

SYSMAC

**CX-Programmer, версия 9.□
CXONE-AL□□C-V4/AL□□D-V4**

**РУКОВОДСТВО
ПОЛЬЗОВАТЕЛЯ
Функциональные
блоки /
Структурированный
текст**

OMRON

SYSMAC

CX-Programmer версии 9.□

CXONE-AL□□C-V4/AL□□D-V4




**Руководство пользователя
Функциональные блоки и
структурированный текст**

Версия: январь, 2011 г.

Замечание:

Продукты компании OMRON должны использоваться надлежащим образом, только для целей, описанных в настоящем руководстве, и только квалифицированным персоналом.

В настоящем руководстве для обозначения различных типов опасности используются следующие предупреждающие знаки и надписи. Обязательно учитывайте информацию, которую они содержат. Пренебрежение этой информацией может стать причиной несчастного случая или материального ущерба.

-  **ОПАСНОСТЬ** Указывает на чрезвычайно опасную ситуацию, которая, если не принять меры к ее устранению, приведет к смерти или серьезной травме. Кроме того, может быть нанесен значительный материальный ущерб.
-  **ВНИМАНИЕ** Указывает на потенциально опасную ситуацию, которая, если не принять меры к ее устранению, может привести к смерти или серьезной травме. Кроме того, может быть нанесен значительный материальный ущерб.
-  **Предупреждение** Указывает на потенциально опасную ситуацию, которая, если не принять меры к ее устранению, может привести к травме средней или легкой степени тяжести либо нанесению материального ущерба.

Вспомогательные обозначения

Сокращение «Ch», которое появляется на некоторых дисплеях и на некоторых продуктах OMRON, часто означает «слово» и в документации в этом смысле имеет сокращение «Wd».

Сокращение «PLC» (ПЛК) означает «Программируемый контроллер». Однако на некоторых экранах CX-Programmer может встречаться сокращение «PC», которое также означает «программируемый контроллер».

Информационные знаки

Для выделения информации различного типа в левой колонке настоящего руководства используются следующие заголовки и обозначения.

Примечание. Особенно интересная и полезная информация о наиболее эффективных и удобных способах работы с изделиями.

- 1,2,3...** 1. Обозначение последовательности действий, перечня или любого другого списка.

© OMRON, 2008

Все права защищены. Воспроизведение, размещение в информационно-поисковой системе или передача третьему лицу какой-либо части настоящего руководства в какой-либо форме и каким-либо способом (механическим, электронным, путем ксерокопирования, записи на носитель или иным способом) не допускается без предварительного письменного разрешения компании OMRON.

Использование информации, содержащейся в настоящем руководстве, не сопряжено с какой-либо патентной ответственностью. Кроме того, поскольку компания OMRON неуклонно стремится к совершенствованию своей продукции, информация, содержащаяся в настоящем руководстве, может быть изменена без предупреждения. Подготовка настоящего руководства выполнялась с надлежащей тщательностью. Тем не менее, компания OMRON не несет ответственности за какие-либо ошибки и упущения. Компания OMRON не несет юридической ответственности за повреждения, явившиеся результатом использования информации, содержащейся в настоящем руководстве.

Часть 1:

Функциональные блоки

РАЗДЕЛ 1 Введение в функциональные блоки

РАЗДЕЛ 2 Характеристики функциональных
блоков

РАЗДЕЛ 3 Создание функциональных блоков

Часть 2:

Структурированный текст

РАЗДЕЛ 4 Введение в структурированный текст

РАЗДЕЛ 5 Характеристики языка
структурированного текста

РАЗДЕЛ 6 Создание программ на языке
структурированного текста

Приложения

СОДЕРЖАНИЕ

МЕРЫ ПРЕДОСТОРОЖНОСТИ	xxi
1 Для кого предназначено руководство	xxii
2 Меры предосторожности общего характера	xxii
3 Меры предосторожности и обеспечения безопасности	xxii
4 Меры предосторожности при эксплуатации	xxiv

Часть 1: Функциональные блоки

РАЗДЕЛ 1

Введение в функциональные блоки	3
1-1 Вводная информация о функциональных блоках	4
1-2 Функциональные блоки	14
1-3 Переменные	22
1-4 Преобразование определений функциональных блоков в библиотечные файлы	28
1-5 Порядок использования	28
1-6 Информация об изменениях в новых версиях	30

РАЗДЕЛ 2

Характеристики функциональных блоков	37
2-1 Характеристики функциональных блоков	38
2-2 Типы данных, поддерживаемые в функциональных блоках	52
2-3 Характеристики экземпляров	53
2-4 Ограничения при программировании	63
2-5 Указания по использованию функциональных блоков	69
2-6 Меры предосторожности в отношении команд с операндами, указывающими первое или последнее слово группы слов	80
2-7 Поддерживаемые команды и ограничения в отношении операндов	84
2-8 Характеристики функциональных блоков в модулях ЦПУ	85
2-9 Количество шагов в программах функциональных блоков и время выполнения экземпляра ..	93

РАЗДЕЛ 3

Создание функциональных блоков	97
3-1 Общий порядок действий	98
3-2 Порядок действий	100

Часть 2: Структурированный текст (ST)

РАЗДЕЛ 4

Введение в структурированный текст	157
4-1 Язык структурированного текста	158
4-2 Характеристики CX-Programmer	159

СОДЕРЖАНИЕ

РАЗДЕЛ 5

Характеристики языка структурированного текста.	161
5-1 Характеристики языка структурированного текста	162
5-2 Допустимые типы данных в программах на языке ST.	163
5-3 Ввод программ на языке структурированного текста	164
5-4 Элементы синтаксиса языка структурированного текста	169
5-5 Описание выражений языка структурированного текста	183
5-6 Пример программы на языке структурированного текста.	204
5-7 Ограничения.	204

РАЗДЕЛ 6

Создание программ на языке структурированного текста .	207
6-1 Порядок действий	208

Приложения

A Системные внешние переменные, поддерживаемые в функциональных блоках	221
B Ошибки программы на языке структурированного текста	223
C Описание функций	227

Указатель	253
----------------------------	------------

Перечень версий	255
----------------------------------	------------

О данном руководстве:

В настоящем руководстве описаны функции и операции, предусмотренные в программном обеспечении CX-Programmer для создания программ на языке функциональных блоков и на языке структурированного текста (ST). Данные функции программного обеспечения CX-Programmer поддерживаются модулями ЦПУ серии CJ2H, CJ2M, модулями ЦПУ серии CS1-H, CJ1-H и CJ1M с версией модуля не ниже 3.0, модулями ЦПУ серии CP, а также контроллерами серии NSJ и FQM1.

Некоторые из этих функций, однако, поддерживаются только модулями ЦПУ CJ2H, а также модулями ЦПУ CS1-H, CJ1-H и CJ1M с версией модуля не ниже 4.0.

Подробные сведения смотрите в разделе *1-6 Информация об изменениях в новых версиях*.

Информацию о функциях, не связанных с функциональными блоками и структурированным текстом, следует искать в одном из следующих руководств.

- CX-Programmer
 - : CX-Programmer — Руководство по работе (W446) и CX-Programmer — Руководство по работе: SFC (W469)
- Модуль ЦПУ
 - : Руководство по эксплуатации для программируемых контроллеров серии CS, серии CJ, серии CP и серии NSJ

Руководства по работе с программным обеспечением CX-Programmer версии 9□

Название	Cat. No.	Содержание
CXONE-AL□□C-V4/CXONE-AL□□D-V4 CX-Programmer — Руководство по работе Функциональные блоки и структурированный текст	W447 (настоящее руководство)	Поясняются функции программного обеспечения CX-Programmer, предназначенные для работы с функциональными блоками и структурированным текстом. Другие совместно используемые функции CX-Programmer описаны в руководстве <i>CX-Programmer — Руководство по работе (W446)</i> .
CXONE-AL□□C-V4/CXONE-AL□□D-V4 CX-Programmer — Руководство по работе	W446	Содержит сведения об установке и работе с программой CX-Programmer и обо всех ее функциях, за исключением функциональных блоков.
CXONE-AL□□C-V4/CXONE-AL□□D-V4 CX-Programmer — Руководство по работе: последовательные функциональные схемы	W469	Поясняется применение функций для создания программ на языке SFC. Другие совместно используемые функции CX-Programmer описаны в руководстве <i>CX-Programmer — Руководство по работе (W446)</i> .
CX-Net — Руководство по работе	W362	Содержит сведения об устройстве и конфигурировании сетей, в частности содержит информацию о настройке логических связей, таблиц маршрутизации и параметров модулей.
CXONE-AL□□C-V4/CXONE-AL□□D-V4 CX-Integrator — Руководство по работе	W445	Описание и порядок работы с программой CX-Integrator.

Руководства для модулей ЦПУ серии CJ2H, CJ2M, CS1-H, CJ1-H и CJ1M

Название	Cat. No.	Содержание
Серия SYSMAC CJ CJ2H-CPU6□-EIP, CJ2H-CPU6□ CJ2M-CPU1□, CJ2M-CPU3□ Программируемые контроллеры — Аппаратные средства. Руководство пользователя	W472	Содержит общие сведения о модулях ЦПУ серии CJ2 и описывает основные операции, включая проектирование, монтаж и обслуживание. Руководство содержит следующую информацию: общие сведения, основные свойства и функции; конфигурация системы; механический и электрический монтаж; поиск и устранение неисправностей. Данное руководство следует использовать совместно с W473.
Серия SYSMAC CJ CJ2H-CPU6□-EIP, CJ2H-CPU6□ CJ2M-CPU1□, CJ2M-CPU3□ Программируемые контроллеры — Программное обеспечение. Руководство пользователя	W473	Описывает программирование и другие возможные способы использования функций модулей ЦПУ CJ2. Руководство содержит следующую информацию: функционирование модуля ЦПУ; внутренние области памяти; программирование; задачи; встроенные функции модулей ЦПУ. Данное руководство следует использовать совместно с W472.
Серия SYSMAC CS/CJ CS1G/H-CPU□□-EV1, CS1G/H-CPU□□H, CS1D-CPU□□H, CS1D-CPU□□S, CJ2H-CPU6□-EIP, CJ2H-CPU6□, CJ2M-CPU1□, CJ2M-CPU3□ CJ1H-CPU□□H-R CJ1G-CPU□□, CJ1G/H-CPU□□H, CJ1G-CPU□□P, CJ1M-CPU□□ Серия SYSMAC One NSJ NSJ□-□□□□(B)-G5D NSJ□-□□□□(B)-M3D Программируемые контроллеры Справочное руководство по командам программирования	W474	Содержит описание команд программирования на языке лестничных диаграмм (LD), поддерживаемых программируемыми контроллерами серии CS/CJ или NSJ. В процессе программирования данное руководство следует использовать вместе с <i>Руководством по эксплуатации или Руководством пользователя по аппаратному обеспечению (W339 (CS1), CJ1 (W393) или CJ2 (W472)) и Руководством по программированию или Руководством пользователя по программному обеспечению (W394 (CS1/CJ1) или W473 (CJ2))</i> .
Серия SYSMAC CS CS1G/H-CPU□□-EV1, CS1G/H-CPU□□H Программируемые контроллеры Руководство по эксплуатации	W339	Содержит общие сведения о ПЛК серии CS и описывает основные операции, включая проектирование, монтаж и обслуживание. Руководство содержит следующую информацию: общие сведения и основные свойства; конфигурация системы; механический и электрический монтаж; распределение адресов памяти ввода/вывода; поиск и устранение неисправностей. Данное руководство следует использовать совместно с W394.
Серия SYSMAC CJ CJ1G-CPU□□, CJ1G/H-CPU□□H, CJ1H-CPU□□H-R, CJ1G-CPU□□P, CJ1M-CPU□□ Программируемые контроллеры Руководство по эксплуатации	W393	Содержит общие сведения о ПЛК серии CJ и описывает основные операции, включая проектирование, монтаж и обслуживание. Руководство содержит следующую информацию: общие сведения и основные свойства; конфигурация системы; механический и электрический монтаж; распределение адресов памяти ввода/вывода; поиск и устранение неисправностей. Данное руководство следует использовать совместно с W394.

Название	Cat. No.	Содержание
Серия SYSMAC CS/CJ CS1G/H-CPU□□-EV1, CS1G/H-CPU□□H, CJ1G-CPU□□, CJ1G/H-CPU□□H, CJ1H-CPU□□H-R, CJ1G-CPU□□P, CJ1M-CPU□□, NSJ□-□□□□(B)-G5D, NSJ□-□□□□(B)-M3D Программируемые контроллеры Руководство по программированию	W394	Описывает программирование и другие способы использования функций ПЛК серий CS/CJ и NSJ. Руководство содержит следующую информацию: программирование; задачи; память файлов; прочие функции. Данное руководство следует использовать совместно с W339 или W393.
Серия SYSMAC CS/CJ CS1G/H-CPU□□-EV1, CS1G/H-CPU□□H, CJ1G-CPU□□, CJ1G/H-CPU□□H, CJ1H-CPU□□H-R, CJ1G-CPU□□P, CJ1M-CPU□□, NSJ□-□□□□(B)-G5D, NSJ□-□□□□(B)-M3D Программируемые контроллеры Справочное руководство по командам программирования	W340	Содержит описание команд программирования на языке лестничных диаграмм (LD), поддерживаемых программируемыми контроллерами серии CS/CJ и NSJ. В процессе программирования данное руководство следует использовать вместе с <i>Руководством по эксплуатации (W339 (CS1) или W393 (CJ1))</i> и <i>Руководством по программированию (W394)</i> .
CS1G/H-CPU□□-EV1, CS1G/H-CPU□□H CS1W-SCB21-V1/41-V1, CS1W-SCU21/41 CJ2H-CPU6□-EIP, CJ2H-CPU6□ CJ2M-CPU1□, CJ2M-CPU3□ CJ1G-CPU□□, CJ1G/H-CPU□□H, CJ1G-CPU□□P CJ1M-CPU□□ CJ1W-SCU21-V1/41-V1 CP1L-M/L-□□□-□ CP1H-X□□□□-□, CP1H-XA□□□□-□, CP1H-Y□□□□-□ CP1E-E□□□□-□, CP1E-N□□□□-□ NSJ□-□□□□(B)-G5D, NSJ□-□□□□(B)- M3D Серия SYSMAC CS/CJ Справочное руководство по командам связи	W342	Содержит описание команд связи, которые можно использовать для обмена данными с модулями ЦПУ серии CS/CJ. Руководство содержит следующую информацию: команды серии C (Host Link); команды FINS. Примечание. В данном руководстве описываются команды, которые могут передаваться модулю ЦПУ, безотносительно используемого порта связи, которым может быть последовательный порт модуля ЦПУ, порт модуля/платы последовательного интерфейса или порт любого другого модуля связи.

Руководство для контроллера серии NSJ

Сведения о характеристиках и способах работы с контроллером серии NSJ, которые не содержатся в настоящем руководстве, можно найти в следующем руководстве.

Cat. No.	Модели	Название	Описание
W452	NSJ5-TQ□□(B)-G5D NSJ5-SQ□□(B)-G5D NSJ8-TV□□(B)-G5D NSJ10-TV□□(B)-G5D NSJ12-TS□□(B)-G5D	Серия NSJ Руководство по эксплуатации	Предоставляет следующую информацию о контроллерах серии NSJ: краткий обзор, свойства и функции; проектирование и конфигурация системы; механический и электрический монтаж; распределение адресов памяти ввода/вывода; устранение неисправностей и техническое обслуживание. Используйте данное руководство совместно со следующими руководствами: Серия SYSMAC CS — Руководство по эксплуатации (W339), Серия SYSMAC CJ — Руководство по эксплуатации (W393), Серия SYSMAC CS/CJ — Руководство по программированию (W394) и Серия NS-V1/-V2 — Руководство по установке и настройке (V083)

Руководства для контроллеров серии FQM1 (версия модуля 3.0 и выше)

Сведения о характеристиках и способах работы с контроллерами серии FQM1 версии 3.0 (FQM1-CM002/MMP22/MMA22), которые отсутствуют в настоящем руководстве, можно найти в следующих руководствах.

Cat. No.	Модели	Название	Описание
O012	FQM1-CM002 FQM1-MMP22 FQM1-MMA22	Серия FQM1 Руководство по эксплуатации	Предоставляет следующую информацию о контроллерах серии FQM1 (версия модуля 3.0): краткий обзор, свойства и функции; проектирование и конфигурация системы; механический и электрический монтаж; распределение адресов памяти ввода/вывода; устранение неисправностей и техническое обслуживание.
O013	FQM1-CM002 FQM1-MMP22 FQM1-MMA22	Серия FQM1 Справочное руководство по командам программирования	Содержит описание каждой команды, которая используется для программирования контроллеров FQM1. В процессе программирования данное руководство следует использовать совместно с руководством <i>Серия FQM1 — Руководство по эксплуатации (O012)</i> .

Руководства для программируемых контроллеров серии CP

Сведения о характеристиках и способах работы с модулями ЦПУ серии CP, которые не содержатся в настоящем руководстве, можно найти в следующих руководствах.

Cat. No.	Модели	Название	Описание
W450	CP1H-X□□□□-□ CP1H-XA□□□□-□ CP1H-Y□□□□-□	Серия SYSMAC CP Модули ЦПУ CP1H — Руководство по эксплуатации	Содержит следующую информацию о ПЛК CP1H серии CP: • общие сведения/основные свойства; • конфигурация системы; • механический и электрический монтаж; • адресное пространство памяти ввода/вывода; • поиск и устранение неисправностей. Данное руководство следует использовать совместно с руководством <i>ПЛК CP1H/CP1L — Руководство по программированию (W451)</i> .
W462	CP1L-M□□□□-□ CP1L-L□□□□-□	Серия SYSMAC CP Модули ЦПУ CP1L — Руководство по эксплуатации	Содержит следующую информацию о ПЛК CP1L серии CP: • общие сведения/основные свойства; • конфигурация системы; • механический и электрический монтаж; • адресное пространство памяти ввода/вывода; • поиск и устранение неисправностей. Данное руководство следует использовать совместно с руководством <i>ПЛК CP1H/CP1L — Руководство по программированию (W451)</i> .
W451	CP1H-X□□□□-□ CP1H-XA□□□□-□ CP1H-Y□□□□-□ CP1L-M□□□□-□ CP1L-L□□□□-□	Серия SYSMAC CP Модули ЦПУ CP1H/ CP1L — Руководство по программированию	Содержит следующую информацию о ПЛК CP1H и CP1L серии CP: • команды языка программирования; • методы программирования; • задачи. Данное руководство следует использовать совместно с руководством <i>ПЛК CP1H/CP1L — Руководство по эксплуатации (W450)</i> .

Установка в качестве компонента CX-One

Подробную информацию о порядке установки программного обеспечения CX-Programmer как одного из компонентов CX-One (программного пакета для систем промышленной автоматизации) см. в руководстве *CX-One, версия 3.0 — Руководство по установке*, которое поставляется в комплекте с CX-One.

Cat. No.	Модель	Название руководства	Содержание
W463	CXONE-AL□□C-V4/ AL□□D-V4	CX-One — Руководство по установке	Процедура установки и краткое описание программного пакета автоматизации CX-One.

Краткое содержание частей и разделов руководства

Раздел *Меры предосторожности* содержит общие меры предосторожности, которые должны соблюдаться при работе с программным обеспечением CX-Programmer.

Часть 1

Часть 1 состоит из следующих разделов.

Раздел 1 содержит общие сведения о функциях программного обеспечения CX-Programmer, предназначенных для работы с функциональными блоками; поясняет функции, которые отсутствуют в версиях CX-Programmer, не поддерживающих функциональные блоки.

Раздел 2 содержит технические данные и характеристики, которыми следует руководствоваться при использовании функциональных блоков, в том числе технические характеристики функциональных блоков, экземпляров и совместимых ПЛК, а также меры предосторожности и указания по использованию.

Раздел 3 описывает порядок действий по созданию функциональных блоков в программе CX-Programmer.

Часть 2

Часть 2 состоит из следующих разделов.

Раздел 4 содержит общие сведения о функциях программного обеспечения CX-Programmer, предназначенных для создания программ на языке структурированного текста; поясняет функции, которые отсутствуют в версиях CX-Programmer, не поддерживающих язык структурированного текста.

Раздел 5 содержит технические данные и характеристики, которыми следует руководствоваться при создании программ на языке структурированного текста, а также примеры программ и сведения об ограничениях.

Раздел 6 подробно описывает процесс создания программ на языке структурированного текста.

Приложения предоставляют информацию об ошибках, которые могут возникать при программировании на языке структурированного текста, а также содержат описание функций, предусмотренных для программирования на языке структурированного текста.



ВНИМАНИЕ Пренебрежение сведениями, содержащимися в настоящем руководстве, может стать причиной несчастного случая, возможно, со смертельным исходом, либо может привести к повреждению изделия или выходу его из строя. Прочитайте, пожалуйста, каждый раздел целиком, внимательно изучив информацию, содержащуюся в разделе и в разделах, с ним связанных, прежде чем приступать к какой-либо из описанных операций или действий.

Внимательно прочитайте настоящее руководство

Пожалуйста, внимательно прочитайте это руководство, прежде чем приступить к использованию продукта. В случае если у Вас имеются какие-либо вопросы или комментарии, обращайтесь, пожалуйста, к региональному представителю компании OMRON.

