гарантируют надежную работу программы безопасности. Внутренние системные блоки увеличивают время компиляции и работы программы.

Пример

Рисунок 5-7: Пользовательские и созданные системой F-блоки

5.12.1 Отказ от использования блоков, влияющих на время: TP, TON, TOF

Каждый такой блок (TP, TON, TOF) требует дополнительных корректировок в блоках и глобальных данных в программном коде программы безопасности.

Рекомендация

Старайтесь как можно реже использовать данные блоки.

5.12.2 Отказ от использования вложенных вызовов

Вложенные вызовы увеличивают обьем сгенерированных системой F блоков, поскольку возникает необходимость в значительном увеличении процедур проверки функций безопасности. Если глубина вложения превысила 8, Вы увидите предупреждение в TIA Portal на этапе компиляции.

Рекомендация

Структурируйте Вашу программу, таким образом, чтобы Вам не потребовалось использования вложенных вызовов.

5.12.3 Разделение стандартной программы и программы безопасности

В сложных проектах, часто требуется произвести обмен данными между стандартной программой и программой безопасности. Если обмен реализован с помощью переменных (например, меркеров), то изменения в стандартной программе могут потребовать компиляцию программы безопасности. Для загрузки изменений, CPU необходимо перевести в режим STOP.

Рекомендация

Используйте стандартные DB (см. главу 5.13 Обмен данными между стандартной и F-программой). Изменения в стандартной программе не будут касаться программы безопасности. Для загрузки стандартной программы не потребуется перевод контроллера в режим STOP, .

5.12.4 Использование мультиэкземпляров

При вызове одного экземплярного DB в нескольких местах программы безопасности, такой DB должен быть обработан более чем один раз за один цикл. Такая обработка требует большего количества внутренних системных F блоков.

Рекомендация

По возможности, всегда используйте мультиэкземпляры. Таким образом, Вы уменьшите количество внутренних системных F блоков.

5.12.5 Отказ от использования инструкций JMP/label

Если блок вызывается через переход на метку (JMP/LABEL), это требует дополнительной обработки в системных внутренних F блоках. В данном случае, при переходе к вызванному блоку, должна запускаться корректировка кода. Данные инструкции снижают производительность и увеличивают время компиляции программы.

Рекомендация

Старайтесь избегать использования инструкций JMP/Label, чтобы снизить количество внутренних системных F блоков.

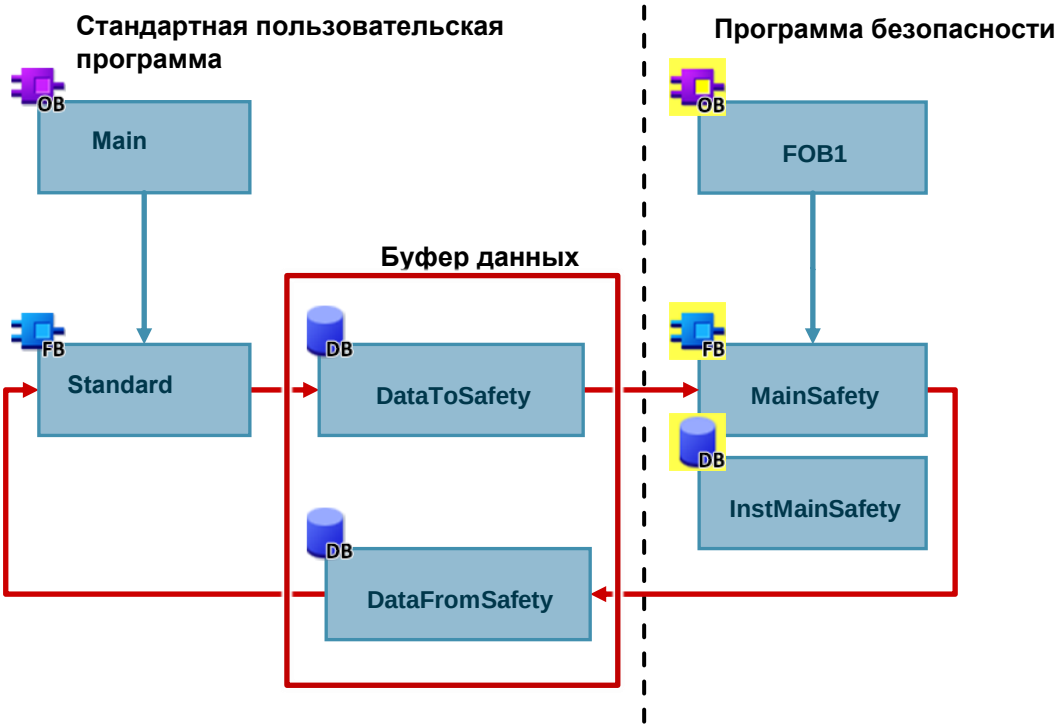
5.13 Обмен данными между стандартной и F-программой