Гарантийные обязательства и ограничение ответственности

ГАРАНТИЙНЫЕ ОБЯЗАТЕЛЬСТВА

- (1) Гарантийный срок для Программного обеспечения составляет один год либо с даты приобретения, либо с даты доставки Программного обеспечения в указанный пункт назначения.
- (2) Если Пользователь обнаружит дефект в Программном обеспечении (то есть значительное расхождение с описанием в руководстве пользователя) и вернет Программное обеспечение компании OMRON в течение указанного выше гарантийного срока, компания OMRON бесплатно заменит Программное обеспечение и предоставит его Пользователю на физическом носителе или через Интернет. Если Пользователь обнаружит дефект в носителе, ответственность за который может быть возложена на компанию OMRON, и вернет Программное обеспечение компании OMRON в течение указанного выше гарантийного срока, компания OMRON бесплатно заменит дефектный носитель. Если компания OMRON не может заменить дефектный носитель или устранить дефект в программном обеспечении, ответственность компании OMRON ограничивается возмещением Пользователю денежных средств, уплаченных Пользователем компании OMRON за приобретение лицензии на Программное обеспечение.

ОГРАНИЧЕНИЕ ОТВЕТСТВЕННОСТИ

- (1) ПРИВЕДЕННЫЕ ВЫШЕ ГАРАНТИЙНЫЕ ОБЯЗАТЕЛЬСТВА ЯВЛЯЮТСЯ ЕДИНСТВЕННЫМ И ИСКЛЮЧИТЕЛЬНЫМ СПОСОБОМ УДОВЛЕТВОРЕНИЯ ПРЕТЕНЗИЙ ПОЛЬЗОВАТЕЛЯ К КОМПАНИИ OMRON. НЕ СУЩЕСТВУЕТ НИКАКИХ ДРУГИХ ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ (НО НЕ ОГРАНИЧИВАЯСЬ ТАКОВЫМИ) ГАРАНТИИ В ОТНОШЕНИИ ПРИГОДНОСТИ ДЛЯ ПРОДАЖИ ИЛИ ВОЗМОЖНОСТИ ПРИМЕНЕНИЯ В КОНКРЕТНЫХ ЦЕЛЯХ. НИ ПРИ КАКИХ ОБСТОЯТЕЛЬСТВАХ КОМПАНИЯ OMRON НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ЗА ЛЮБУЮ УПУЩЕННУЮ ВЫГОДУ ИЛИ ИНЫЕ КОСВЕННЫЕ, СЛУЧАЙНЫЕ, ПОБОЧНЫЕ ИЛИ ФАКТИЧЕСКИЕ УБЫТКИ, ПРИЧИНОЙ ВОЗНИКНОВЕНИЯ КОТОРЫХ ЯВИЛОСЬ ИСПОЛЬЗОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.
- (2) КОМПАНИЯ OMRON НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ЗА ДЕФЕКТЫ В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ, ВОЗНИКШИЕ В РЕЗУЛЬТАТЕ ВНЕСЕНИЯ ЛЮБЫХ ИЗМЕНЕНИЙ В ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПОЛЬЗОВАТЕЛЕМ ИЛИ ЛЮБОЙ ТРЕТЬЕЙ СТОРОНОЙ.
- (3) КОМПАНИЯ OMRON НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ЗА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, РАЗРАБОТАННОЕ ПОЛЬЗОВАТЕЛЕМ ИЛИ ЛЮБОЙ ТРЕТЬЕЙ СТОРОНОЙ С ИСПОЛЬЗОВАНИЕМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОМПАНИИ OMRON, РАВНО КАК ЗА ПОСЛЕДСТВИЯ ЕГО ПРИМЕНЕНИЯ.

Замечания по применению

ПРИГОДНОСТЬ ДЛЯ КОНКРЕТНОГО ПРИМЕНЕНИЯ

ПОЛЬЗОВАТЕЛЬ НЕ ДОЛЖЕН ИСПОЛЬЗОВАТЬ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ЦЕЛЕЙ, КОТОРЫЕ НЕ ОПИСАНЫ В ПРИЛАГАЕМОМ РУКОВОДСТВЕ ПОЛЬЗОВАТЕЛЯ.

Отказ от ответственности

ИЗМЕНЕНИЕ ХАРАКТЕРИСТИК

Характеристики программного обеспечения и дополнительные принадлежности могут быть изменены в любое время с целью внесения улучшений и по другим причинам.

ПРЕДОСТАВЛЯЕМЫЕ УСЛУГИ

В стоимость лицензии на программное обеспечение не включена стоимость услуг, таких как предоставление технических специалистов.

ОШИБКИ И ОПЕЧАТКИ

Информация, содержащаяся в настоящем руководстве, была тщательно проверена и, вероятнее всего, является точной; тем не менее, компания OMRON не несет ответственности за допущенные типографские и редакторские ошибки и опечатки.

МЕРЫ ПРЕДОСТОРОЖНОСТИ

Данный раздел содержит общие меры предосторожности, которые требуется соблюдать при работе с программным обеспечением CX-Programmer и программируемым логическим контроллером.

Данный раздел содержит важную информацию о безотказном и безопасном применении программного обеспечения CX-Programmer и программируемого контроллера. Обязательно прочитайте этот раздел и примите к сведению всю содержащуюся в нем информацию, прежде чем приступать к настройке или использованию программного обеспечения CX-Programmer и программируемого контроллера.

1	Для кого предназначено руководство	xxii
2	Меры предосторожности общего характера.	xxii
3	Меры предосторожности и обеспечения безопасности	xxii
4	Меры предосторожности при эксплуатации	xxiv

1 Для кого предназначено руководство

Данное руководство предназначено для лиц, обладающих специальными знаниями в области электрических систем (инженер-электрик и т. п.).

- Персонал, ответственный за установку промышленных систем автоматизации.
- Персонал, ответственный за разработку промышленных систем автоматизации.
- Персонал, ответственный за администрирование оборудования промышленных систем автоматизации.

2 Меры предосторожности общего характера

Пользователь должен применять изделие в соответствии с эксплуатационными характеристиками, приведенными в руководствах по эксплуатации.

Прежде чем использовать изделие в условиях, которые не описаны в руководстве, а также в случае применения изделия в системах управления объектами ядерной энергетики, в железнодорожных системах, в авиации, в транспортных средствах, в тепловых системах, в медицинском оборудовании, в игровых автоматах и аттракционах, в оборудовании защиты и других системах, машинах и установках, которые могут серьезно повлиять на здоровье людей и привести к повреждению имущества при условии неправильной эксплуатации, обязательно проконсультируйтесь в ближайшем представительстве компании Omron.

Убедитесь в том, что номинальные значения и рабочие характеристики изделия достаточны для систем, машин и оборудования, и предусматривайте в системах, машинах и оборудовании механизмы удвоенной надежности.

Данное руководство содержит сведения о программировании и эксплуатации изделия. Прежде чем приступить к эксплуатации изделия, обязательно прочитайте данное руководство и храните его в легко доступном месте, чтобы использовать во время работы.

ВНИМАНИЕ

Очень важно, чтобы ПЛК и все его модули использовались только для оговоренных целей и только при оговоренных условиях эксплуатации, особенно в тех приложениях, в которых они могут прямым или косвенным образом повлиять на здоровье человека. Прежде чем применять систему ПЛК в описанных выше приложениях, обязательно проконсультируйтесь в ближайшем представительстве компании Omron.

3 Меры предосторожности и обеспечения безопасности

ВНИМАНИЕ

Прежде чем передавать содержимое области памяти ввода/вывода из CX-Programmer в физический модуль ЦПУ, убедитесь в безопасности этой операции. Устройства, подсоединенные к модулям выходов, могут сработать непредусмотренным образом, независимо от текущего режима работы модуля ЦПУ. Перечисленные ниже операции следует выполнять с особой осторожностью.

- Изменение состояния физических входов/выходов (область CIO) модуля ЦПУ с помощью окна PLC Memory («Память ПЛК») в CX-Programmer.

**Предупреждение**

- Изменение состояния физических входов/выходов (область CIO) модуля ЦПУ путем передачи содержимого файловой памяти с помощью окна Memory Card («Карта памяти»).

Если в операнде команды указывается адрес первого или последнего слова некоторой последовательности слов, должны использоваться переменные с параметром AT (или внешние переменные), либо размер переменных должен совпадать с размером данных, обрабатываемых командой.

1. Если в этом случае будет использована переменная, не являющаяся массивом, отличающаяся по размеру и не имеющая параметра AT, CX-Programmer выдаст сообщение об ошибке при компиляции программы.

2. Характеристики переменных-массивов

- Фиксированный размер (количество данных) операнда команды: Количество элементов массива должно точно совпадать с количеством элементов данных, обрабатываемых командой. В противном случае CX-Programmer выдаст сообщение об ошибке при компиляции.

- Нефиксированный размер (количество данных) операнда команды: Количество элементов массива должно быть больше или равно размеру (количеству данных), указанному в других операндах.

- Если другой операнд, указывающий размер, является константой, CX-Programmer выдаст сообщение об ошибке при компиляции программы.

- Если другой операнд, указывающий размер, является переменной, CX-Programmer не выдаст сообщение об ошибке при компиляции, даже если размер переменной-массива будет отличаться от размера, указанного другим операндом (переменной). В этом случае, однако, отобразится сообщение с предупреждением. В частности, если количество элементов массива окажется меньше размера (количества элементов данных), указанного другим операндом (например, если размер операнда команды равен 16, а в фактической таблице переменных зарегистрировано 10 элементов данных), команда произведет чтение или запись того количества элементов данных области памяти, которое указано ее операндом. Так, в описанном выше случае будут прочитаны или записаны не только 10 элементов данных, которые зарегистрированы в фактической таблице переменных, но и 6 слов, расположенных за ними. Если эти слова используются другими командами (в том числе внутренними переменными функциональных блоков), система может работать непредсказуемым образом, что может стать причиной несчастного случая с тяжелыми последствиями.



Перед началом работы ПЛК обязательно следует убедиться в том, что наличие операндов, в которых указаны переменные с размером меньше, чем размер, указанный в определении операнда, не повлияет отрицательно на работу управляемой системы.

**Предупреждение**

Прежде чем загружать программу или изменять содержимое области памяти входов/выходов другого узла, убедитесь в безопасности данной операции для адресуемого узла. Невыполнение этого требования может стать причиной несчастного случая.

**Предупреждение**

Редактирование в режиме онлайн можно осуществлять лишь в том случае, если увеличение времени цикла не приведет к нежелательному воздействию на систему. В противном случае входные сигналы могут оказаться нечитаемыми.

-  **Предупреждение** Если используется синхронная работа модулей, редактирование в режиме онлайн можно осуществлять лишь в том случае, если увеличение времени синхронной обработки не повлияет отрицательно на работу главной и ведомой осей.
-  **Предупреждение** Прежде чем вызывать окно Ladder Section («Сегмент лестничной диаграммы») для мониторинга сигналов и текущих значений или вызывать окно Watch («Таблица мониторинга») для мониторинга текущих значений, убедитесь в достаточной безопасности этих операций. Принудительная установка/сброс состояний или вызов/отмена операций в результате случайного нажатия клавиш быстрого вызова команд могут привести к непредусмотренному срабатыванию устройств, подсоединенных к модулям выходов, независимо от текущего режима работы модуля ЦПУ.

4 Меры предосторожности при эксплуатации

При работе с программным обеспечением CX-Programmer должны соблюдаться следующие меры предосторожности.

- Программы пользователя невозможно считывать в CX-Programmer.
- Перед запуском CX-Programmer должны быть соблюдены следующие меры предосторожности.
 - Прекратите работу всех программ, не связанных с CX-Programmer. В частности, не должна работать ни одна программа, запускающаяся периодически или автоматически, например экранные заставки, антивирусные программы, почтовые программы, любые другие программы связи, планировщики и т. п.
 - Запретите доступ к жестким дискам, принтерам или другим устройствам для других компьютеров по какой-либо сети.
 - В некоторых переносных компьютерах (ноутбуках и т. п.) порт RS-232C по умолчанию отведен под модем или инфракрасный порт. Следуя инструкциям, содержащимся в документации на ваш компьютер, сконфигурируйте порт RS-232C в качестве обычного последовательного порта.
 - В некоторых переносных компьютерах (ноутбуках и т. п.) функция энергосбережения по умолчанию настроена таким образом, что на порт RS-232C не подается номинальное питание. Одновременно могут действовать параметры собственной функции энергосбережения Windows, параметры специальных служебных программ и параметры в BIOS. Следуя инструкциям, содержащимся в документации на ваш компьютер, отключите все параметры энергосбережения.
- Не отключайте питание ПЛК и не отсоединяйте кабель связи, пока между ПЛК и CX-Programmer установлена связь (режим «онлайн»). В работе компьютера может произойти сбой.
- Убедитесь в том, что ни одна из следующих операций не приведет к нежелательным последствиям для системы. Невыполнение этого требования может привести к непредсказуемой работе оборудования.
 - Изменение режима работы ПЛК.
 - Принудительная установка/сброс любого бита в памяти.
 - Изменение текущего значения любого слова или любого установленного значения в памяти.

- Проверьте правильность выполнения программы пользователя перед тем, как запустить ее на модуле в рабочем состоянии. Невыполнение этого требования может привести к непредсказуемой работе оборудования.
- При выполнении редактирования в режиме онлайн резервная копия программы пользователя и содержимого области параметров модуля ЦПУ серии CJ2, CS1-H, CJ1-H, CJ1M и CP1H сохраняется во встроенную флэш-память. В процессе резервного копирования на передней панели модуля ЦПУ светится индикатор «BKUP». Не выключайте напряжение питания модуля ЦПУ, пока светится индикатор «BKUP». Если питание будет отключено, резервная копия данных создана не будет. Чтобы текущее состояние записи во флэш-память отображалось в CX-Programmer, установите флажок *Display dialog to show PLC Memory Backup Status (Отображать окно состояния резервного копирования памяти ПЛК)* в свойствах ПЛК, после чего выберите **Windows – PLC Memory Backup Status (Windows – Состояние резервного копирования памяти ПЛК)** в меню View (Вид).
- Программы, содержащие функциональные блоки (созданные на языке лестничных диаграмм (LD) или на языке структурированного текста (ST)) могут загружаться и считываться точно так же, как стандартные программы, не имеющие функциональных блоков. В то же время, индивидуальная загрузка задач, содержащих функциональные блоки, невозможна (считывание возможно).
- Если программа пользователя, содержащая функциональные блоки и созданная с помощью CX-Programmer версии 5.0 или более поздней версии, будет загружена в модуль ЦПУ, не поддерживающий функциональные блоки (модуль ЦПУ серии CS/CJ с версией модуля 2.0 и ниже), все экземпляры функциональных блоков будут восприняты как недопустимые команды, редактирование или выполнение программы пользователя будет невозможно.
- Если входная переменная не является переменной логического типа и в параметр вводится числовое значение без какого-либо префикса (напр., 20), в функциональный блок передается не само это значение, а значение, содержащееся по данному адресу области СЮ (напр., 0020). Для ввода числового значения обязательно размещайте перед ним префикс &, # или +, -.
- Во входных параметрах могут указываться адреса, но само значение адреса не может быть передано в качестве входной переменной. (Даже если в качестве входного параметра указывается адрес, в функциональный блок все равно передается значение, размер (тип данных) которого соответствует размеру входной переменной.) Следовательно, входную переменную нельзя использовать в качестве операнда команды в функциональном блоке, если в операнде указывается первое или последнее слово последовательности слов. В CX-Programmer версии 7.0 в этом случае следует использовать входную-выходную переменную, определенную как переменная-массив (адрес первого элемента которой задается во входном параметре) и указать первый или последний элемент переменной-массива. Либо, независимо от версии CX-Programmer, использовать внутреннюю переменную с заданным параметром AT. Можно также, как вариант, указать первый или последний элемент во внутренней переменной, определенной как переменная-массив.

- Значения всех входных параметров передаются во входные или входные-выходные переменные функционального блока одновременно, до начала выполнения программы функционального блока (а не одновременно с выполнением команд программы функционального блока). Если требуется, чтобы значение параметра передавалось во входную или входную-выходную переменную непосредственно во время выполнения команды программы функционального блока, вместо входной или входной-выходной переменной следует использовать внутреннюю или внешнюю переменную. Сказанное также справедливо и в отношении записи значений выходных переменных в выходные параметры.
- Всегда используйте внутренние переменные с заданным параметром АТ в перечисленных ниже случаях.
 - Адреса, выделенные для базовых модулей ввода/вывода, специальных модулей ввода/вывода и модулей шины ЦПУ, невозможно зарегистрировать в таблице глобальных символов, и такие переменные невозможно указать в качестве внешних переменных (например, значения, заданные для глобальных переменных, могут быть нестабильными).
 - Используйте внутренние переменные, если биты вспомогательной области, не зарегистрированные предварительно в качестве внешних переменных, объявляются как глобальные символы, и при этом эти переменные не указываются в качестве внешних переменных.
 - Используйте внутренние переменные, если требуется указать адреса ПЛК, относящиеся к другому узлу в сети: например, адрес первого слова назначения на удаленном узле для команды SEND(090) или адрес первого исходного слова на удаленном узле для команды RECV(098).
 - Используйте внутренние переменные, если в операнде команды указывается первое или последнее слово последовательности слов и при этом операнд не может быть указан в качестве переменной-массива (например, не может быть указано количество элементов массива).

Часть 1:

Функциональные блоки

РАЗДЕЛ 1

Введение в функциональные блоки

Данный раздел содержит общие сведения о возможностях программного обеспечения CX-Programmer в части работы с функциональными блоками и поясняет функции, которые отсутствуют в версиях CX-Programmer, не поддерживающих функциональные блоки.

1-1	Вводная информация о функциональных блоках	4
1-1-1	Общие сведения и основные свойства	4
1-1-2	Характеристики функциональных блоков	6
1-1-3	Файлы, создаваемые в CX-Programmer версии 6.0 или выше	8
1-1-4	Меню для работы с функциональными блоками в CX-Programmer версии 5.0 (и более поздних версий)	9
1-2	Функциональные блоки	14
1-2-1	Общие сведения	14
1-2-2	Преимущества применения функциональных блоков	15
1-2-3	Структура функционального блока.	16
1-3	Переменные	22
1-3-1	Введение	22
1-3-2	Типы и атрибуты переменных.	23
1-3-3	Атрибуты переменных	24
1-3-4	Атрибуты и типы использования переменных	25
1-3-5	Внутреннее распределение переменных по адресам	26
1-4	Преобразование определений функциональных блоков в библиотечные файлы.	28
1-5	Порядок использования	28
1-5-1	Создание функциональных блоков и выполнение экземпляров	28
1-5-2	Повторное использование функциональных блоков.	29
1-6	Информация об изменениях в новых версиях	30

1-1 Вводная информация о функциональных блоках

1-1-1 Общие сведения и основные свойства

Программное обеспечение CX-Programmer, начиная с версии 5.0, поддерживает возможность использования функциональных блоков, описанных стандартом МЭК 61131-3. Функции для работы с функциональными блоками, предусмотренные в CX-Programmer, поддерживаются модулями ЦПУ CJ2, модулями ЦПУ CP1H, контроллерами серии NSJ и контроллерами движения FQM1, а также модулями ЦПУ серии CS/CJ с версией модуля не ниже 3.0. Эти функции открывают следующие возможности для разработчиков прикладных программ.

- Программируя некоторые технологические процессы, пользователь может представлять их в виде схемы, состоящей из функциональных блоков.
- Программы (алгоритмы) функциональных блоков могут создаваться на языке релейно-контактных схем (LD) или на языке структурированного текста (ST) (см. примеч.).
 - При программировании на языке релейно-контактных схем могут использоваться (с помощью операций копирования и вставки) старые программы, созданные в CX-Programmer версии 4.0 и более ранних версий.
 - На языке структурированного текста намного проще программировать различные математические операции, для которых использование операторов языка LD затруднено.

Примечание. Язык структурированного текста (ST) — это язык высокого уровня, предназначенный для промышленных средств автоматизации (преимущественно, для ПЛК) и описанный в стандарте МЭК 61131-3. Язык структурированного текста, поддерживаемый в CX-Programmer, соответствует стандарту МЭК 61131-3.

- Переменные не требуется объявлять в тексте программы, что значительно упрощает создание блоков. Достаточно зарегистрировать переменные в таблице переменных. Более того, при вводе переменных в программу на языке LD или ST переменные могут регистрироваться автоматически. В создаваемую прикладную программу также можно вводить переменные, которые уже были ранее зарегистрированы в таблице переменных.
- Отдельный функциональный блок можно преобразовать в библиотечную функцию в виде отдельного файла, что упростит его последующее многократное применение при программировании стандартных операций.
- Проверку программы можно выполнить отдельно для одного функционального блока, чтобы убедиться в его работоспособности и пригодности для применения в качестве библиотечной функции.
- Программы, содержащие функциональные блоки (написанные на языке LD или ST), можно загружать и считывать точно так же, как стандартные программы, не имеющие в своем составе функциональных блоков. В то же время, если некоторая задача содержит функциональные блоки, ее невозможно загрузить в ПЛК отдельно от других задач (считывание возможно).
- Поддержка одномерных массивов переменных упрощает обработку данных во многих случаях применения.

Примечание. Стандарт МЭК 61131 был разработан Международной электротехнической комиссией (МЭК) в качестве международного

стандарта для программируемых логических контроллеров (ПЛК). Стандарт состоит из семи отдельных частей. Языки программирования ПЛК описаны и детализированы в *Части 3. Языки программирования (МЭК 61131-3)*.

- Функциональный блок (созданный на языке LD или ST) может быть вызван из другого функционального блока (созданного на языке LD или ST). Функциональные блоки можно вкладывать друг в друга (до 8 уровней вложения), при создании функциональных блоков языки LD и ST можно комбинировать произвольным образом.

1-1-2 Характеристики функциональных блоков

Информацию о характеристиках, не нашедших отражения в следующей таблице, следует искать в руководстве *CX-Programmer — Руководство по работе (W446)*.

Параметр	Характеристики
Номер модели	CXONE-AL□□C-V4/AL□□D-V4
Установочный диск	CXONE-AL□□C-V4: CD-ROM CXONE-AL□□D-V4: DVD-ROM
Совместимые модули ЦПУ (модели ПЛК) Примечание. Функции для работы с функциональными блоками и структурированным текстом, поддерживаемые в модулях ЦПУ серии CS/CJ с версией модуля 4.0 и выше, недоступны для модулей ЦПУ серии CS/CJ с версией модуля 3.0 и ниже, а также для ПЛК серии CP, ПЛК серии NSJ и ПЛК серии FQM1. Подробную информацию см. в разделе 1-6 <i>Информация об изменениях в новых версиях</i> .	<p>Модули ЦПУ серии CS/CJ: CS1-H, CJ1-H и CJ1M версии 3.0 и выше. Тип устройства Тип ЦПУ</p> <ul style="list-style-type: none"> • CJ2H CJ2H-CPU68/67/66/65/64/68-EIP/67-EIP/66-EIP/65-EIP/64-EIP • CJ2M CJ2M-CPU11/12/13/14/15/31/32/33/34/35 • CS1G-H CS1G-CPU42H/43H/44H/45H • CS1H-H CS1H-CPU63H/64H/65H/66H/67H • CJ1G-H CJ1G-CPU42H/43H/44H/45H • CJ1H-H CJ1H-CPU65H/66H/67H/64H-R/65H-R/66H-R/67H-R • CJ1M CJ1M-CPU11/12/13/21/22/23 <p>Модули ЦПУ серии CP.</p> <ul style="list-style-type: none"> • CP1H CP1H-X/XA/Y • CP1L CP1L-M/L <p>Примечание. Если программа пользователя, содержащая функциональные блоки и созданная с помощью CX-Programmer версии 5.0 или более поздней версии, будет загружена в модуль ЦПУ, не поддерживающий функциональные блоки (модуль ЦПУ серии CS/CJ с версией модуля 2.0 и ниже), все экземпляры функциональных блоков будут восприняты как недопустимые команды, редактирование или выполнение программы пользователя будет невозможно.</p> <ul style="list-style-type: none"> • NSJ G5D (используется для NSJ5-TQ0□-G5D, NSJ5-SQ0□-G5D, NSJ8-TV0□-G5D, NSJ10-TV0□-G5D и NSJ12-TS0□-G5D) M3D (используется для NSJ5-TQ0□-M3D, NSJ5-SQ0□-M3D и NSJ8-TV0□-M3D) • FQM1-CM FQM1-CM02 • FQM1-MMA FQM1-MMA22 • FQM1-MMP FQM1-MMP22 <p>Функциональные ограничения для серий CS/CJ/CP</p> <ul style="list-style-type: none"> • В определениях функциональных блоков не поддерживаются следующие команды: команды программных блоков (BPRG и BEND), команды подпрограмм (SBS, GSBS, RET, MCRO и SBN), команды перехода (JMP, CJP и CJPN), команды пошаговых программ (STEP и SNXT), команды с мгновенным обновлением (!), ОБНОВИТЬ ВХОДЫ/ВЫХОДЫ (IORF), 1 мс ТАЙМЕР (TMNH и TMNHX) (эти таймеры могут использоваться в модулях ЦПУ CJ1-H-R). <p>Примечание. Информацию о других ограничениях см. в разделе 2-4-3 <i>Ограничения при программировании</i>.</p>

Параметр		Характеристики		
Функции, не поддерживаемые в CX-Programmer версии 4.0 и ниже.	Определение и создание функциональных блоков	Количество определенных функциональных блоков	<p>Модули CJ2H:</p> <ul style="list-style-type: none"> • CJ2H-CPU6□(-EIP): макс. 2048 на модуль ЦПУ <p>Модули ЦПУ CJ2M:</p> <ul style="list-style-type: none"> • CJ2M-CPU□1/□2/□3: макс. 256 на модуль ЦПУ • CJ2M-CPU□4/□5: макс. 2048 на модуль ЦПУ <p>Модули ЦПУ CS1-H/CJ1-H:</p> <ul style="list-style-type: none"> • Суффикс -CPU44H/45H/64H/65H/66H/67H/64H-R/65H-R/66H-R/67H-R: макс. 1024 на модуль ЦПУ • Суффикс -CPU42H/43H/63H: макс. 128 на модуль ЦПУ <p>Модули ЦПУ CJ1M:</p> <ul style="list-style-type: none"> • CJ1M-CPU11/12/13/21/22/23: макс. 128 на модуль ЦПУ <p>Модули ЦПУ CP1H:</p> <ul style="list-style-type: none"> • Все модели: макс. 128 на модуль ЦПУ <p>Модули ЦПУ CP1L:</p> <ul style="list-style-type: none"> • CP1L-M/L: макс. 128 на модуль ЦПУ <p>Контроллеры NSJ:</p> <ul style="list-style-type: none"> • NSJ□-□□□□-G5D: макс. 1024 на контроллер; • NSJ□-□□□□-M3D: макс. 128 на контроллер <p>Контроллеры движения FQM1:</p> <ul style="list-style-type: none"> • FQM1-CM002/MMA22/MMP22: макс. 128 на контроллер 	
		Имена функциональных блоков	Макс. 64 символа	
Функции, не поддерживаемые в CX-Programmer версии 4.0 и ниже.	Определение и создание функциональных блоков	Переменные	Имена переменных	Макс. 30 000 символов
			Типы переменных	Входные переменные (Inputs), выходные переменные (Outputs), входные-выходные переменные (In Out), внутренние переменные (Internals) и внешние переменные (Externals)
		Количество переменных, используемое в функциональном блоке (не включая внутренние переменные, внешние переменные, EN и EN0)	Макс. количество переменных на одно определение функционального блока	<ul style="list-style-type: none"> • Входные-выходные переменные: макс. 16 • Входные переменные + входные-выходные переменные: макс. 64 • Выходные переменные + входные-выходные переменные: макс. 64
		Назначение адресов переменным	Автоматическое назначение (пользователем может быть задан диапазон выделяемых адресов)	
		Назначение конкретного адреса	Поддерживается	
		Использование переменных-массивов	Поддерживается (только одномерные массивы и только для внутренних и входных-выходных переменных)	
Язык	Возможно создание функциональных блоков на языке релейно-контактных схем (LD) или на языке структурированного текста (ST, см. примеч.).			

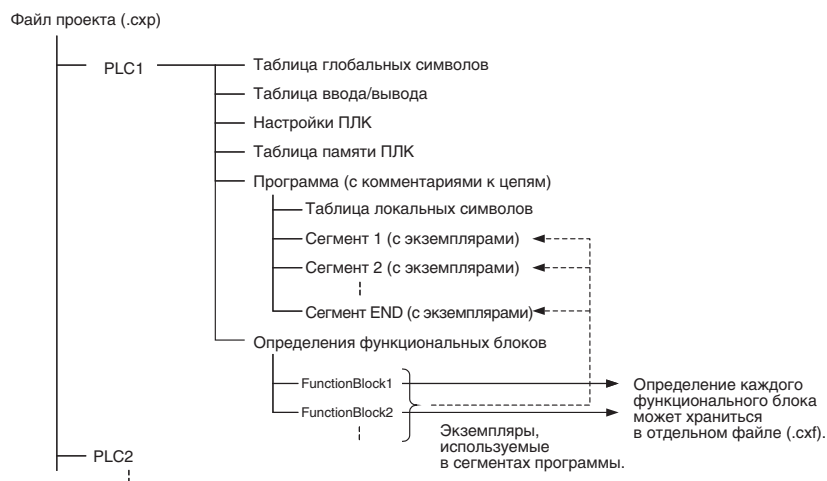
Параметр			Характеристики
Функции, не поддерживаемые в CX-Programmer версии 4.0 и ниже.	Создание экземпляров	Количество экземпляров	Модули ЦПУ CJ2H: <ul style="list-style-type: none"> • CJ2H-CPU6□(-EIP): макс. 2048 на модуль ЦПУ Модули ЦПУ CJ2M: <ul style="list-style-type: none"> • CJ2M-CPU□1/□2/□3: макс. 256 на модуль ЦПУ • CJ2M-CPU□4/□5: макс. 2048 на модуль ЦПУ Модули ЦПУ CS1-H/CJ1-H: <ul style="list-style-type: none"> • Суффикс -CPU44H/45H/64H/65H/66H/67H/64H-R/65H-R/66H-R/67H-R: макс. 2048 на модуль ЦПУ • Суффикс -CPU42H/43H/63H: макс. 256 на модуль ЦПУ Модули ЦПУ CJ1M: <ul style="list-style-type: none"> • CJ1M-CPU11/12/13/21/22/23: макс. 256 на модуль ЦПУ Модули ЦПУ CP1H: <ul style="list-style-type: none"> • Все модели: макс. 256 на модуль ЦПУ Модули ЦПУ CP1L: <ul style="list-style-type: none"> • CP1L-M/L: макс. 256 на модуль ЦПУ Контроллеры NSJ: <ul style="list-style-type: none"> • SJ□-□□□□-G5D: макс. 2048 на контроллер; • NSJ□-□□□□-M3D: макс. 256 на контроллер Контроллеры движения FQM1: <ul style="list-style-type: none"> • FQM1-CM002/MMA22/MMP22: макс. 256 на контроллер
		Имена экземпляров	Макс. 15 000 символов
	Хранение функциональных блоков в виде файлов	Файлы проекта	Файл проекта (.sxp/sxt) содержит определение и экземпляры функциональных блоков.
		Файлы программы	Файл программы в памяти файлов (*.obj) содержит определение и экземпляры функциональных блоков.
Библиотечные файлы функциональных блоков		Каждое определение функционального блока может быть сохранено в виде отдельного файла (.sxf) для последующего повторного использования в других проектах.	

Примечание. Язык структурированного текста (ST) соответствует стандарту МЭК 61131-3, однако в CX-Programmer версии 5.0 поддерживаются только следующие его элементы: выражения присваивания, выражения выбора (CASE и IF), выражения цикла (FOR, WHILE, REPEAT и EXIT), выражения RETURN, операторы арифметических и логических операций, функции сравнения, числовые функции, стандартные функции для работы со строками, функции для работы с числовыми значениями в формате строк, расширенные функции OMRON и комментарии. Подробную информацию см. в разделе **РАЗДЕЛ 5 Характеристики языка структурированного текста** в части **Часть 2: Структурированный текст (ST) настоящего руководства.**

1-1-3 Файлы, создаваемые в CX-Programmer версии 6.0 или выше

Файлы проекта (*.sxp) и файлы программ в памяти файлов (*.obj)

Создаваемые с помощью CX-Programmer проекты, содержащие определения функциональных блоков, и проекты с экземплярами функциональных блоков сохраняются в одни и те же стандартные файлы проектов (*.sxp) и файлы программ для памяти файлов (*.obj). Содержание проекта показано на следующем рисунке. Создаваемые определения функциональных блоков сохраняются в папку, которая, вместе с папкой файлов программ, находится внутри папки соответствующего ПЛК.



Файлы библиотеки функциональных блоков (*.cxf)

Определение функционального блока, созданное в проекте с использованием CX-Programmer версии 6.0, может быть сохранено в виде отдельного файла (для каждого определения создается отдельный файл) и может быть впоследствии загружено в другие проекты для повторного использования.

Примечание.

Если некоторый функциональный блок содержит вложенные функциональные блоки, библиотечный файл этого функционального блока (.cxf) также будет содержать определения всех вложенных функциональных блоков.

Текстовые файлы проекта с функциональными блоками (*.cxt)

Данные, сохраняемые программой CX-Programmer версии 6.0 в файлы проекта (*.cpr), также могут сохраняться в текстовые файлы СХТ (*.cxt).

1-1-4 Меню для работы с функциональными блоками в CX-Programmer версии 5.0 (и более поздних версий)

В приведенных далее таблицах перечислены пункты и команды меню, предусмотренные в CX-Programmer версии 5.0 и более поздних версий для работы с функциональными блоками. Подробную информацию о всех пунктах и командах меню можно найти в руководстве CX-Programmer — *Руководство по работе (W446)*.

Главное меню

Главное меню	Подменю		Быстрый вызов	Описание
File (Файл)	Function Block (Функциональный блок)	Load Function Block from File (Загрузить функциональный блок из файла)	---	Чтение сохраненного библиотечного файла функционального блока (*.cxf).
		Save Function Block to File (Сохранить функциональный блок в файл)	---	Сохранение созданного определения функционального блока в файл ([библиотечный файл функц. блока]*.cxf).

Главное меню	Подменю		Быстрый вызов	Описание
Edit (Правка)	Update Function Block (Обновить функциональный блок)		---	Если после создания экземпляра некоторого функционального блока вносятся изменения во входные, выходные или входные-выходные переменные в определении этого функционального блока, отрезок левой шины, к которому присоединен экземпляр, отображается красным цветом в знак ошибки. Данная команда актуализирует экземпляр в соответствии с произведенными изменениями и сбрасывает ошибку.
	To Lower Layer (На уровень ниже)		---	Переход к определению функционального блока для выбранного экземпляра.
	Function Block (ladder) generation (Сгенерировать функциональный блок (LD))		---	Генерация функционального блока на языке LD для выбранного сегмента программы с автоматическим определением условий назначения адресов.
View (Вид)	Monitor FB Instance (Мониторинг экземпляра функц. блока)		---	Во время мониторинга программы в режиме онлайн осуществляется мониторинг состояний переменных программы на языке ST, а также состояний битов и слов входов/выходов (I/O bit monitor) программы на языке LD в экземпляре функц. блока. (Поддерживается только в CX-Programmer версии 6.1 и выше).
	To Lower Layer (На уровень ниже)		---	Отображение с правой стороны содержимого определения функционального блока для выбранного экземпляра. (Поддерживается только в CX-Programmer версии 6.0 и выше).
	To Upper Layer (На уровень выше)		---	Возврат к вызывающему экземпляру (представлению на языке LD или ST). (Поддерживается только в CX-Programmer версии 6.0 и выше).
	Window (Окно)	FB Instance Viewer (Окно обзора экземпляров функц. блоков)	---	Отображение окна обзора экземпляров функциональных блоков. (В данном окне отображаются такие подробности, как иерархия уровней вложения экземпляров (при наличии вложений) и значения адресов, назначаемых переменным в экземплярах.)
	Function Block Invocation (Вызов функционального блока)		F	Создание экземпляра функционального блока в программе (сегменте программы) по месту расположения курсора.
Insert (Вставка)	Function Block Parameter (Параметр функционального блока)		P	Задание входного или выходного параметра переменной, когда курсор располагается слева от входной переменной или справа от выходной переменной.

Главное меню	Подменю			Быстрый вызов	Описание
PLC (ПЛК)	Memory Allocation (Распределение памяти)	Function Block/SFC Memory (Память функц. блоков/SFC)	Function Block /SFC Memory Allocation (Адреса памяти для функц. блоков/SFC)	---	Установка диапазона адресов (областей памяти для экземпляров функц. блоков), выделяемых для переменных внутри выбранных экземпляров.
			Function Block /SFC Memory Statistics (Статистика использования памяти для функц. блоков/SFC)	---	Проверка состояния адресов, выделяемых для переменных внутри выбранных экземпляров.
			Function Block Instance Address (Адрес экземпляра функционального блока)	---	Проверка адресов, выделенных для каждой переменной внутри выбранного экземпляра.
			Optimize Function Block/SFC Memory (Оптимизировать память для функц. блоков/SFC)	---	Оптимизация распределения адресов, выделяемых для переменных внутри экземпляров.
Program (Программа)	Online Edit (Редактировать онлайн)		Begin (Начать)	---	Запуск онлайн-редактирования функционального блока.
			Send Change (Передать изменения)	---	Передача изменений, произведенных во время онлайн-редактирования функционального блока.
			Cancel (Отмена)	---	Отмена изменений, произведенных во время онлайн-редактирования функционального блока.
			Transfer FB Source (Передать исходный код функционального блока)	---	Передача только исходного кода функционального блока.
			Release FB Online Edit Access Rights (Освободить право доступа к функц. блоку для онлайн-редактирования)	---	Принудительное освобождение прав доступа к функциональному блоку, SFC и ST для онлайн-редактирования, если они в данный момент удерживаются другим пользователем.

Главное меню	Подменю		Быстрый вызов	Описание
Tools (Инструменты)	Simulation (Имитация)	Break Point Set/ Clear Break Point (Точка остановки Установить/убрать точку остановки)	---	Установка или снятие точки остановки.
		Break Point Clear All Break Point (Точка остановки Убрать все точки остановки)	---	Снятие всех точек остановки.
		Mode Run (Monitor Mode) (Режим Выполнение (режим мониторинга))	---	Непрерывное циклическое выполнение программы. (Перевод эмулятора выполнения программы (LD) в режим «Мониторинг».)
		Mode Stop (Program Mode) (Режим Стоп (режим программирования))	---	Перевод эмулятора в режим «Программирование».
		Mode Pause (Режим Пауза)	---	Приостановка работы эмулятора.
		Step Run (Выполнить шаг)	---	Выполнение только одного шага программы в эмуляторе.
		Step Run Step In (Выполнить шаг Шаг с заходом)	---	Если в программе имеется команда вызова функционального блока, данная команда меню переходит к выполнению шага внутренней программы функционального блока.
		Step Run Step Out (Выполнить шаг Шаг с выходом)	---	Если в данное время выполняется шаг внутренней программы функционального блока, данная команда меню возвращает эмулятор к программе вышестоящего уровня (вызвавшей данный функциональный блок) и приостанавливает выполнение.
		Step Run Continuous Step Run (Выполнить шаг Непрерывное выполнение)	---	Непрерывное выполнение шагов в течение фиксированного промежутка времени.
		Step Run Scan Run (Выполнить шаг Один цикл)	---	Выполнение одного цикла программы и приостановка выполнения.
	Always Display Current Execution Point (Всегда отображать текущую точку выполнения)	---	Если для команды Step Run (Выполнить шаг) или Continuous Step Run (Непрерывное выполнение) выбран данный пункт меню, программа на экране автоматически пролистывается, так что в конце выполнения на экране всегда отображается точка, в которой было приостановлено выполнение программы.	
	Break Point List (Список точек остановки)	---	Вызов списка установленных точек остановки (с возможностью перехода к выбранной точке остановки).	
	Change Input mode (Изменить режим ввода)	Smart Input Mode (Режим интеллектуального ввода)	---	При вводе программы в режиме интеллектуального ввода на экране автоматически отображаются наиболее вероятные варианты команд и адресов.
Classic Mode (Классический режим)		---	Классический режим — это режим ввода, который использовался в прежней версии CX-Programmer.	

Основные всплывающие меню**Всплывающее меню для определений функциональных блоков**

Всплывающее меню		Описание
Insert Function Block (Вставить функциональный блок)	Ladder (Язык LD)	Создание определения функционального блока с программой на языке релейно-контактных схем.
	Structured Text (Язык ST)	Создание определения функционального блока с программой на языке структурированного текста.
	From file (Из файла)	Чтение определения функционального блока из библиотечного файла функционального блока (*.cxf).

Всплывающее меню для функциональных блоков, вставленных в программу

Всплывающее меню		Описание
Open (Открыть)		Отображение содержания выбранного определения функционального блока в правой части окна.
Save Function Block File (Сохранить функциональный блок в файл)		Сохранение выбранного определения функционального блока в файл.
Compile (Компилировать)		Компиляция выбранного определения функционального блока.
FB online Edit (Редактировать функц. блок онлайн)	Begin (Начать)	Запуск онлайн-редактирования функционального блока.
	Send Change (Передать изменения)	Передача изменений, произведенных во время онлайн-редактирования функционального блока.
	Cancel (Отмена)	Отмена изменений, произведенных во время онлайн-редактирования функционального блока.
	Transfer FB Source (Передать исходный код функционального блока)	Передача только исходного кода функционального блока.
	Release FB Online Edit Access Rights (Освободить право доступа к функц. блоку для онлайн-редактирования)	Принудительное освобождение прав доступа к функциональному блоку для онлайн-редактирования, если они в данный момент удерживаются другим пользователем.

Всплывающее меню для таблиц переменных функционального блока

Всплывающее меню		Описание
Edit (Правка)		Внесение изменений в переменную.
Insert Variable (Вставить переменную)		Добавление переменной в последнюю строку таблицы.
Insert Variable (Вставить переменную)	Above (Над курсором)	Вставка переменной в позицию, расположенную выше текущей позиции курсора.
	Below (Под курсором)	Вставка переменной в позицию, расположенную ниже текущей позиции курсора.
Cut (Вырезать)		Удаление переменной с размещением в буфере обмена.
Copy (Копировать)		Копирование переменной.
Paste (Вставить)		Вставка переменной из буфера обмена.
Find (Найти)		Поиск переменной. Поиск переменной можно производить по имени, комментарию или всему содержимому (текстовые строки).
Replace (Заменить)		Замена переменной.
Delete (Удалить)		Удаление переменной.
Rename (Переименовать)		Изменение имени переменной (только имени).