В некоторых случаях, требуется произвести обмен данными между стандартной программой и программой безопасности. Необходимо придерживаться следующих рекомендаций, чтобы гарантировать консистентность данных стандартной программы и программы безопасности.

Рекомендации

- Не производить обмен данными через меркеры (см. главу 4.2 Переход от использования меркерной памяти к глобальным блокам данных
- Сосредоточьте доступ между программой безопасности и стандартной програмой в двух стандартных DB.

Рисунок 5-8: Обмен данными между стандартной программой и программой безопасности



5.14 Тестирование программы безопасности

В дополнение к всегда контролируемым данным стандартной пользовательской программы, Вы можете изменить следующие даные программы безопасности при деактивированном режиме защиты.

- Область отображения F−I/O
- F−DB (за исключением DB для F−runtime коммуникационной группы), экземпляры−DB F-FB
- F-I/O DB

Свойства

- Доступ к F−I/O возможен только если F−CPU в режиме RUN.
- В таблице наблюдений, Вы можете наблюдать максимум за 5 входами/ выходами в программе безопасности.
- Вы можете использовать несколько таблиц наблюдения.
- Триггер необходимо установить в “permanent” (постоянно) или “once” (однократно) для “cycle start” (начало цикла) или “cycle end” (конец цикла).
- Форсирование недоступно для F-I/O.
- Установка точек прерывания в стандартной программе, повлечет за собой появление ошибок в программе безопасности:
  - Превышение контрольного времени F цикла
  - Ошибки в процессе коммуникации с F-I/O
  - Ошибки в процессе коммуникации F-CPU-CPU
  - Внутренние ошибки CPU
- Если, Вам все же необходимо использовать точки прерывания для тестирования, то Вы должны деактивировать режим защиты. Что приведет к следующим ошибкам:
  - Ошибки в процессе коммуникации с F-I/O
  - Ошибки в процессе коммуникации F-CPU-CPU

5.15 STOP режим в случае появления F-ошибок

В следующих случая, F-CPU перейдет в режим STOP:

- В папке "System blocks" (Системные блоки), Вы не должны добавлять, изменять или удалять никакие блоки.
- В случае, если результат выполнения инструкции выходит за допустимые границы типа данных (переполнение). Причина появления диагностического события записывается в диагностический буфер F CPU.
- В программе должно отсутствовать обращение к любым экземплярным DB F-FB, которые не вызываются в программе безопасности.
- Если “Maximal cycle time of the F run-time group” (Максимальное время цикла F-runtime группы) было превышено, то F-CPU перейдет в STOP. Задайте максимальное разрешенное время для "Maximum cycle time der F run-time group" (Максимальное время цикла F-runtime группы), которое должно истечь между двумя вызовами данной F run-time группы (максимально 20,000 мс).
- Если F run-time группа, из которой должны читаться теги DB для F-run-time группы (главный блок F run-time группы), то F-CPU перейдет в режим STOP.
- Изменение стартовых значений экземплярных DB F-FB запрещено online и offline, может привести F-CPU в STOP.
- Главный блок программы безопасности не должен содержать необъявленных параметров.
- Выходы F-FC должны быть всегда инициализированы.

5.16 Миграция программ безопасности

Информация по миграции программ безопасности доступна по ссылке:

https://support.industry.siemens.com/cs/ww/en/view/109475826

5.17 Основные рекомендации по безопасности

При работе с STEP 7 Safety и F модулями в основном, необходимо придерживаться следующих рекомендаций.

- По возможности, всегда используйте F контроллеры. В дальнейшем, расширение функции безопасности могут быть реализованы без особого труда.
- Всегда используйте один пароль для программы безопасности, чтобы предотвратить неавторизованный доступ. Пароль задается в редакторе “Safety administration” (Управление безопасностью).