Всплывающее меню для экземпляров

Всплывающее меню	Описание
Edit (Правка)	Изменение имени экземпляра.
Update Invocation (Обновить вызов)	Если после создания экземпляра некоторого функционального блока вносятся изменения во входные, выходные или входные-выходные переменные в определении этого функционального блока, отрезок левой шины, к которому присоединен экземпляр, отображается красным цветом в знак ошибки. Данная команда актуализирует экземпляр в соответствии с произведенными изменениями и сбрасывает ошибку.
Monitor FB Ladder Instance (Мониторинг LD экземпляра функц. блока)	Во время мониторинга программы в режиме онлайн осуществляется мониторинг состояний битов и слов входов/выходов (I/O bit monitor) программы на языке LD в экземпляре функц. блока. (Поддерживается только в CX-Programmer версии 6,0 и выше).
Monitor FB Instance (Мониторинг экземпляра функц. блока)	Во время мониторинга программы в режиме онлайн осуществляется мониторинг состояний переменных программы на языке ST, а также состояний битов и слов входов/выходов (I/O bit monitor) программы на языке LD в экземпляре функц. блока. (Поддерживается только в CX-Programmer версии 6.1 и выше).
Register in Watch Window (Зарегистрировать в окне таблицы мониторинга)	Вызов диалогового окна <i>FB variables registration (Регистрация переменных функц. блока)</i> с целью регистрации переменной, содержащейся в выбранном экземпляре, в окне <i>Watch (Таблица мониторинга)</i> .
Function Block Definition (Определение функционального блока)	Отображение определения функционального блока для выбранного экземпляра в правой части окна.

Сочетания клавиш для быстрого вызова команд

Клавиша F: вставка определений функциональных блоков в программу

Находясь в окне Ladder Section («Сегмент лестничной диаграммы»), установите курсор в позиции, в которой требуется создать экземпляр функционального блока, и нажмите клавишу **F**. Данная операция эквивалентна выбору команды меню *Insert – Function Block Invocation (Вставка – Вызов функционального блока)*.

Клавиша Enter: ввод параметров

Установите курсор в позицию слева от входной или входной-выходной переменной либо справа от выходной переменной, после чего нажмите клавишу **Enter**. Данная операция эквивалентна выбору команды меню *Insert – Function Block Parameter (Вставка – Параметр функционального блока)*.

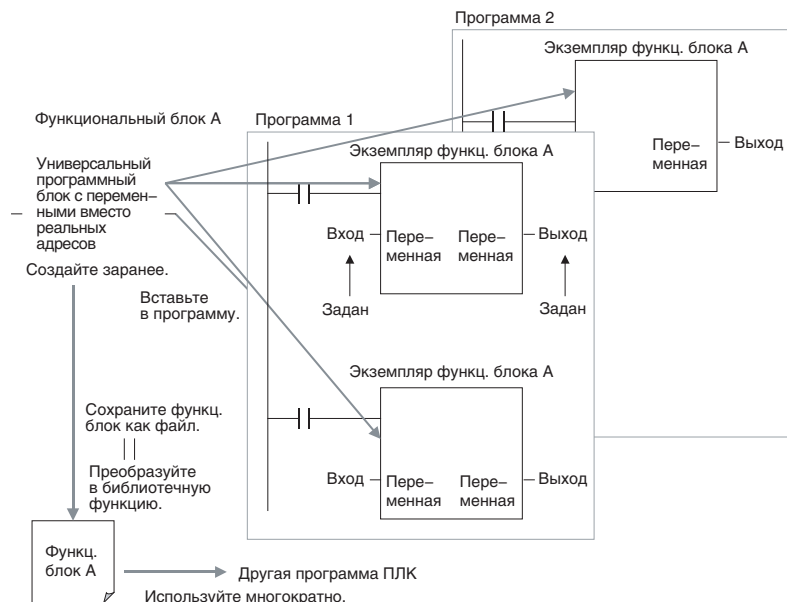
1-2 Функциональные блоки

1-2-1 Общие сведения

Функциональный блок — заранее определенный элемент основной программы, который содержит программу некоторой стандартной функции вычисления или обработки данных. Создав один раз функциональный блок, пользователь затем может просто вставлять его в свою программу, указывая требуемые входные и выходные данные для каждого вставляемого экземпляра функционального блока.

Поскольку функциональный блок заключает в себе некоторую стандартную вычислительную функцию, при его создании используются не конкретные физические адреса памяти, а переменные. Для применения функционального блока в программе пользователь указывает для этих переменных адреса или константы. Эти адреса или константы называются параметрами. Адреса, используемые самими переменными, автоматически назначаются программой CX-Programmer для каждого экземпляра функционального блока.

В CX-Programmer предусмотрена возможность сохранения отдельного функционального блока в отдельный файл, который затем можно использовать повторно в других программах ПЛК. Таким образом, имеется возможность создания библиотеки стандартных функций.



1-2-2 Преимущества применения функциональных блоков

Сложные программные модули, оформленные в виде функциональных блоков, могут быть легко использованы повторно в пределах той же программы или при создании новых программ. Оформив стандартную программу как функциональный блок и сохранив этот блок в отдельный файл, пользователь затем может в любое время использовать эту программу, просто вставив экземпляр функционального блока в основную программу и задав для него входные и выходные параметры. Возможность повторного применения созданных ранее функциональных блоков значительно сокращает время программирования и отладки, снижает количество ошибок в программе и делает программы более наглядными и понятными.

Структурированное программирование

Структурированные программы, создаваемые с применением функциональных блоков, обладают более совершенной конструкцией и требуют меньше времени на разработку.

Наглядная структура программы

При вводе или чтении программы пользователю не требуется дополнительно тратить время на изучение внутреннего алгоритма работы функционального блока. Он обращается с функциональным блоком как с «черным ящиком», входные и выходные операнды которого отображаются в программе в виде имен переменных.

Применение одного функционального блока для разных процессов

Указывая в качестве параметров разные входные данные (например, уставки таймеров, константы управления, параметры скорости или расстояния), один и тот же функциональный блок, реализующий некоторую стандартную функцию обработки, можно использовать для процессов разной природы.

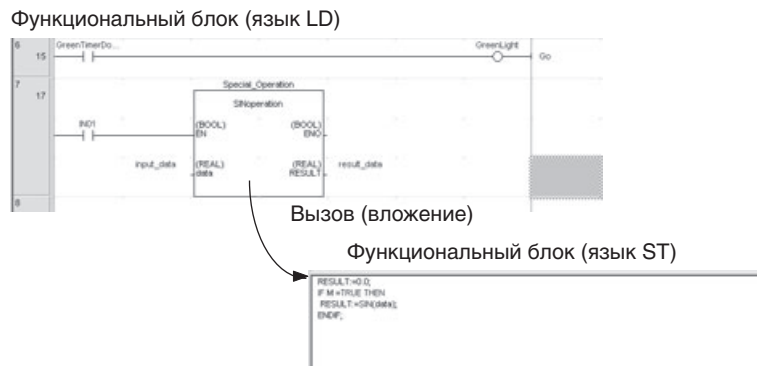
Сокращение ошибок в программе

Возможность повторного использования уже отлаженных блоков способствует уменьшению количества ошибок в программе.

- Защита ноу-хау** Функциональный блок можно закрыть для чтения, чтобы сохранить содержание его программы в секрете.
- Защита данных** Переменные функционального блока недоступны для прямого обращения извне, что позволяет защитить данные (от случайного изменения и т. п.).
- Удобство многократного использования благодаря переменным** Благодаря тому что входы и выходы функционального блока вводятся как переменные, при каждом использовании функционального блока не требуется менять адреса данных (как это происходит в случае обычного копирования и повторного использования фрагментов программы).

Создание библиотек Функционально обособленные и при этом часто используемые процессы и операции (например, процессы и операции отдельных технологических этапов, механизмов, оборудования или систем управления) могут быть сохранены в виде определений функциональных блоков и преобразованы в библиотечные функции.
 Поскольку при создании функциональных блоков указываются не физические адреса памяти, а имена переменных, можно очень легко разрабатывать новые программы, просто считывая определения функциональных блоков из файла и размещая их в новой программе.

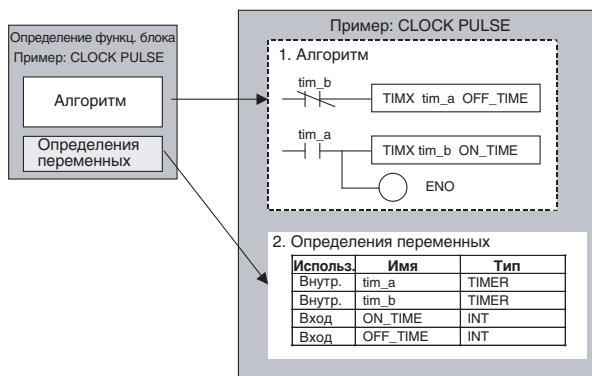
Поддержка вложений и программирование на разных языках Для программирования математических выражений можно использовать язык структурированного текста (ST).
 В CX-Programmer версии 6.0 и более поздних версий допускается вложение функциональных блоков. Например, основной функциональный блок может создаваться на языке релейно-контактных схем, а для выполнения сложных математических действий в нем могут вызываться функциональные блоки, написанные на языке структурированного текста.



1-2-3 Структура функционального блока

С понятием «функциональный блок» связаны два основных понятия: определение функционального блока и экземпляр функционального блока. Каждое из этих понятий подробно рассмотрено ниже.

Определение функционального блока Определение функционального блока — это программа, содержащаяся в функциональном блоке. Определение функционального блока, в свою очередь, состоит из собственно программы (далее по тексту также будет использоваться термин «алгоритм») и определений переменных. Пример определения функционального блока показан на рисунке ниже.



1. Алгоритм

Программа, универсальность которой обеспечивается за счет того, что вместо конкретных адресов памяти ввода/вывода используются абстрактные имена переменных. При создании программы функционального блока в CX-Programmer может использоваться язык релейно-контактных схем или язык структурированного текста.

2. Определения переменных

Таблица переменных содержит список всех используемых переменных функционального блока (входных, выходных, входных-выходных или внутренних) вместе с их свойствами (тип данных и т. п.). Подробные сведения смотрите в разделе 1-3 *Переменные*.

Максимальное количество определений функциональных блоков, которое может быть создано для одного модуля ЦПУ, варьируется от 128 до 1024 и зависит от модели используемого модуля ЦПУ.

Количество определений функциональных блоков

Экземпляры

В программу вставляется не само определение функционального блока, а его копия в виде условного обозначения. Копия определения функционального блока, вставляемая в программу, называется «экземпляром» или «экземпляром функционального блока». Каждому экземпляру присваивается некоторое имя, идентифицирующее этот экземпляр в программе.

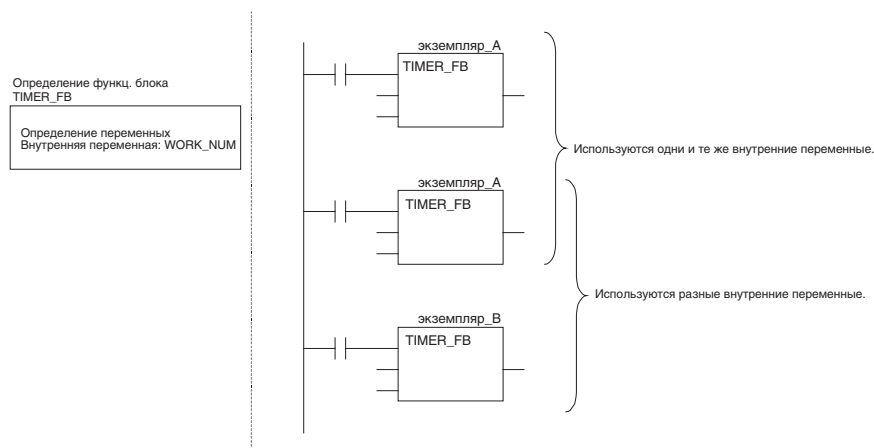
Тиражируя экземпляры некоторого функционального блока, разработчик программы может использовать одну и ту же функцию для обработки разных входных и выходных данных.



Примечание. Для управления экземплярами используются имена. Допускается включать в программу несколько экземпляров с одним и тем же именем. Если несколько (два или больше) экземпляров имеют одинаковое имя, они также используют одни и те же внутренние переменные. Напротив, экземпляры с отличающимися именами используют разные внутренние переменные.

Допустим, к примеру, что имеется несколько экземпляров некоторого функционального блока, в котором в качестве внутренней переменной используется таймер. В этом случае имена всех экземпляров должны отличаться. Если имя какого-нибудь из экземпляров совпадет с именем другого экземпляра, оба этих экземпляра будут использовать один и тот же таймер, то есть один и тот же таймер будет использоваться одновременно разными сегментами программы.

С другой стороны, если внутренние переменные не используются или используются только временно и при следующем выполнении экземпляров инициализируются, использование экземпляров с одинаковыми именами вполне допустимо и способствует более экономному расходованию памяти.

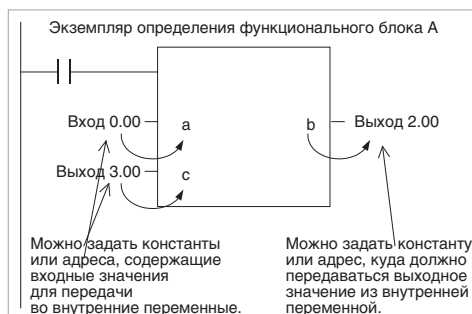


Количество экземпляров

В программе может быть создано множество экземпляров одного и того же определения функционального блока. Программа одного модуля ЦПУ, в зависимости от его модели, может содержать максимум 256 или 2048 экземпляров. Допустимое количество экземпляров никак не связано с количеством определений функциональных блоков и количеством задач, в которые вставляются экземпляры.

Параметры

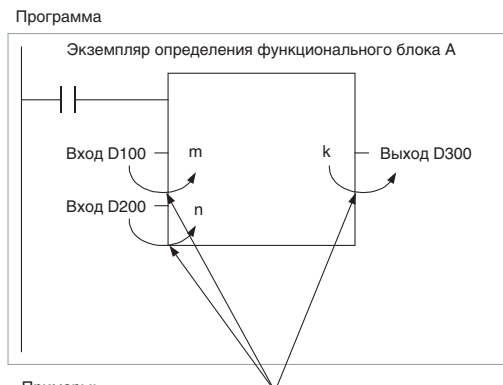
Каждый раз, когда экземпляр функционального блока вставляется в программу, для его входных, выходных и входных-выходных переменных должны задаваться фактические адреса памяти ввода/вывода или константы. Это делается с целью передачи входных данных экземпляру и получения выходных данных от экземпляра. Эти адреса и константы называются параметрами.



Использование входных и выходных переменных

Адрес, указываемый для входной переменной, в общем случае является не адресом какого-то одного элемента данных, а опорным адресом совокупности данных, форма и размер которой определяются типом данных входной переменной, принимающей эти данные. Аналогичным образом, адрес, указываемый для выходной переменной, в общем случае является опорным адресом некоторой области памяти, куда записывается совокупность данных, форма и размер которой определяются типом данных выходной переменной, отдающей эти данные.

Другими словами, даже если в качестве входного или выходного параметра будет указан адрес слова данных, будет передаваться не одно слово данных, а такое количество слов, которое соответствует типу данных входной или выходной переменной. Указанный адрес будет являться адресом первого слова передаваемой совокупности слов.



Примеры:
 Если m — переменная типа WORD, в нее передается содержимое одного слова (D100).
 Если n — переменная типа DWORD, в нее передается содержимое двойного слова (D200, D201).
 Если k — переменная типа LWORD, в нее передается содержимое четверного слова (D300...D303).

Примечание

- (1) В качестве параметров функциональных блоков могут использоваться только адреса следующих областей памяти: область СЮ, вспомогательная область, область DM, область EM (банки 0...С), область хранения и рабочая область. Следующие области и адреса использовать невозможно: регистры указателей и регистры данных (как с прямым, так и с косвенным указанием), а также косвенные адреса области DM и области EM (как в двоичном, так и в двоично-десятичном формате).
- (2) В качестве параметров функциональных блоков также могут указываться локальные и глобальные символьные имена программы пользователя. Но это возможно лишь в том случае, если

размер данных локального или глобального символьного имени совпадает с размером данных переменной функционального блока.

- (3) Входные значения передаются из параметров во входные переменные до выполнения внутренней программы функционального блока. Напротив, выходные значения передаются из выходных переменных в параметры после выполнения программы функционального блока. Если требуется прочитать или записать значение непосредственно во время выполнения программы функционального блока, не следует записывать это значение в параметр или считывать его из параметра. Следует присвоить значение внутренней переменной и использовать параметр AT (для принудительного назначения адреса).

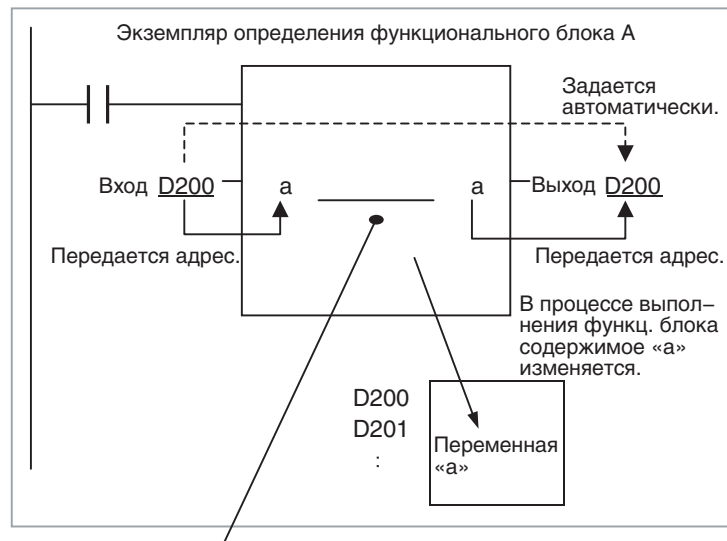
Предупреждение Если во входной переменной указывается адрес, во входную переменную передаются значения, содержащиеся по этому адресу. Сам адрес передан быть не может.

Предупреждение Параметры не могут использоваться для чтения или записи значений во время выполнения программы функционального блока. Для этих целей следует использовать внутренние переменные с заданным параметром AT (т. е. переменные с принудительно назначенными адресами). Другим вариантом является использование глобального символа в качестве внешней переменной.

Использование входных-выходных переменных

При использовании входной-выходной переменной во входном параметре может быть указан только адрес. Константу в этом случае указать невозможно. Значение адреса, заданное во входном параметре, передается в функциональный блок. Если во внутренних вычислениях функционального блока используется входная-выходная переменная, результаты этих вычислений, в объеме, который определяется размером переменной, записываются в область памяти, начиная с адреса, который указан во входном параметре.

Программа



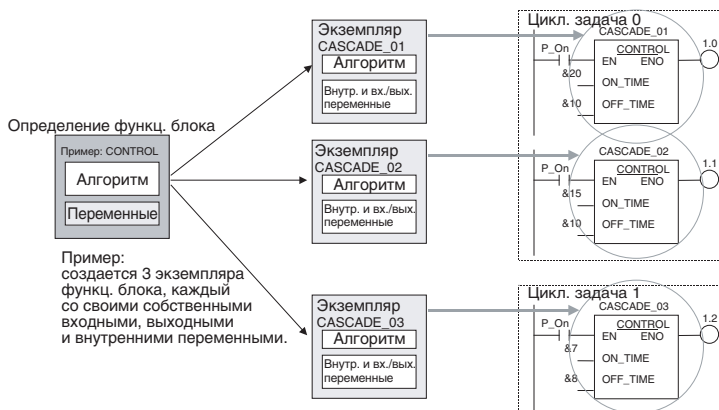
Во внутренних операциях функционального блока используется переменная «а». Полученное в итоге значение записывается в память ввода/вывода, начиная с адреса D200. Количество записываемых слов определяется размером (т. е. типом данных) переменной «а».

Примечание. Для того чтобы определить переменную в качестве «входной-выходной», в таблице переменных (в CX-Programmer) в графе Usage («Использование») следует выбрать «In Out» (Вх.-Вых.).

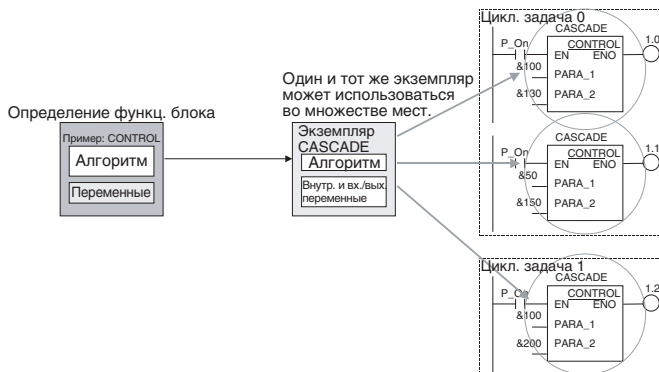
■ Справочная информация

Используя входные переменные как своего рода технологические параметры (например, константы) и в каждом экземпляре задавая для этих параметров разные значения, с помощью всего одного функционального блока можно легко реализовать множество различных процессов.

Пример: создание трех экземпляров одного определения функционального блока



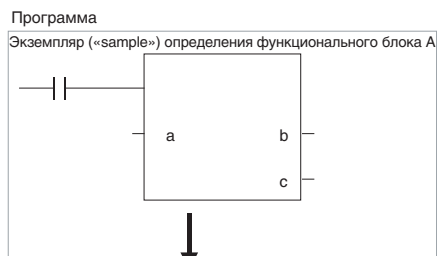
В следующих случаях допускается использовать одно и то же имя экземпляра в разных местах программы: если внутренние переменные не используются, если это не повлияет на выполнение программы или если внутренние переменные используются в других местах.



Однако при использовании одной и той же области памяти требуется соблюдать некоторые меры предосторожности. Например, если в нескольких местах одной программы используется один и тот же экземпляр, содержащий команду таймера, номер таймера также будет один и тот же в каждом из этих экземпляров, поэтому при одновременном выполнении команд этот таймер будет работать неправильно.

Регистрация экземпляров

Каждое имя экземпляра регистрируется как имя файла в таблице глобальных символов.



Экземпляр регистрируется в таблице глобальных символов, в качестве имени символа используется имя экземпляра.

Имя	Тип данных	Адрес/ значение	
sample	FB [FunctionBlock1]	нет [авто]	

Имя экземпляра ↑
Имя определения функц. блока регистрируется в квадратных скобках [] после FB.

1-3 Переменные

1-3-1 Введение

В функциональном блоке для ссылки на обрабатываемые данные вводятся не какие-то конкретные адреса памяти ввода/вывода, а имена переменных. CX-Programmer автоматически выделяет адреса для переменных в обозначенных областях памяти ввода/вывода уже при вставке экземпляра функционального блока в программу. Следовательно, пользователю нет необходимости знать настоящие значения адресов, используемые в функциональном блоке, аналогично тому, как пользователю ЭВМ не требуется знать, по каким именно адресам располагаются те или иные данные в оперативной памяти. Используя вместо адресов переменные, функциональный блок в этом смысле отличается от обычной подпрограммы и предстает перед пользователем как полностью готовый к применению компонент, внутреннее устройство которого пользователю знать необязательно.

Пример:



Примечание. Константы в качестве переменных не регистрируются. Они вводятся непосредственно как операнды команды.

- Язык релейно-контактных схем: вводится шестнадцатеричное числовое значение с префиксом # или десятичное значение с префиксом &.
- Язык структурированного текста: вводится шестнадцатеричное числовое значение с префиксом 16# или десятичное числовое значение без какого-либо префикса.

Исключение: напрямую или косвенно указываемые адреса для регистров указателей IR0...IR15 и регистров данных DR0...DR15 следует вводить непосредственно в операнды команды.

1-3-2 Типы и атрибуты переменных

Типы переменных

По способу использования различают переменные следующих типов.

Внутренние: Внутренние переменные, как следует из их названия, используются только внутри экземпляра функционального блока. Эти переменные не имеют прямой связи со входными и выходными параметрами и не предназначены для обмена данными с ними.

Входные: Входные переменные могут принимать данные от входных параметров за пределами экземпляра. По умолчанию входной переменной является переменная EN («разрешение»), служащая для ввода состояния входного условия.

Выходные: Выходные переменные могут выдавать данные в выходные параметры за пределами экземпляра. По умолчанию в качестве выходной переменной используется переменная ENO («выходное разрешение»), передающая состояние выполнения экземпляра.

Входные-выходные: Входные-выходные переменные могут принимать данные от входных параметров за пределами экземпляра и могут выдавать результаты выполнения экземпляра функционального блока во внешние параметры.

Внешние: Внешние переменные — это либо системные переменные, заранее зарегистрированные с помощью CX-Programmer, такие как флаги условий и некоторые из битов вспомогательной области, либо определенные пользователем глобальные символы, предназначенные для использования в пределах экземпляров.

Более детальную информацию об использовании переменных можно найти в строке *Определения переменных* таблицы в разделе 2-1-2 *Элементы функциональных блоков*.

Далее будет приведена таблица, указывающая возможное количество используемых переменных и вид переменной, создаваемой по умолчанию для каждого из типов использования переменной.

1-3-3 Атрибуты переменных

Переменные обладают следующими атрибутами (свойствами).

Имя переменной

Имя переменной служит для идентификации этой переменной в функциональном блоке. В разных функциональных блоках может использоваться одно и то же имя переменной, в этом нет ничего страшного.

Примечание.

Имя переменной может быть длиной до 30 000 символов, но ни в коем случае не должно начинаться с числа. Кроме того, имя не должно содержать два символа подчеркивания, следующих подряд друг за другом. В качестве имен также не должны использоваться обозначения регистров-указателей, такие как «IR0», «IR1»...«IR15». Дополнительная информация о других ограничениях приводится в пункте *Определения переменных* в разделе 2-1-2 *Элементы функциональных блоков*.

Тип данных

Для переменной может быть выбран один из следующих типов данных: BOOL, INT, UINT, DINT, UDINT, LINT, ULINT, WORD, DWORD, LWORD, REAL, LREAL, TIMER, COUNTER и STRING

Дополнительная информация о типах данных переменных содержится в пункте *Определения переменных* в разделе 2-1-2 *Элементы функциональных блоков*.

Параметр AT (назначение конкретных адресов)

Вместо автоматического назначения адреса системой для переменной может быть задан некоторый определенный адрес памяти ввода/вывода. Для этих целей предусмотрен параметр AT, в который пользователь может ввести требуемый адрес памяти ввода/вывода. Данный параметр может быть задан только для внутренних переменных. Даже если для переменной задается конкретный адрес, в программе функционального блока все равно должно использоваться имя переменной.

Дополнительную информацию о настройке параметра AT можно найти в пункте *Определения переменных* в разделе 2-1-2 *Элементы функциональных блоков*, а подробное описание использования параметра AT содержится в разделе 2-5-3 *Использование параметра AT для внутренних переменных*.

Параметры массива

Переменная может быть определена как одиночный массив значений (данных) с одинаковыми свойствами. Для преобразования переменной в массив надо просто указать, что переменная является массивом, и указать максимальное количество элементов этого массива.

Данный параметр может быть задан только для внутренних и входных-выходных переменных. В программном обеспечении CX-Programmer версии 5.0 и более поздних версий поддерживаются только одномерные массивы.

- Порядок настройки
Щелкните кнопку **Advanced (Дополнительно)**, выберите пункт *Array Variable (Переменная-массив)* и введите максимальное количество элементов.
- При вводе имени переменной-массива в программу определения функционального блока после количества переменных следует также вводить номер индекса массива в квадратных скобках.

Дополнительную информацию о настройке массивов можно найти в пункте *Определения переменных* в разделе 2-1-2 *Элементы функциональных блоков*.

Исходное значение

Исходное значение, устанавливаемое до первого случая выполнения экземпляра. В ходе выполнения экземпляра это значение может измениться.

Например, для переменной логического типа (BOOL) можно задать либо значение «1» (Истина), либо значение «0» (Ложь). Для переменной типа WORD может быть задано значение от 0 до 65 535 (от 0000 до FFFF hex). Если начальное значение не задается, в переменную записывается 0. Например, переменная логического типа будет содержать «0» (Ложь), а переменная типа WORD будет содержать 0000 hex.

Сохранение

Если требуется, чтобы при включении ПЛК и в начале выполнения программы ПЛК в переменной сохранялось ее прежнее значение, следует выбрать опцию *Retain (Сохранение)*.

- Порядок настройки
Выберите опцию *Retain (Сохранение)*.

Размер

При использовании переменной типа STRING размер, необходимый для сохранения текстовой строки, может быть задан в пределах от 1 до 255 символов.

1-3-4 Атрибуты и типы использования переменных

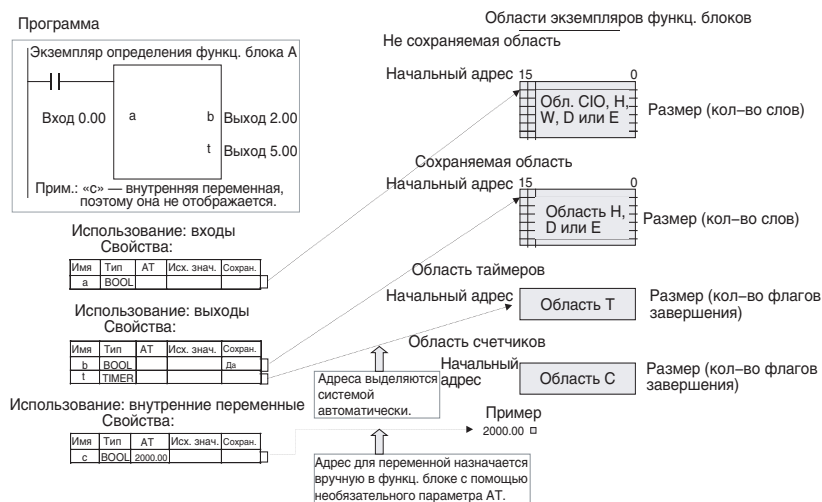
В следующей таблице показано, какие атрибуты переменных должны быть заданы, могут быть заданы или не могут быть заданы в зависимости от типа использования переменной.

Атрибут	Тип использования переменной			
	Внутренняя	Входная	Выходная	Входная-выходная
Имя	Обязательно	Обязательно	Обязательно	Обязательно
Тип данных	Обязательно	Обязательно	Обязательно	Обязательно
АТ (конкретный адрес)	Возможно	Невозможно	Невозможно	Невозможно
Определение массива	Обязательно	Невозможно	Невозможно	Обязательно
Исходное значение	Возможно	Невозможно (см. прим. 1)	Возможно	Невозможно
Сохранение	Возможно	Невозможно (см. прим. 1)	Возможно	Невозможно
Размер	Возможно (см. прим. 2)	Невозможно	Невозможно	Невозможно

- (1) Передается значение входного параметра.
- (2) Действительно только для строковых переменных (STRING).

1-3-5 Внутреннее распределение переменных по адресам

При вставке экземпляра функционального блока в программу CX-Programmer автоматически (без участия пользователя) назначает каждой переменной экземпляра конкретный адрес памяти. Адреса назначаются всем переменным, зарегистрированным в определении функционального блока, за исключением тех переменных, которым с помощью параметра AT были назначены фиксированные адреса.



Назначение областей памяти для использования переменными

Пользователь должен указать, какая именно область памяти будет использоваться системой для автоматического назначения адресов переменным экземпляра функционального блока. Для переменных будут выделяться адреса в той области памяти, которая была указана пользователем.

Порядок настройки

Выберите **Function Block/SFC Memory – Function Block/SFC Memory Allocation (Память функц. блоков/SFC – Адреса памяти для функц. блоков/SFC)** в меню PLC (ПЛК). Укажите требуемые области памяти в диалоговом окне Block/SFC Memory Allocation Dialog (Адреса памяти для функц. блоков/SFC).

Области памяти для экземпляров функц. блоков

Модули ЦПУ серии CJ2

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	H512	H1407	896	CIO, WR, HR, DM, EM (см. примеч.)
Сохран.	H1408	H1535	128	HR, DM, EM (см. примеч.)
Таймеры	T3072	T4095	1024	TIM
Счетчики	C3072	C4095	1024	CNT

Примечание. Принудительная установка/сброс возможны, если указаны следующие банки памяти EM:

CJ2H-CPU64(-EIP)/-CPU65(-EIP)	EM, банк 3
CJ2H-CPU66(-EIP)	EM, банки 6...9
CJ2H-CPU67(-EIP)	EM, банки 7...E
CJ2H-CPU68(-EIP)	EM, банки 11...18

Модули ЦПУ серии CS/CJ версии 3.0 или выше и контроллеры NSJ

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	H512	H1407	896	CIO, WR, HR, DM, EM
Сохран.	H1408	H1535	128	HR, DM, EM
Таймеры	T3072	T4095	1024	TIM
Счетчики	C3072	C4095	1024	CNT

Контроллеры движения FQM1

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	5000	5999	1000	CIO, WR, DM
Сохран.	Нет			
Таймеры	T206	T255	50	TIM
Счетчики	C206	C255	50	CNT

Модули ЦПУ серии CP

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	H512	H1407	896	CIO, WR, HR, DM (см. примеч.)
Сохран.	H1408	H1535	128	HR, DM (см. примеч.)
Таймеры	T3072	T4095	1024	TIM
Счетчики	C3072	C4095	1024	CNT

Примечание.

Область DM в ЦПУ CP1L-L

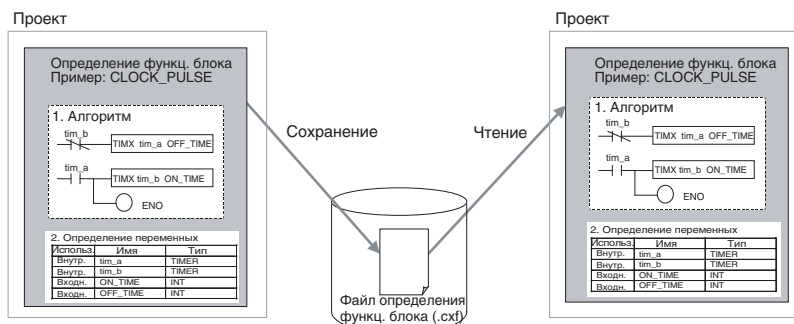
Адрес	CP1L-L
D0000...D9999	Есть
D10000...D31999	Нет
D32000...D32767	Есть

Слова области хранения для функциональных блоков (H512...H1535)

Для функциональных блоков выделяются слова области хранения в диапазоне от H512 до H1535. Эти слова отличаются от стандартных слов области хранения, используемых для программ (H000...H511), и предназначены только для области экземпляров функциональных блоков (адреса которой распределяются между переменными автоматически). Эти слова нельзя указывать в качестве операндов команд. Если эти слова будут использованы в любом другом месте (т. е. не в программе функционального блока), они будут отображаться красным цветом. Хотя ввод этих слов при создании функционального блока возможен, при проверке программы будет выдано сообщение об ошибке. Если в диалоговом окне Function Block Memory Allocation (Адреса памяти для функц. блоков) не выбран параметр «Сохранение», содержимое данной области после выключения питания и при запуске выполнения программы не сохраняется.

1-4 Преобразование определений функциональных блоков в библиотечные файлы

Определение функционального блока, созданное с помощью программы CX-Programmer, может быть сохранено в виде отдельного файла с расширением *.cxf. Такой файл называется файлом определения функционального блока. Впоследствии такой файл может быть повторно использован при разработке новых проектов (программ для ПЛК).



1-5 Порядок использования

Сначала создается определение функционального блока. Затем, когда требуется выполнить программу функционального блока, в программе размещается вызов экземпляра этого функционального блока. Кроме того, созданное определение функционального блока может быть сохранено в файл для повторного использования в других проектах (программах ПЛК).

1-5-1 Создание функциональных блоков и выполнение экземпляров

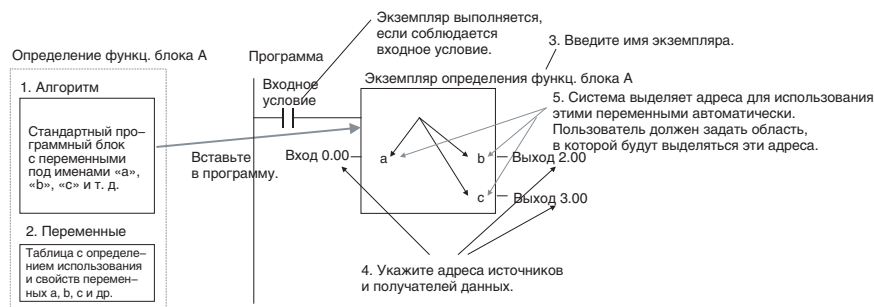
Процедура создания и выполнения функционального блока подробно описана ниже.

- 1,2,3...
1. Сначала требуется создать определение функционального блока, в том числе программу функционального блока на языке LD или ST, а также определения переменных. Можно также вставить готовый функциональный блок из созданного ранее библиотечного файла.

Примечание

- (a) В программе функционального блока должны использоваться только имена переменных.
 - (b) При написании программы на языке релейно-контактных схем допускается использовать файлы проектов, созданные в программном обеспечении CX-Programmer до версии 5.0. Для этого необходимо открыть файл проекта в программе CX-Programmer версии 5.0 или выше, скопировать требуемые части программы и вставить их в программу функционального блока.
 - (c) Уже существующая программа, написанная на языке релейно-контактных схем, может быть автоматически преобразована в функциональный блок с помощью команды **Edit – Function Block (ladder) generation (Правка – Сгенерировать функциональный блок (LD))**.
2. В создаваемую прикладную программу вставляется копия созданного определения функционального блока. На этом этапе создается экземпляр функционального блока.

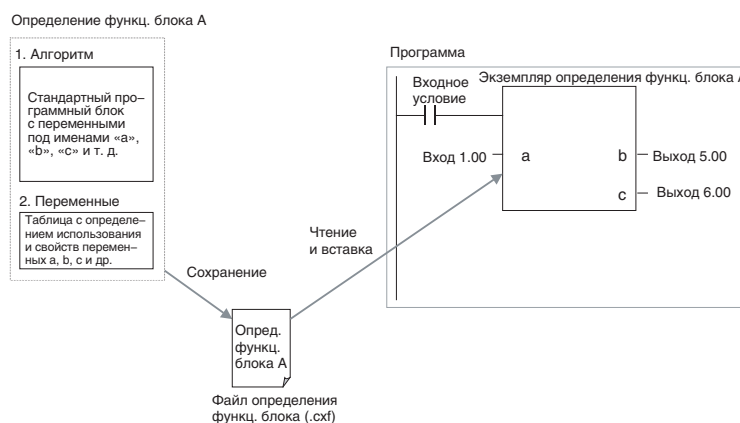
3. Для каждого экземпляра вводится имя экземпляра.
4. Задаются входные и выходные параметры экземпляра функционального блока. Для входных переменных в качестве параметров задаются адреса с входными данными и/или константы, а для выходных переменных в качестве параметров задаются адреса для записи данных и/или константы.
5. Следует выделить созданный экземпляр, вызвать команду **Function Block Memory – Function Block Memory Allocation (Память функц. блоков – Адреса памяти для функц. блоков)** в меню PLC (ПЛК) и задать используемую область памяти для каждого типа переменной.
6. Следует загрузить прикладную программу в модуль ЦПУ.
7. В ходе выполнения прикладной программы в модуле ЦПУ происходит вызов экземпляра, и экземпляр выполняется, если включено его условие выполнения.



1-5-2 Повторное использование функциональных блоков

Ниже описана процедура сохранения определения функционального блока в файл и использование его в программе другого ПЛК.

- 1,2,3...
1. Выберите функциональный блок, который требуется сохранить, и сохраните его как файл определения функционального блока (*.sxf).
 2. Откройте проект другого ПЛК и откройте/считайте ранее сохраненный файл определения функционального блока (*.sxf).
 3. Вставьте определение функционального блока в создаваемую программу нового проекта.



Примечание. В CX-Programmer, начиная с версии 5.0, для каждого определения функционального блока можно индивидуально выполнить компиляцию и проверку программы. Прежде чем сохранить тот или иной функциональный блок в отдельный файл и использовать его в другом проекте, рекомендуется выполнить компиляцию с целью проверки его программы.

1-6 Информация об изменениях в новых версиях

Сведения об обновлениях в функциях, не связанных с функциональными блоками и структурированным текстом, можно найти в руководстве *CX-Programmer — Руководство по работе* (W446).

Информация об изменениях в связи с обновлением версии: 9.2 -> 9.3

Редактор программ на языке ST

- В окне редактирования программ на языке ST добавлена индикация номеров строк. Может быть указан номер строки для непосредственного перехода к этой строке.
- Выбор функций и зарегистрированных символов возможен в списках слов (Word List).
- Нажатие клавиши табуляции при выделенной начальной функции управляющего выражения позволяет легко перейти к кадру управляющего выражения.
- Имеющиеся в программе ошибки синтаксиса выделяются волнистыми линиями красного цвета. Благодаря этому не требуется специальная проверка программы.

Интеллектуальный ввод в окне программы функц. блока на языке LD

В окне представления программы функционального блока на языке LD может использоваться режим интеллектуального ввода (точно так же, как в окне программы задачи на языке LD).

Информация об изменениях в связи с обновлением версии: 9.1 -> 9.2

Улучшения в использовании структур

Расширены возможности применения структур при создании программ для модулей ЦПУ CJ2.

- Добавлена возможность применения структур (структурных переменных, полей структурных переменных и массивов структурных переменных) в программах на языке структурированного текста.
- Структурная переменная может быть зарегистрирована и использована в качестве внешней переменной функционального блока (на языке LD или ST).