6 Наиболее важные рекомендации

- Использование оптимизированных блоков
  - Глава 2.6 Оптимизированные блоки
- Использование типа данных VARIANT вместо ANY
  - Глава 2.8.5 Тип данных VARIANT
- Структурирование программы
  - Глава 3.2 Организационные блоки (OB)
- Добавление инструкций в качестве мультиэкземпляра (TON, TOF ..)
  - Глава 3.2.5 Мультиэкземпляры
- Повторное использование блоков
  - Глава 3.2.8 Возможность повторного использования блоков
- Символьное программирование
  - Глава 3.6 Символьная адресация
- При работе с данными, используйте массив
  - Глава 3.6.2 Тип данных ARRAY и косвенный доступ к элементам
- Создание PLC data types
  - Глава 3.6.4 Доступ к областям ввода/вывода с помощью PLC data types
- Использование библиотек для хранения элементов программы
  - Глава 3.7 Библиотеки
- Переход от меркерной памяти к глобальным блокам данных
  - Глава 4.2 Переход от меркерной памяти к глобальным блокам данных

7 Использованная литература

Таблица 7-1

	Информация	Ссылка
\1\	Техническая поддержка Siemens Industry Online	https://support.industry.siemens.com/cs/start?lc=en-DE
\2\	Страница загрузок	https://support.industry.siemens.com/cs/ww/en/view/81318674
\3\	Руководство по программированию S7-1200 и S7-1500	https://support.industry.siemens.com/cs/ww/en/view/81318674
\4\	TIA Portal - Обзор наиболее важных документов и ссылки	https://support.industry.siemens.com/cs/ww/en/view/65601780
\5\	Руководства по STEP 7 (TIA Portal)	https://support.industry.siemens.com/cs/ww/en/ps/14673/man
\6\	Руководства по S7-1200	https://support.industry.siemens.com/cs/ww/en/ps/13683/man
\7\	Руководства по S7-1500 (F)	https://support.industry.siemens.com/cs/ww/de/ps/13716/man
\8\	Руководства ET 200SP CPU	https://support.industry.siemens.com/cs/ww/en/ps/13888/man
\9\	Начало работы с S7-1200	https://support.industry.siemens.com/cs/ww/de/view/39644875
\10\	Начало работы с S7-1500	https://support.industry.siemens.com/cs/ww/de/view/78027451
\11\	Сравнение языков программирования SIMATIC S7-1200 / S7-1500	http://support.automation.siemens.com/WW/view/en/86630375

8 История

Таблица 8-1

Версия	Дата	Изменения
V1.0	09/2013	Первая версия
V1.1	10/2013	Следующие главы были изменены: 2.6.3 Оптимальный вариант хранения данных в S7-1500 2.13 Внутренний ссылочный ID для контроллера и тегов HMI 3.2.2 Функции (FC) 3.2.3 Функциональные блоки (FB) 3.4.3 Локальная память
V1.2	03/2014	Добавлены главы: 2.6.4 Преобразование между оптимизированными и неоптимизированными переменными 2.6.6 Коммуникация с помощью оптимизированных блоков 2.9.2 Инструкции MOVE 2.9.3 Инструкции с VARIANT 3.6.4 Доступ к областям ввода/вывода с помощью PLC data types Дополнены следующие главы: 2.2 Термины 2.3 Языки программирования 2.6 Оптимизированные блоки 2.10 Символы и комментарии 3.2 Программные блоки 3.5 Сохранямость 4.3 Программирование "синхробайта" Некоторые изменения в различных главах
V1.3	09/2014	Добавлены главы: 2.8.4 Типы данных Unicode 2.10.2 Комментарии в таблице наблюдений 2.12 Пользовательские константы 3.2.9 Автоматическое назначения номера блокам 5 STEP 7 Safety в TIA Portal Дополнены следующие главы: 2.7 Размер блока 2.8 Новые типы данных для S7-1200/1500 2.9 Инструкции 2.10 Символы и комментарии 3.6.3 Тип данных STRUCT и PLC data types 3.7 Библиотеки Некоторые изменения в различных главах

Версия	Дата	Изменения
1.4	11/2015	Добавлены главы: 2.6.5 Передача параметров между блоками с оптимизированным доступом и стандартным доступом 3.3.3 Передача параметров 3.10.4 Применение циклов FOR, REPEAT и WHILE 5.12 Оптимзация компиляции и работы программы