Поддержка структурных переменных: сравнение с предыдущими версиями

Использование		Версия 9.1 и ниже	Версия 9.2 и выше
Таблица глобальных символов		Да	Да
Программа на языке LD	Таблица локальных символов	Да	Да
	Окно сегмента программы на языке LD	Да	Да
Программа на языке ST	Таблица локальных символов	Нет	Да
	Редактор программ на языке ST	Нет	Да
Программа на языке SFC	Таблица локальных символов	Нет	Да
	Окно программы на языке SFC	Нет	Нет
	Окно сегмента программы на языке SFC	Нет	Нет
	Таблица символов сегмента программы на языке SFC	Нет	Да
	Окно программы на языке LD для действия	Нет	Да
	Окно программы на языке ST для действия	Нет	Да
	Окно программы на языке LD для перехода	Нет	Да
Функциональный блок на языке LD	Переменные	Внутренние переменные	Да
		Входные переменные	Нет
		Выходные переменные	Нет
		Входные-выходные переменные	Да
		Внешние переменные	Нет
Функциональный блок на языке ST	Переменные	Внутренние переменные	Нет
		Входные переменные	Нет
		Выходные переменные	Нет
		Входные-выходные переменные	Нет
		Внешние переменные	Да

Улучшения в использовании переменных типа TIMER/COUNTER

Расширены возможности применения переменных типа TIMER/COUNTER (таймер/счетчик) при создании программ для модулей ЦПУ CJ2.

- Переменные типа TIMER/COUNTER стали доступны для программ на языке ST. В программах на языке ST могут использоваться флаги завершения и текущие значения таймеров/счетчиков.
- Из программы на языке ST можно запускать и останавливать работу таймеров/счетчиков.
- Переменные типа TIMER/COUNTER можно регистрировать и использовать в качестве внешних переменных функционального блока.

Поддержка переменных типа TIMER/COUNTER: сравнение с предыдущими версиями

Использование		Версия 9.1 и ниже	Версия 9.2 и выше
Таблица глобальных символов		Да	Да
Программа на языке LD	Таблица локальных символов	Да	Да
	Окно сегмента программы на языке LD	Да	Да
Программа на языке ST	Таблица локальных символов	Да	Да
	Редактор программ на языке ST	Нет	Да
Программа на языке SFC	Таблица локальных символов	Да	Да
	Окно программы на языке SFC	Нет	Нет
	Окно сегмента программы на языке SFC	Нет	Нет
	Таблица символов сегмента программы на языке SFC	Да	Да
	Окно программы на языке LD для действия	Да	Да
	Окно программы на языке ST для действия	Нет	Да
	Окно программы на языке LD для перехода	Да	Да
	Окно программы на языке ST для перехода	Нет	Да
Функциональный блок на языке LD	Переменные	Внутренние переменные	Да
		Входные переменные	Нет
		Выходные переменные	Нет
		Входные-выходные переменные	Нет
		Внешние переменные	Нет
Функциональный блок на языке ST	Переменные	Внутренние переменные	Нет
		Входные переменные	Нет
		Выходные переменные	Нет
		Входные-выходные переменные	Нет
		Внешние переменные	Нет

Информация об изменениях в связи с обновлением версии: 9.0 -> 9.1

Добавлена поддержка новых моделей ЦПУ CJ2M-CPU□□, поддерживающих функциональные блоки и структурированный текст. Если в качестве ПЛК выбрана модель CJ2M, в окне состояния памяти можно просматривать использование области памяти программ функциональных блоков.

Информация об изменениях в связи с обновлением версии: 8.3 -> 9.0**Поддержка структур данных в качестве символьных типов данных**

Версия 8.3	Версия 9.0
Структуры данных не поддерживаются.	Модули ЦПУ CJ2 поддерживают структуры данных в качестве символьного типа данных.

Информация об изменениях в связи с обновлением версии: 8.0 -> 8.1

Добавлена поддержка новых моделей ПЛК CJ2H-CPU6□, поддерживающих функциональные блоки и структурированный текст.

Информация об изменениях в связи с обновлением версии: 7.2 -> 8.0

Добавлена поддержка новых моделей ПЛК CJ2H-CPU6□-EIP, поддерживающих функциональные блоки и структурированный текст.

Информация об изменениях в связи с обновлением версии: 7.0 -> 7.2**Улучшения в поддержке функциональных блоков и структурированного текста**

Сведения об улучшениях в других функциях CX-Programmer для данного обновления можно посмотреть в руководстве *CX-Programmer — Руководство по работе* (W446).

Улучшения в поддержке языков МЭК 61131-3

Были внесены улучшения в части поддержки языков стандарта МЭК 61131-3: ST и SFC. При создании программы можно свободно комбинировать языки LD, ST и SFC, используя наиболее подходящий язык для каждой программируемой операции или процесса. За счет этого сокращается время программирования, программы становятся более наглядными и понятными.

Поддержка языка ST при программировании задач

Версия 7.0	Версия 7.2
Язык ST мог использоваться только в функциональных блоках.	Язык ST может использоваться не только для программирования функциональных блоков, но и при написании программ задач. В одной программе пользователя могут в произвольном сочетании использоваться разные языки программирования. В частности, арифметические операции могут программироваться на языке ST, а для других операций и процессов могут использоваться языки LD или SFC. Примечание. Структурированный текст поддерживается только модулями ЦПУ серии CS/CJ с версией модуля не ниже 4.0. Он не поддерживается модулями ЦПУ серии CP.

Сравнение определений функциональных блоков и программ на языке ST

Версия 7.0	Версия 7.2
Сравнение определений функциональных блоков было невозможно.	<ul style="list-style-type: none"> • Появилась возможность сравнения функциональных блоков, благодаря которой можно легко выявить отличия между разными определениями функциональных блоков в прикладной программе. • Также возможно сравнение программ на языке ST.

Информация об изменениях в связи с обновлением версии: 6.1 -> 7.0**Удобные функции для преобразования программ на языке LD в функциональные блоки**

Версия 6.1	Версия 7.0
В создаваемое определение функционального блока может быть скопирован фрагмент существующей программы на языке LD. Однако при этом необходимо проверять символы и адреса, используемые в программе, а также определять и регистрировать вручную входные, внутренние и выходные переменные.	С помощью специальной команды меню (Сгенерировать функциональный блок (LD)) один или несколько выбранных сегментов прикладной программы могут быть автоматически преобразованы в определения функционального блока. Переменные при этом регистрируются автоматически в соответствии с символами и адресами, которые используются в данных сегментах программы (позже в них можно внести необходимые изменения). Таким образом, уже существующие программы могут быть легко преобразованы в функциональные блоки.

Редактирование функциональных блоков в режиме онлайн

Версия 6.1	Версия 7.0
Изменение определений функциональных блоков (т. е. программ и таблиц переменных) невозможно в режиме онлайн во время работы ПЛК. (Возможно изменение только входных/выходных параметров экземпляров функциональных блоков.)	<p>Программы и таблицы переменных функциональных блоков можно изменять непосредственно во время работы ПЛК (см. примеч.). Благодаря этому можно производить отладку и правку определений функциональных блоков в круглосуточно работающих системах, не допускающих прекращения работы.</p> <p>Действия: щелкнуть правой кнопкой мыши по определению функционального блока в рабочей области проекта и выбрать пункт FB Online Edit – Begin (Редактировать функц. блок онлайн – Начать) в контекстном меню.</p> <p>Примечание. Добавление новых экземпляров функциональных блоков невозможно.</p> <p>Примечание. Данная функция недоступна при имитации режима выполнения в CX-Simulator.</p>

Поддержка данных типа STRING и функций обработки данных в программах на языках ST

Версия 6.1	Версия 7.0
<ul style="list-style-type: none"> При программировании на языке ST невозможно использовать тип данных STRING (текстовые данные) (см. примеч.). При программировании на языке ST не поддерживаются функции обработки текстовых данных. Даже если используется язык LD, при выполнении команд обработки текстовых строк, команд преобразования данных и команд последовательного интерфейса с целью отображения сообщений и обмена данными в беспrotocolном режиме (см. примеч.) необходимо уделять внимание кодам ASCII и размеру текстовой строки. <p>Примечание. Ввод текстовых строк в память ввода/вывода можно осуществлять с помощью функций CX-Programmer для работы с памятью ПЛК. При этом, однако, требуется учитывать объем данных, размещаемых в памяти ввода/вывода.</p>	<ul style="list-style-type: none"> При программировании на языке ST доступен тип данных STRING (текстовые данные). Это, в частности, позволяет присваивать переменным непосредственно текстовые значения (например, a := '□READ');. При этом пользователю не нужно заботиться об эквивалентных кодах ASCII или о размере кодов. При создании программы на языке ST поддерживаются функции обработки текстовых данных, в том числе функции извлечения, конкатенации и поиска текстовых данных. Таким образом, с помощью программы на языке ST внутри функционального блока можно легко обрабатывать текстовые строки и отображать сообщения. Также поддерживаются функции передачи и приема текстовых строк. Благодаря этому с помощью языка ST в функциональном блоке можно легко реализовать обмен данными в беспrotocolном режиме, не заботясь при этом о кодах ASCII.

Поддержка входных-выходных переменных

Версия 6.1	Версия 7.0
<ul style="list-style-type: none"> Входные-выходные переменные в функциональных блоках не поддерживаются (могут использоваться только входные, внутренние и выходные переменные). Входные переменные не могут определяться как массивы. Значения входных параметров передаются во входные переменные. 	<ul style="list-style-type: none"> В функциональных блоках возможно использование входных-выходных переменных. Входные-выходные переменные могут быть определены как массивы. Во входные переменные из входных параметров передаются не значения (содержимое ячеек памяти), а адреса. Используя внутри функциональных блоков входные-выходные переменные, определенные как массивы, из входных параметров в функциональные блоки можно легко передавать большие объемы данных.

Информация об изменениях в связи с обновлением версии: 6.0 -> 6.1

Поддержка контроллеров серии NSJ В качестве модели ПЛК (в поле «тип устройства») может быть выбрана модель «NSJ», в качестве модели ЦПУ может быть выбрана модель G5D.

Поддержка контроллеров FQM1 с версией модуля 3.0 Добавлена поддержка новых моделей контроллеров движения FQM1 (модуля координирования FQM1-CM002 и модулей управления движением FQM1-MMA22/MMP22).

Функция имитации выполнения программы экземпляра на языке ST/LD

Прежняя версия (версия 6.0)	Новая версия (версия 6.1)
Программа CX-Simulator может использоваться для следующих операций: выполнение одного шага программы на языке LD (Step Run), непрерывное выполнение шагов (Continuous Step Run), выполнение одного цикла (Scan Run) и установка входных/выходных условий для точек остановки.	Операции выполнения шага, непрерывного выполнения шагов, выполнения цикла и установки/сброса точек остановки могут выполняться как функции программы CX-Programmer. Все эти функции также доступны для программ функциональных блоков, написанных на языках LD и ST. Примечание. Для использования этих функций должна быть установлена программа CX-Simulator версии 1.6 (продается отдельно). Примечание. Установка входных/выходных условий остановки невозможна.

Усовершенствованные функции для работы с функциональными блоками**Мониторинг программ функциональных блоков на языке ST**

Прежняя версия (версия 6.0)	Новая версия (версия 6.1)
При мониторинге прикладной программы в режиме онлайн мониторинг работы программы на языке ST внутри экземпляра функционального блока невозможен. (Можно было проверить содержимое программы определения функционального блока и осуществлять мониторинг входных/выходных состояний программы экземпляра функционального блока, написанной на языке LD.)	Во время мониторинга основной программы также можно осуществлять мониторинг состояния программы экземпляра функционального блока на языке ST. Для мониторинга состояния программы на языке ST следует либо дважды щелкнуть по экземпляру функционального блока, либо щелкнуть правой кнопкой мыши по экземпляру и выбрать пункт Monitor FB Instance (Мониторинг экземпляра функц. блока) в контекстном меню. В режиме мониторинга также можно изменять текущие значения, а также принудительно устанавливать/сбрасывать биты. Примечание. Онлайн-редактирование не поддерживается.

Защита функциональных блоков с помощью пароля

Прежняя версия (версия 6.0)	Новая версия (версия 6.1)
В параметрах функционального блока можно запретить отображение программы определения функционального блока.	Может быть установлен один из двух следующих видов парольной защиты. • Парольная защита от чтения и записи. • Парольная защита только от записи.

Информация об изменениях в связи с обновлением версии: 5.0 -> 6.0**Вложение функциональных блоков**

Прежняя версия (версия 5.0)	Новая версия (версия 6.0)
Вызов одного функционального блока из другого невозможен (вложение функциональных блоков не поддерживается).	Один функциональный блок может быть вызван из другого (вложение функциональных блоков). Допускается до 8 уровней вложения. Вызывающий функциональный блок и вызываемый функциональный блок могут быть запрограммированы на разных языках (LD или ST). Взаимосвязь между вложенными функциональными блоками может быть отображена в виде иерархического дерева. При сохранении некоторого функционального блока в файл библиотеки функциональных блоков (файл с расширением .sxf) в этот файл также сохраняются все вложенные функциональные блоки (т. е. блоки, вызываемые сохраняемым блоком).

Поддержка мониторинга битов входов/выходов в программах функциональных блоков на языке LD

Прежняя версия (версия 5.0)	Новая версия (версия 6.0)
Во время мониторинга прикладной программы в режиме онлайн мониторинг входных/выходных состояний программы экземпляра функционального блока на языке LD невозможен. (Можно было только проверить программу определения функционального блока.)	Во время мониторинга основной программы в режиме онлайн также можно осуществлять мониторинг входных/выходных состояний программы экземпляра функционального блока, написанной на языке LD. Для мониторинга входных/выходных состояний следует либо дважды щелкнуть по экземпляру функционального блока, либо щелкнуть правой кнопкой мыши по экземпляру и выбрать Monitor FB Ladder Instance (Мониторинг LD экземпляра функц. блока) в контекстном меню. В режиме мониторинга также можно производить мониторинг состояний битов входов/выходов и содержимого слов, изменять текущие значения, принудительно устанавливать/сбрасывать биты, контролировать смену состояний битов. Примечание. Онлайн-редактирование не поддерживается, изменение уставок таймеров/счетчиков невозможно.

Регистрация и мониторинг переменных экземпляра функционального блока в окне таблицы мониторинга

Прежняя версия (версия 5.0)	Новая версия (версия 6.0)
Для регистрации переменной некоторого экземпляра функционального блока в окне таблицы мониторинга (Watch) требовалось отобразить окно таблицы мониторинга, дважды щелкнуть по нему и выбрать требуемую переменную в раскрывающемся списке.	Можно легко зарегистрировать сразу несколько переменных экземпляра функционального блока для наблюдения за ними в окне таблицы мониторинга. Для вызова диалогового окна регистрации переменных функционального блока можно использовать любой из следующих способов. В данном окне могут быть зарегистрированы сразу все требуемые переменные. <ul style="list-style-type: none"> Щелкнуть правой кнопкой мыши по экземпляру функционального блока и выбрать пункт Register in Watch Window (Зарегистрировать в окне таблицы мониторинга) в контекстном меню. Выделить в программе требуемый экземпляр функционального блока или таблицу переменных и разместить их в окне таблицы мониторинга путем копирования/вставки или просто путем перетаскивания. Разместить курсор на пустой строке окна таблицы мониторинга и выбрать пункт Register in Watch Window (Зарегистрировать в окне таблицы мониторинга) в контекстном меню.

Прочие улучшения в части поддержки функциональных блоков

- Для программы функционального блока, написанной на языке LD, может быть вызвано всплывающее окно с информацией о перекрестных ссылках.
- В редакторе программ на языке ST с помощью всплывающего меню может быть вызвана Справка по языку программирования ST.
- Определение функционального блока можно вызвать, просто дважды щелкнув по экземпляру функционального блока.
- После подтверждения ввода параметра экземпляра функционального блока курсор автоматически перемещается вниз.

РАЗДЕЛ 2

Характеристики функциональных блоков

Данный раздел содержит технические данные и характеристики, которыми следует руководствоваться при использовании функциональных блоков, в том числе технические характеристики функциональных блоков, экземпляров и совместимых ПЛК, а также меры предосторожности и указания по использованию.

2-1	Характеристики функциональных блоков	38
2-1-1	Характеристики функциональных блоков	38
2-1-2	Элементы функциональных блоков	39
2-2	Типы данных, поддерживаемые в функциональных блоках.	52
2-2-1	Основные типы данных	52
2-2-2	Производные типы данных	53
2-3	Характеристики экземпляров.	53
2-3-1	Состав экземпляра	53
2-3-2	Характеристики параметров	59
2-3-3	Порядок работы	61
2-4	Ограничения при программировании.	63
2-4-1	Ограничения при программировании на языке LD.	63
2-4-2	Ограничения при программировании на языке ST	66
2-4-3	Ограничения при программировании.	66
2-5	Указания по использованию функциональных блоков	69
2-5-1	Выбор типа данных для переменной	69
2-5-2	Определение типа переменной (входные, выходные, входные-выходные, внешние и внутренние).	70
2-5-3	Использование параметра AT для внутренних переменных	73
2-5-4	Определение входных-выходных и внутренних переменных в качестве массива.	73
2-5-5	Указание адресов для специальных модулей ввода/вывода	75
2-5-6	Использование регистров указателей.	77
2-6	Меры предосторожности в отношении команд с операндами, указывающими первое или последнее слово группы слов.	80
2-7	Поддерживаемые команды и ограничения в отношении операндов	84
2-8	Характеристики функциональных блоков в модулях ЦПУ	85
2-8-1	Технические характеристики.	85
2-8-2	Особенности работы команд таймеров.	92
2-9	Количество шагов в программах функциональных блоков и время выполнения экземпляра	93
2-9-1	Количество шагов в программах функциональных блоков	93
2-9-2	Время выполнения экземпляра функционального блока	95

2-1 Характеристики функциональных блоков

2-1-1 Характеристики функциональных блоков

Параметр	Описание
Количество определений функциональных блоков	<p>Модули ЦПУ CJ2H:</p> <ul style="list-style-type: none"> • CJ2H-CPU6□(-EIP): макс. 2048 на модуль ЦПУ <p>Модули ЦПУ CJ2M:</p> <ul style="list-style-type: none"> • CJ2M-CPU□1/□2/□3: макс. 256 на модуль ЦПУ • CJ2M-CPU□4/□5: макс. 2048 на модуль ЦПУ <p>Модули ЦПУ CS1-H/CJ1-H:</p> <ul style="list-style-type: none"> • Суффикс -CPU44H/45H/64H/65H/66H/67H/64H-R/65H-R/66H-R/67H-R: макс. 1024 на модуль ЦПУ • Суффикс -CPU42H/43H/63H: макс. 128 на модуль ЦПУ <p>Модули ЦПУ CJ1M:</p> <ul style="list-style-type: none"> • CJ1M-CPU11/12/13/21/22/23: макс. 128 на модуль ЦПУ <p>Модули ЦПУ CP1H:</p> <ul style="list-style-type: none"> • CP1H-XA/X/Y: макс. 128 на модуль ЦПУ <p>Модули ЦПУ CP1L:</p> <ul style="list-style-type: none"> • CP1L-M/L: макс. 128 на модуль ЦПУ <p>Контроллеры NSJ:</p> <ul style="list-style-type: none"> • Все модели: макс. 1024 на контроллер <p>Контроллеры движения FQM1:</p> <ul style="list-style-type: none"> • FQM1-CM002/MMA22/MMP22: макс. 128 на контроллер
Количество экземпляров	<p>Модули ЦПУ CJ2H:</p> <ul style="list-style-type: none"> • CJ2H-CPU6□(-EIP): макс. 2048 на модуль ЦПУ <p>Модули ЦПУ CJ2M:</p> <ul style="list-style-type: none"> • CJ2M-CPU□1/□2/□3: макс. 256 на модуль ЦПУ • CJ2M-CPU□4/□5: макс. 2048 на модуль ЦПУ <p>Модули ЦПУ CS1-H/CJ1-H:</p> <ul style="list-style-type: none"> • Суффикс -CPU44H/45H/64H/65H/66H/67H/64H-R/65H-R/66H-R/67H-R: макс. 2048 на модуль ЦПУ • Суффикс -CPU42H/43H/63H: макс. 256 на модуль ЦПУ <p>Модули ЦПУ CJ1M:</p> <ul style="list-style-type: none"> • CJ1M-CPU11/12/13/21/22/23: макс. 256 на модуль ЦПУ <p>Модули ЦПУ CP1H:</p> <ul style="list-style-type: none"> • CP1H-XA/X/Y: макс. 256 на модуль ЦПУ <p>Модули ЦПУ CP1L:</p> <ul style="list-style-type: none"> • CP1L-M/L: макс. 256 на модуль ЦПУ <p>Контроллеры NSJ:</p> <ul style="list-style-type: none"> • Все модели: макс. 2048 на контроллер <p>Контроллеры движения FQM1:</p> <ul style="list-style-type: none"> • FQM1-CM002/MMA22/MMP22: макс. 256 на контроллер
Количество уровней вложения экземпляров	<ul style="list-style-type: none"> • CX-Programmer версии 5.0: вложение не поддерживается. • CX-Programmer версии 6.0 и более поздних версий: допускается до 8 уровней вложения. (Экземпляр, вызываемый из программы, учитывается как один уровень вложения.)
Количество переменных, используемое в функциональном блоке (не включая внутренние переменные, внешние переменные, EN и EN0)	<p>Макс. количество переменных на одно определение функционального блока</p> <ul style="list-style-type: none"> • Входные-выходные переменные: макс. 16 • Входные переменные + входные-выходные переменные: макс. 64 • Выходные переменные + входные-выходные переменные: макс. 64

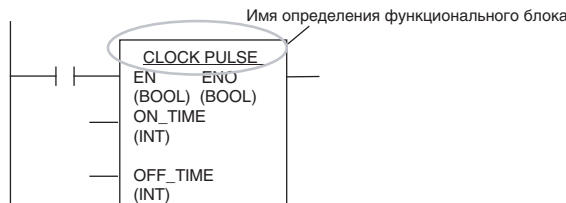
2-1-2 Элементы функциональных блоков

В следующей таблице перечислены все составляющие функционального блока, которые должны быть введены пользователем при создании определения функционального блока.

Параметр	Описание
Имя определения	Имя определения функционального блока
Язык	Язык программирования, используемый в определении функционального блока. Следует выбрать язык релейно-контактных схем (LD) или язык структурированного текста (ST).
Определения переменных	Атрибуты и параметры переменных, необходимые для выполнения функционального блока: операнды, возвращаемые значения и т. п. <ul style="list-style-type: none"> • Тип переменной (тип использования) • Имя переменной • Тип данных переменной • Начальное значение переменной
Алгоритм	Программа функционального блока на языке LD или ST. <ul style="list-style-type: none"> • В качестве обрабатываемых данных (операндов) в программе используются переменные. • Также можно вводить непосредственно фиксированные значения (константы), не регистрируя переменные.
Комментарий	Функциональный блок можно снабжать комментарием.

Имя определения функционального блока

Каждое определение функционального блока обладает именем. Длина имени не должна превышать 64 символа, среди которых не должно быть запрещенных символов. По умолчанию функциональному блоку присваивается имя FunctionBlock□, где □ — номер, присваиваемый в порядке очередности.



Язык

Программу функционального блока можно создавать либо на языке релейно-контактных схем (LD), либо на языке структурированного текста (ST).

Примечание

- (1) Подробные сведения о языке программирования ST см. в РАЗДЕЛ 5 Характеристики языка структурированного текста в Часть 2: Структурированный текст (ST).
- (2) При вложении функциональных блоков допускается комбинировать языки ST и LD произвольным образом (только в версии 6.0 выше).

Определения переменных

Операнды и переменные, используемые в определении функционального блока, должны быть заранее определены (skonфигурированы).

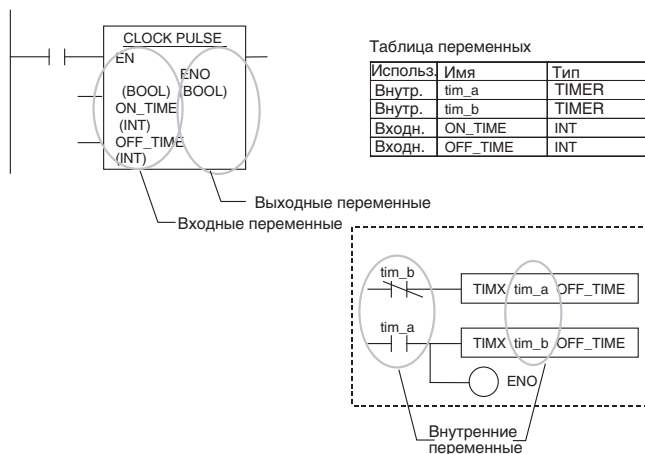
Имена переменных

- Имя переменной может состоять максимум из 30 000 символов.
- Имя переменной не должно содержать «пробелы», а также следующие символы:
! " # \$ % & ' () = - ~ ^ \ | ' @ { [+ ; * : } < , > . ? /
- Имя переменной не должно начинаться с цифры (0...9).

- Имя переменной не должно содержать два символа подчеркивания подряд.
- Не допускается использовать следующие символы для обозначения адресов памяти ввода/вывода.

Буквы A, W, H (или HR), D (или DM), E (или EM), T (или TIM), C (или CNT) с числовыми значениями после них (обозначающие адрес слова).

Обозначения переменных



Классификация переменных по типу использования

Параметр (см. примеч. 3)	Тип переменной				
	Входная	Выходная	Входная- выходная	Внутренняя	Внешняя (см. примеч. 1)
Определение	Ввод данных в экземпляр	Вывод данных из экземпляра	Переменные, используемые для ввода данных в экземпляр и вывода данных из экземпляра с использованием адресов	Переменные, используемые только внутри экземпляра	Глобальные символы, уже зарегистрированные в CX-Programmer как переменные, или глобальные символы, определенные пользователем.
Состояние значения при следующем выполнении	Передается значение входного параметра.	Значение продолжает использоваться при следующем выполнении.	Значение внешнего параметра.	Значение продолжает использоваться при следующем выполнении.	Значение внешней зарегистрированной переменной
Отображение	Отображается в левой части экземпляра.	Отображается в правой части экземпляра.	Отображаются в левой и правой частях экземпляра.	Не отображается.	Не отображается.
Допустимое количество	Макс. 64 на функц. блок (не включая EN)	Макс. 64 на функц. блок (не включая ENO)	Макс. 16 на функц. блок	Не ограничено	Не ограничено
Параметр AT	Нет	Нет	Нет	Поддерживается	Нет
Определение в качестве массива	Нет	Нет	Поддерживается	Поддерживается	Нет

Параметр (см. примеч. 3)	Тип переменной				
	Входная	Выходная	Входная- выходная	Внутренняя	Внешняя (см. примеч. 1)
Выбор сохранения значения	Поддерживается (см. примеч. 2)	Поддерживается	Нет	Поддерживается	Нет
Переменные, создаваемые по умолчанию	EN (Разрешение): принимает входное условие.	ENO (Выходное разрешение): выдает состояние выполнения функционального блока.	Нет	Нет	Символы, заранее зарегистрированные в CX-Programmer в качестве переменных, например флаги условий и некоторые из битов вспомогательной области.

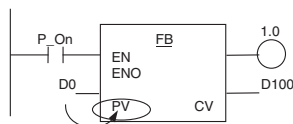
Примечание

- (1) Подробные сведения о внешних переменных см. в Приложение А Системные внешние переменные, поддерживаемые в функциональных блоках.
- (2) Передается значение входного параметра.
- (3) Структурные переменные и переменные типа TIMER/COUNTER могут использоваться только в качестве переменных следующего типа:
 Структурные переменные: внутренние, входные-выходные и внешние переменные.
 Переменные типа TIMER/COUNTER: внутренние и внешние переменные.

■ Входные переменные

Входные переменные служат для передачи содержимого внешних операндов в экземпляр функционального блока. Входные переменные отображаются в левой части экземпляра.

Во входную переменную передается значение, которое содержится в соответствующем входном параметре непосредственно в момент вызова экземпляра функционального блока.



Значение указанного входного параметра (значение D0) передается во входную переменную экземпляра функц. блока (PV).

Пример

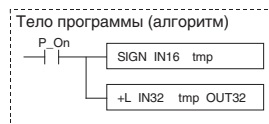
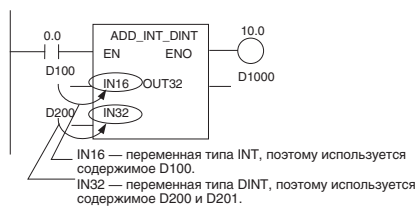
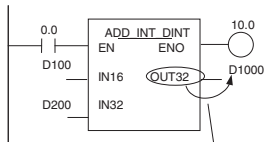


Таблица переменных

Используй	Имя	Тип
Внутр.	tmp	DINT
Входн.	EN	BOOL
Входн.	IN16	INT
Входн.	IN32	DINT
Выходн.	ENO	BOOL
Выходн.	OUT32	DINT

Пример



OUT32 — переменная типа DINT, поэтому значение переменной передается в D1000 и D1001.

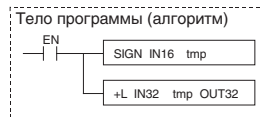


Таблица переменных

Используй	Имя	Тип данных
Внутр.	tmp	DINT
Входн.	EN	BOOL
Входн.	IN16	INT
Входн.	IN32	DINT
Выходн.	ENO	BOOL
Выходн.	OUT32	DINT

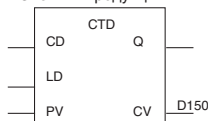
Name	Data Type	AT	Initial Value	Retained	Comment
ENO	BOOL		FALSE		Indicates successful execution of the Fun...
EN0	BOOL		FALSE		
OUT32	DINT		0		

Как и внутренние переменные, выходные переменные сохраняют свои значения вплоть до следующего выполнения экземпляра (т. е. когда выключается вход EN, значение выходной переменной не изменяется).

Пример.

В приведенном ниже примере содержимое выходной переменной CV не изменяется, пока данный экземпляр функционального блока не выполняется еще раз.

Счетчик продукции A



Примечание

1. Одно и то же имя не может быть присвоено одновременно входной и выходной переменным. Если требуется получить значение от внешней переменной, измените переменную внутри функционального блока, а затем возвратите результат во внешнюю переменную, либо используйте входную-выходную переменную.
2. Выходные значения передаются из выходных переменных в выходные параметры после полного выполнения внутренней программы функционального блока. Во время работы экземпляра содержимое выходной переменной в выходной параметр передано быть не может. Если требуется получить некоторое значение от функционального блока непосредственно во время выполнения его программы, нет смысла считывать это значение из параметра. Следует присвоить значение внутренней переменной и использовать параметр AT (для принудительного назначения адреса).

Исходное значение

Начальное значение может быть задано для выходной переменной, содержимое которой не сохраняется (т. е. не выбрана опция Retain (Сохранение)). Если для выходной переменной выбрана опция Retain (Сохранение), начальное значение для этой переменной задать невозможно.

Начальное значение также не будет записано в выходную переменную, если установлен бит сохранения памяти ввода/вывода (A500.12).

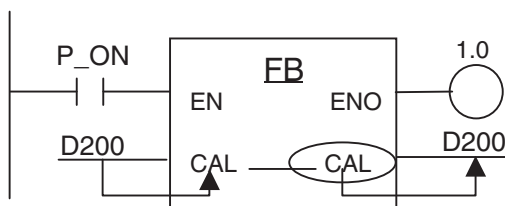
Управляющий бит вспомогательной области		Исходное значение
Бит сохранения IOM (A500.12)	ВКЛ	Исходное значение не устанавливается.

Переменная ENO («выход разрешения»)

Переменная ENO является выходной переменной, создаваемой по умолчанию. Выход переменной ENO включается, когда производится вызов экземпляра. Состояние этой переменной может быть изменено пользователем. По состоянию выходной переменной ENO можно судить о том, нормально ли завершилось выполнение экземпляра.

Входные-выходные переменные

Входные-выходные переменные служат для передачи данных внутрь экземпляра и получения данных из экземпляра с использованием адресов. Входная-выходная переменная отображается одновременно в левой и правой частях экземпляра. По завершении выполнения экземпляра значение входной-выходной переменной записывается не во внутренний адрес, выделенный для входной-выходной переменной автоматически, а по адресу (и по следующим за ним адресам, если того требует тип данных (размер) переменной), указанному в параметре, который используется для передачи данных во входную-выходную переменную и получения данных от нее.



Адрес D200 передается во входную-выходную переменную CAL. Внутри функционального блока обрабатывается указанное количество слов памяти ввода/вывода, начиная с D200. Таким образом, изменения оказываются доступны за пределами экземпляра функционального блока.

Примечание. Для того чтобы определить переменную в качестве «входной-выходной», в таблице переменных (в CX-Programmer) в графе Usage («Использование») следует выбрать «In Out» (Вх.-Вых.).

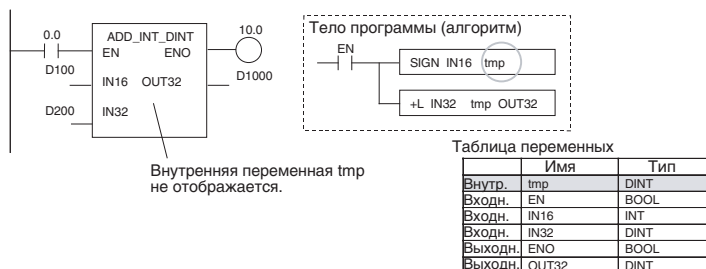
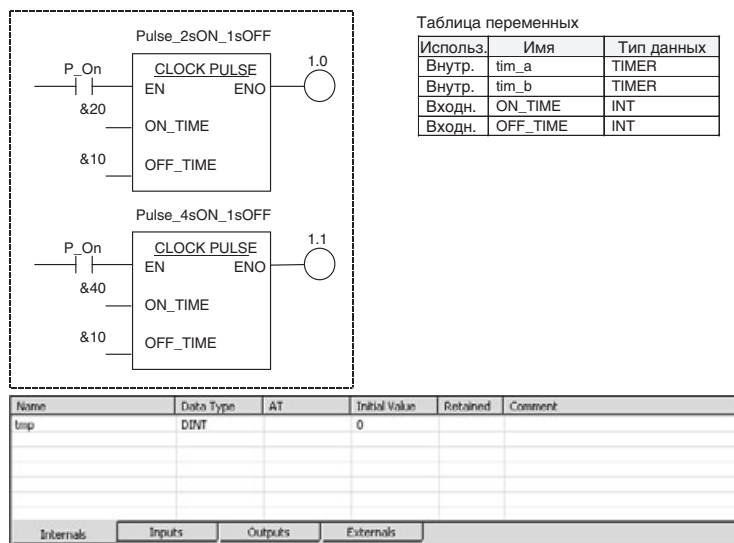
■ **Внутренние переменные**

Внутренние переменные используются внутри экземпляра. Внутренние переменные недоступны за пределами экземпляра и не отображаются на изображении экземпляра в программе.

Внутренние переменные сохраняют свои значения вплоть до следующего выполнения экземпляра (т. е. при выключении входа EN значения внутренних переменных не изменяются). Следовательно, даже если экземпляры одного и того же определения функционального блока выполняются с одними и теми же входными и выходными параметрами, они могут возвращать разные результаты.

Пример.

Внутренняя переменная *tim_a* экземпляра *Pulse_2sON_1sOFF* и внутренняя переменная *tim_a* экземпляра *Pulse_4sON_1sOFF* — это разные переменные. Ни один из экземпляров не может обращаться к переменной *tim_a* другого экземпляра и не может повлиять на ее значение.



Сохранение значений при прерывании питания и в начале работы

Внутренние переменные сохраняют значения, полученные ими при самом последнем вызове экземпляра функционального блока. Помимо этого для внутренней переменной может быть выбрана опция Retain (Сохранение). В этом случае текущее значение внутренней переменной также сохраняется при прерывании питания и при запуске программы модуля ЦПУ (т. е. при переключении из режима «Программирование» в режим «Выполнение» или «Мониторинг»).

Даже если выбрана опция Retain (Сохранение), обязательным условием сохранения значений переменных при прерывании питания или в начале работы программы является наличие батареи в модуле ЦПУ. При отсутствии в модуле ЦПУ технически исправной батареи сохранение значений не гарантировано.

Переменные	Условие	Состояние
Переменные, для которых выбрана опция <i>Retain</i> (Сохранение)	В начале работы	Сохраняются
	При включении питания	Сохраняются

Если опция Retain (Сохранение) не выбрана, значения переменных при прерывании питания или при запуске программы не сохраняются. Даже если опция Retain (Сохранение) для переменной не выбрана, прежнее значение переменной будет сохраняться в начале работы модуля ЦПУ, если включен бит сохранения памяти ввода/вывода (A500.12). А если в настройках ПЛК соответствующим образом настроен параметр «Состояние бита сохранения IOM при запуске», прежнее значение переменной также будет сохраняться и при прерывании питания. Все это более детально поясняется в следующей таблице.

Переменные	Условие	Состояние бита сохранения IOM (A500.12)		
		ВЫКЛ	ВКЛ	
			«Состояние бита сохранения IOM при запуске» (настр. ПЛК) — выбрано сохранение	«Состояние бита сохранения IOM при запуске» (настр. ПЛК) — сохранение не выбрано
Переменные, для которых не выбрана опция <i>Retain</i> (Сохранение)	В начале работы	Не сохраняются	Сохраняются	Сохраняются
	При включении питания	Не сохраняются	Сохраняются	Не сохраняются

Примечание. Поддержка бита сохранения памяти ввода/вывода (A500.12) предусмотрена для совместимости с предыдущими моделями. Для сохранения значений переменных в функциональных блоках следует, однако, использовать опцию *Retain* (Сохранения), а не бит сохранения памяти ввода/вывода.

Исходное значение

Исходное значение может быть задано для внутренней переменной, значение которой не сохраняется (т. е. не выбрана опция Retain (Сохранение)). Если для внутренней переменной выбрана опция Retain (Сохранение), исходное значение для этой переменной задать невозможно.

В несохраняемые внутренние переменные в качестве начального значения записывается 0.

Начальные значения не записываются во внутренние переменные, если включен бит сохранения памяти ввода/вывода (A500.12).

Управляющий бит вспомогательной области	Исходное значение	
Бит сохранения IOM (A500.12)	ВКЛ	Исходное значение не устанавливается.
	ВЫКЛ	Устанавливается исходное значение.

■ **Внешние переменные**

Внешними переменными являются системные переменные, заранее зарегистрированные в CX-Programmer, а также переменные, ссылающиеся на переменные пользователя, зарегистрированные в таблице глобальных символов (внешние по отношению к функциональному блоку).

- Подробные сведения о системных переменных см. в *Приложение А Системные внешние переменные, поддерживаемые в функциональных блоках.*
- Для ссылки на внешнюю переменную, определенную пользователем в таблице глобальных символов, в таблице определения переменных функционального блока должна быть зарегистрирована внешняя переменная с таким же именем и типом данных. При этом, однако, невозможно сослаться на переменные, определенные пользователем в качестве сетевых символов.

Атрибуты переменных

Имя переменной

Имя переменной служит для идентификации этой переменной в функциональном блоке. Длина имени не должна превышать 30 000 символов. Одно и то же имя может использоваться в разных функциональных блоках.

Примечание. Даже если для переменной указывается конкретный адрес с помощью параметра АТ, для нее все равно должно быть введено имя.

Тип данных

Может использоваться любой из следующих типов данных.

Тип данных	Содержание	Размер	Вход-ные	Выход-ные	Вх.-вых.	Внут-ренние	Внешние
BOOL	Логическое значение	1 бит	ОК	ОК	ОК	ОК	ОК
INT	Целое	16 бит	ОК	ОК	ОК	ОК	ОК
UNIT	Целое без знака	16 бит	ОК	ОК	ОК	ОК	ОК
DINT	Двойное целое	32 бит	ОК	ОК	ОК	ОК	ОК
UDINT	Двойное целое без знака	32 бит	ОК	ОК	ОК	ОК	ОК
LINT	Длинное (4 слова) целое	64 бит	ОК	ОК	ОК	ОК	ОК
ULINT	Длинное (4 слова) целое без знака	64 бит	ОК	ОК	ОК	ОК	ОК
WORD	16-битовое значение	16 бит	ОК	ОК	ОК	ОК	ОК
DWORD	32-битовое значение	32 бит	ОК	ОК	ОК	ОК	ОК
LWORD	64-битовое значение	64 бит	ОК	ОК	ОК	ОК	ОК
REAL	Вещественное число	32 бит	ОК	ОК	ОК	ОК	ОК
LREAL	Длинное вещественное число	64 бит	ОК	ОК	ОК	ОК	ОК
TIMER	Таймер (см. примеч. 1)	Флаг: 1 бит Тек. зн.: 16 бит	Не предусм.	Не предусм.	Не предусм.	ОК	ОК
COUNTER	Счетчик (см. примеч. 2)	Флаг: 1 бит Тек. зн.: 16 бит	Не предусм.	Не предусм.	Не предусм.	ОК	ОК
STRING	Текстовая строка	Перемен-ный	Не предусм.	Не предусм.	ОК	ОК	Не предусм.
STRUCT	Пользовательский тип данных	Перемен-ный	Не предусм.	Не предусм.	ОК	ОК	ОК

Примечание (1) Тип данных TIMER используется для переменных, являющихся операндами команд-таймеров (TIM, TIMH и т. п.) и используемых для ввода номеров таймеров (0...4095). Если эта переменная

используется в другой команде, указывается флаг завершения таймера (1 бит) или текущее значение таймера (16 бит) (в зависимости от операнда команды).

- (2) Тип данных COUNTER используется для переменных, являющихся операндами команд-счетчиков (CNT, CNTR и т. п.) и используемых для ввода номеров счетчиков (0..4095). Если эта переменная используется в другой команде, указывается флаг завершения счетчика (1 бит) или текущее значение счетчика (16 бит) (в зависимости от операнда команды).

Параметр AT (назначение конкретных адресов)

Вместо автоматического назначения адреса системой для внутренней переменной может быть задан некоторый определенный адрес памяти ввода/вывода. Для этих целей предусмотрен параметр AT, в который пользователь может ввести требуемый адрес. Даже если для переменной задан конкретный адрес, в программе все равно должно использоваться имя переменной.

Примечание

- (1) Параметр AT может быть задан только для внутренней переменной.
- (2) С помощью параметра AT может быть указан адрес только одной из следующих областей памяти: CIO (область ввода/вывода), A (вспомогательная область), D (область памяти данных), E (расширенная область памяти), H (область хранения), W (внутренняя рабочая область).
С помощью параметра AT не может быть задан адрес ни одной из следующих областей памяти:
- Области регистров указателей и регистров данных (с прямым или косвенным указанием адресов)
 - Косвенно указываемые адреса областей DM/EM (в двоичном и двоично-десятичном формате)
- (3) С помощью параметра AT могут быть указаны следующие адреса
- Адреса памяти для базовых модулей ввода/вывода, модулей шины ЦПУ или специальных модулей ввода/вывода
 - Биты вспомогательной области, не зарегистрированные заранее в качестве внешних переменных
 - Адреса ПЛК в других узлах сети

Пример.

Если в определении функционального блока используется команда (FREAD) (Прочитать файл данных) и требуется проверить состояние флага операции над памятью файлов (A343.13), следует воспользоваться внутренней переменной, указав для нее с помощью параметра AT адрес данного флага.

Зарегистрируйте внутреннюю переменную, установите для нее галочку AT и укажите адрес A343.13. Теперь эта внутренняя переменная будет отражать состояние флага операций над памятью файлов.



Адрес A34313 закрепляется за внутренней переменной логического типа с именем NOW_CARD_ACCESS.

Если для какой-либо переменной функционального блока используется параметр AT, функциональный блок утрачивает свою универсальность. Поэтому параметром AT следует пользоваться только в случае крайней необходимости.

Определение массива

Внутренние переменные, а также входные-выходные переменные могут быть определены как массив.

Примечание.

В текущей версии программа CX-Programmer поддерживает только одномерные массивы.

Зарегистрировав всего одну переменную, но указав, что она является массивом, пользователь получает в свое распоряжение большое количество переменных с одинаковыми свойствами.

- Внутренняя переменная, определенная как массив, может содержать от 1 до 32 000 элементов. Количество элементов, которое может содержать входная-выходная переменная, определенная как массив, зависит от типа данных, что отражено в следующей таблице.

Тип данных	Количество элементов
BOOL	2048
INT/UINT/WORD	2048
DINT/UDINT/DWORD	1024
LINT/ULINT/LWORD	512

- Атрибут «массив» может быть выбран только для внутренней и входной-выходной переменной.
- Для внутренней переменной, определенной как массив, может быть указан любой тип данных, кроме STRING.
- При вводе имени переменной-массива в программе определения функционального блока после имени переменной в квадратных скобках следует указывать номер элемента массива. Для указания элемента массива можно использовать любой из трех следующих способов (во всех трех примерах a[] — это переменная-массив).
 - С помощью литерала (в программе на языке LD или ST)
Пример: a[2].
 - С помощью переменной (в программе на языке LD или ST)
Пример: a[n], где n — переменная.

Примечание. Может использоваться переменная с типом данных INT, DINT, LINT, UINT, UDINT или ULINT.

- С помощью выражения (только в программе на языке ST)
Пример: a[b+c], где b и c — переменные.

Примечание. Выражения могут содержать только арифметические операторы (+, -, * и /).

Массив — это совокупность элементов данных с одинаковым типом данных. Любой элемент массива идентифицируется по имени переменной (общее для всех элементов массива) и уникальному номеру (индексу) элемента. (Индекс фактически указывает на местоположение элемента в массиве.)

Элементы одномерного массива идентифицируются с помощью всего одного индекса.

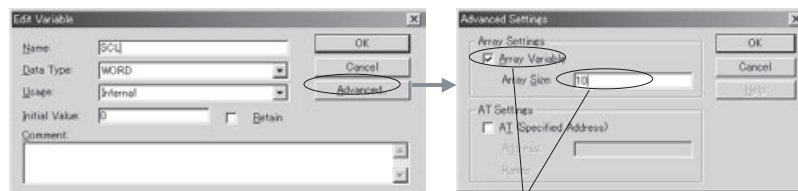
Пример. Если внутренняя переменная с именем SCL определена как массив с 10 элементами, в программе функционального блока могут использоваться 10 следующих переменных:

SCL[0], SCL[1], SCL[2], SCL[3], SCL[4], SCL[5], SCL[6], SCL[7], SCL[8] и SCL[9].

SCL

0	Перем. типа WORD
1	Перем. типа WORD
2	Перем. типа WORD
3	Перем. типа WORD
4	Перем. типа WORD
5	Перем. типа WORD
6	Перем. типа WORD
7	Перем. типа WORD
8	Перем. типа WORD
9	Перем. типа WORD

← Для обращения к этому элементу данных следует указать SCL[3].



Переменная SCL объявляется как переменная-массив с элементами с номерами 0...9.

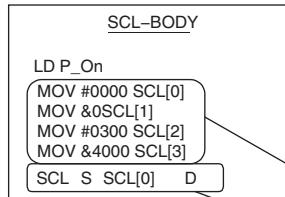
Примечание. Использование переменной-массива делает возможным повторное применение функционального блока в том случае, когда в операнде команды указывается первое или последнее слово некоторой последовательности слов, и в качестве операнда не может быть использована внутренняя переменная с параметром AT и не может быть использована внешняя переменная. Во входном параметре входной-выходной переменной, являющейся массивом, должен указываться адрес первого слова данных (CX-Programmer версии 7.0 и выше). Если некоторая внутренняя переменная определена как массив, во все ее элементы следует предварительно записать требуемые значения, а в операнде команды указать первый или последний элемент переменной-массива.

Пример:

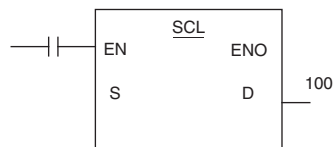
Определение функц. блока



Алгоритм



Экземпляр



SCL

0	#0000	← Указание этого элемента массива в команде SCL эквивалентно указанию адреса первого элемента.
1	&0	
2	#0300	
3	&4000	

В элементы переменной-массива записываются значения операндов.

В команде SCL указывается начало массива.

Примечание.

Подробные сведения смотрите в разделе 2-6 *Меры предосторожности в отношении команд с операндами, указывающими первое или последнее слово группы слов.*

Исходные значения

При первом выполнении экземпляра во входные, внутренние и выходные переменные могут быть записаны требуемые начальные значения. См. подробную информацию в подразделе *Исходное значение* в предшествующих разделах, посвященных входным, внутренним и выходным переменным.

Name	Data Type	AT	Initial Value	Retained	Comm
EN	BOOL		FALSE		Contr
CD	BOOL		FALSE		
LD	BOOL		FALSE		
PV	UINT		30		

Edit Variable

Name:

Data Type:

Usage:

Initial Value: Retain

Comment:

Сохранение значений переменных при прерывании питания и в начале работы

Содержимое внутренних переменных может сохраняться при прерывании питания и в начале работы модуля ЦПУ. Если для переменной выбрана опция Retain (Сохранение), для этой переменной отводится адрес в той области памяти, содержимое которой сохраняется при прерывании питания или при запуске программы ПЛК.

Алгоритм

Создайте программу функционального блока, используя зарегистрированные переменные.

Ограничения при вводе операндов

В операнды команд программы функционального блока не могут вводиться непосредственно адреса ячеек памяти. Даже если будет введен адрес, он будет восприниматься как имя переменной.

Примечание.

Исключение: напрямую или косвенно указываемые адреса для регистров указателей IR0...IR15 и регистров данных DR0...DR15 следует вводить непосредственно в операнды команды. Переменные в этом случае вводить не следует.

В качестве операндов команд можно также вводить фиксированные значения (константы).

- Программа на языке LD: вводится десятичное значение с префиксом & или шестнадцатеричное числовое значение с префиксом #.
- Язык структурированного текста: вводится десятичное числовое значение без префикса или шестнадцатеричное числовое значение с префиксом 16#.

Комментарий

Может быть введен комментарий длиной до 30 000 символов.

2-2 Типы данных, поддерживаемые в функциональных блоках

2-2-1 Основные типы данных

Тип данных	Содержание	Размер	Диапазон значений
BOOL	Логическое значение	1	0 (ЛОЖЬ), 1 (ИСТИНА)
INT	Целое	16	-32 768...+32 767
DINT	Двойное целое	32	-2 147 483 648...+2 147 483 647
LINT	Длинное (8 байтов) целое	64	-9 223 372 036 854 775 808...+9 223 372 036 854 775 807
UINT	Целое без знака	16	&0...65 535
UDINT	Двойное целое без знака	32	&0...4 294 967 295
ULINT	Длинное (8 байтов) целое без знака	64	&0...18 446 744 073 709 551 615
REAL	Вещественное число	32	-3,402823 × 10 ³⁸ ...-1,175494 × 10 ⁻³⁸ , 0, +1,175494 × 10 ⁻³⁸ ...+3,402823 × 10 ³⁸
LREAL	Длинное вещественное число	64	-1,79769313486232 × 10 ³⁰⁸ ...-2,22507385850720 × 10 ⁻³⁰⁸ , 0, 2,22507385850720 × 10 ⁻³⁰⁸ ...1,79769313486232 × 10 ³⁰⁸
WORD	16-битовое значение	16	#0000...FFFF или &0...65 535
DWORD	32-битовое значение	32	#00000000...FFFFFFFF или &0...4 294 967 295
LWORD	64-битовое значение	64	#0000000000000000...FFFFFFFFFFFFFFFF или &0...18 446 744 073 709 551 615
STRING	Текстовая строка	Переменная	1...255 символов ASCII
TIMER	Таймер	Флаг: 1 бит Тек. зн.: 16 бит	Номер таймера: 0...4095 Флаг завершения: 0 или 1 Тек. знач. таймера: 0...9999 (BCD), 0...65535 (двоичн.)
COUNTER	Счетчик	Флаг: 1 бит Тек. зн.: 16 бит	Номер счетчика: 0...4095 Флаг завершения: 0 или 1 Тек. знач. счетчика: 0...9999 (BCD), 0...65535 (двоичный)
FUNCTION BLOCK	Экземпляр функционального блока	---	---

2-2-2 Производные типы данных

Тип данных	Содержание
Массив	1-мерный массив; макс. 32 000 элементов
Структура	Пользовательский тип данных

2-3 Характеристики экземпляров

2-3-1 Состав экземпляра

В следующей таблице перечислены компоненты, которые должны быть определены пользователем при регистрации экземпляра функционального блока.

Компонент	Описание
Имя экземпляра	Собственное имя экземпляра
Язык Определения переменных	Тот же язык программирования и те же переменные, что и в определении функционального блока.
Области памяти экземпляра функц. блока	Диапазоны адресов памяти, используемые для переменных
Комментарии	Для каждого экземпляра может быть введен комментарий.

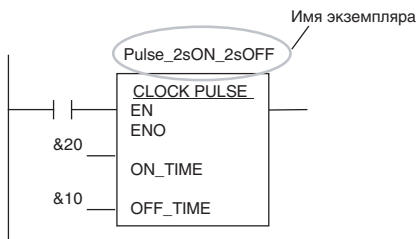
Имя экземпляра

Собственное имя экземпляра.

- Имя экземпляра может содержать максимум 30 000 символов.
- Имя экземпляра не должно содержать «пробелы» и ни один из следующих символов:
! " # \$ % & ' () = - ~ ^ \ | ' @ { [+ ; * : } < , > . ? /
- Имя экземпляра не должно начинаться с цифры (0...9).

Каких-либо других ограничений не существует.

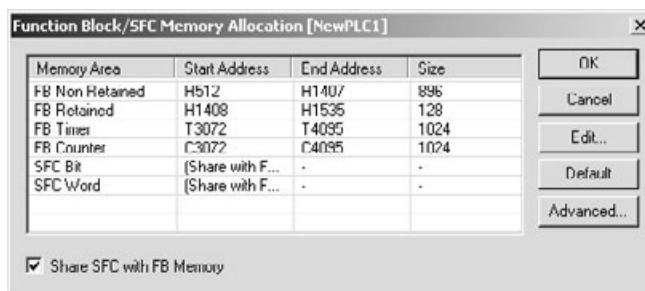
В лестничной диаграмме имя экземпляра отображается над символом экземпляра.



Области памяти экземпляра функц. блока

Для того чтобы экземпляр функционального блока мог быть выполнен, для всех его внутренних, входных, выходных и входных-выходных переменных должны быть выделены адреса в тех или иных областях памяти модуля ЦПУ. Эти области памяти называются областями экземпляра функционального блока. Начальные адреса и размеры этих областей заранее указываются пользователем. Единицей измерения (т. е. минимальным элементом данных) при указании начальных адресов и размеров областей является одно слово данных.

Если в процессе компиляции программы будет обнаружено, что программа пользователя обращается к словам одной из областей, выделенных для экземпляра функционального блока, CX-Programmer выдаст сообщение об ошибке.



Модули ЦПУ серии CJ2

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	H512	H1407	896	CIO, WR, HR, DM, EM (см. примеч.)
Сохран.	H1408	H1535	128	HR, DM, EM (см. примеч.)
Таймеры	T3072	T4095	1024	TIM
Счетчики	C3072	C4095	1024	CNT

Примечание. Принудительная установка/сброс возможны, если указаны следующие банки памяти EM:

CJ2H-CPU64(-EIP)/-CPU65(-EIP)	EM, банк 3
CJ2H-CPU66(-EIP)	EM, банки 6...9
CJ2H-CPU67(-EIP)	EM, банки 7...E
CJ2H-CPU68(-EIP)	EM, банки 11...18

Модули ЦПУ серии CS/CJ версии 3.0 или выше и контроллеры NSJ

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	H512	H1407	896	CIO, WR, HR, DM, EM
Сохран.	H1408	H1535	128	HR, DM, EM
Таймеры	T3072	T4095	1024	TIM
Счетчики	C3072	C4095	1024	CNT

Контроллеры движения FQM1

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	5000	5999	1000	CIO, WR, DM
Сохран.	Нет			
Таймеры	T206	T255	50	TIM
Счетчики	C206	C255	50	CNT

Модули ЦПУ серии CP

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	H512	H1407	896	CIO, WR, HR, DM (см. примеч.)
Сохран.	H1408	H1535	128	HR, DM (см. примеч.)
Таймеры	T3072	T4095	1024	TIM
Счетчики	C3072	C4095	1024	CNT

Примечание.

Область DM в ЦПУ CP1L-L

Адрес	CP1L-L
D0000...D9999	Есть
D10000...D31999	Нет
D32000...D32767	Есть

Типы областей для экземпляров функц. блоков

Ниже поясняются аспекты настройки областей памяти для экземпляров функциональных блоков для модулей ЦПУ и контроллеров разных моделей.

Модули ЦПУ серии CS/CJ версии 3.0 и выше, ПЛК серии CP и контроллеры NSJ

Не сохраняемые области

Параметр	Содержание
Размещаемые переменные	Переменные, для которых не выбрано сохранение содержимого при выключении питания и запуске работы (см. примеч. 1).
Применимые области	H (спец. область хранения для функциональных блоков), I/O (область CIO), N (область хранения), W (внутренняя рабочая область), D (область памяти данных) (см. примеч. 2), E (расширенная область памяти данных) (см. примеч. 2 и 3)
Минимальный размер элемента данных	Слово
Выделяемые слова (по умолчанию)	H512...H1407

Примечание

- (1) Кроме случаев, когда выбран тип данных TIMER или COUNTER.
- (2) Обращение к значениям битов возможно, даже если область DM или EM указана для несохраняемой области или сохраняемой области.
- (3) Если область EM указана для несохраняемой области или сохраняемой области, банк с таким же номером не может быть указан в качестве текущего банка в программе пользователя.

Сохраняемая область

Параметр	Содержание
Размещаемые переменные	Переменные, для которых выбрано сохранение содержимого при выключении питания и запуске работы (см. примеч. 1).
Применимые области	H (спец. область хранения для функциональных блоков), N (область хранения), D (область памяти данных) (см. примеч. 1), E (расширенная область памяти данных) (см. примеч. 2 и 3)

Параметр	Содержание
Минимальный размер элемента данных	Слово
Выделяемые слова (по умолчанию)	H1408...H1535

Примечание

- (1) Кроме случаев, когда выбран тип данных TIMER или COUNTER.
- (2) Обращение к значениям битов возможно, даже если область DM или EM указана для несохраняемой области или сохраняемой области.
- (3) Если область EM указана для несохраняемой области или сохраняемой области, банк с таким же номером не может быть указан в качестве текущего банка в программе пользователя.

Область таймеров

Параметр	Содержание
Размещаемые переменные	Переменные, для которых указан тип данных TIMER.
Применимые области	T (область таймеров): флаги завершения (1 бит) или текущие значения (16 бит) таймеров
Выделяемые слова (по умолчанию)	T3072...T4095: флаги завершения (1 бит) или текущие значения (16 бит) таймеров

Область счетчиков

Параметр	Содержание
Размещаемые переменные	Переменные, для которых указан тип данных COUNTER.
Применимые области	C (область счетчиков): флаги завершения (1 бит) или текущие значения (16 бит) счетчиков
Выделяемые слова (по умолчанию)	C3072...C4095: флаги завершения (1 бит) или текущие значения (16 бит) счетчиков

Область хранения для функциональных блоков (H512...H1535)

В качестве слов области хранения для функциональных блоков по умолчанию назначаются сохраняемые и несохраняемые слова по адресам H512...H1535. Эти слова отличаются от стандартных слов области хранения, используемых для программ (H000...H511), и предназначены только для области экземпляров функциональных блоков (адреса которой распределяются между переменными автоматически).

- Адреса этих слов не могут быть указаны с помощью параметра AT для внутренних переменных.
- Эти слова нельзя указывать в качестве операндов команд.
 - Если эти слова используются в любом другом месте (т. е. не в программе функционального блока), они отображаются красным цветом.
 - Хотя ввод этих слов при создании функционального блока возможен, при проверке программы будет выдано сообщение об ошибке.
- Если эта область определяется как несохраняемая, содержимое данной области после выключения питания и при запуске выполнения программы не сохраняется.

Примечание.

Для того чтобы адреса области памяти экземпляра не смешивались с адресами, которые используются прикладной программой, в качестве несохраняемой и сохраняемой областей памяти следует использовать

адреса H512...H1535 (слова области хранения для функциональных блоков). Если этих слов окажется недостаточно, следует задействовать слова других областей памяти, не используемых в программе пользователя.

Контроллер движения FQM1

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	5000	5999	1000	CIO, WR, DM
Сохран.	Нет			
Таймеры	T206	T255	50	TIM
Счетчики	C206	C255	50	CNT

Не сохраняемые области

Параметр	Содержание
Размещаемые переменные	Переменные, для которых выбрано сохранение содержимого при выключении питания и запуске работы (см. примеч. 1).
Применимые области	I/O (CIO), W (рабочая область) и D (область DM) (см. примеч. 2)
Минимальный размер элемента данных	Слово
Выделяемые слова (по умолчанию)	CIO 5000...CIO 5999

Примечание

- (1) Кроме случаев, когда выбран тип данных TIMER или COUNTER.
- (2) Обращение к состояниям битов возможно, даже если область DM указана для несохраняемой области.

Сохраняемая область

Нет

Область таймеров

Параметр	Содержание
Размещаемые переменные	Переменные, для которых указан тип данных TIMER.
Применимые области	T (область таймеров): флаги завершения (1 бит) или текущие значения (16 бит) таймеров
Выделяемые слова (по умолчанию)	T206...T255: флаги завершения (1 бит) или текущие значения (16 бит) таймеров

Область счетчиков

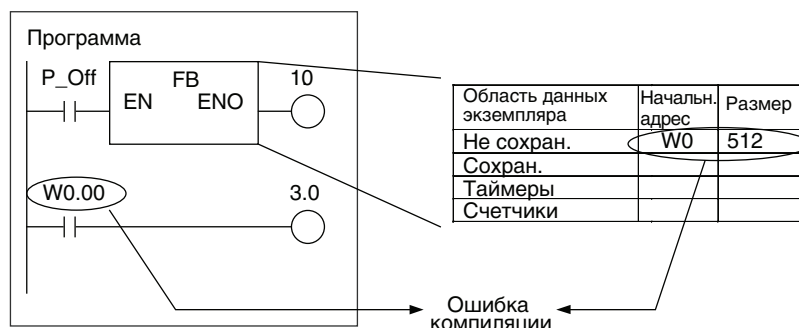
Параметр	Содержание
Размещаемые переменные	Переменные, для которых указан тип данных COUNTER.
Применимые области	C (область счетчиков): флаги завершения (1 бит) или текущие значения (16 бит) счетчиков
Выделяемые слова (по умолчанию)	C206...C255: флаги завершения (1 бит) или текущие значения (16 бит) счетчиков

Обращение к области экземпляра функц. блока из программы пользователя

Если прикладная программа ПЛК содержит команду, обращающуюся к области памяти экземпляра функционального блока, при попытке выполнения любой из следующих операций на вкладке Compile (Компиляция) окна вывода информации CX-Programmer будет отображено сообщение об ошибке.

- Попытка записи во время онлайн-редактирования (запись невозможна).
- Выполнение проверки программы (выбор команды *Compile* (Компилировать) в меню Program (Программа) или команды *Compile All PLC Programs* (Компилировать все программы ПЛК) в меню PLC (ПЛК)).

Пример. Если в качестве несохраняемой области памяти экземпляра функционального блока выбраны слова W0...W511, а в прикладной программе ПЛК используется бит W0.00, при компиляции программы произойдет ошибка и отобразится сообщение «ERROR: [опущено]...-Address - W0.00 is reserved for Function Block use» («ОШИБКА: [опущено]...- Адрес W0.00 зарезервирован за функциональным блоком»).



Примечание.

При удалении или добавлении переменной адреса, выделенные для переменных в области экземпляра функционального блока, автоматически переназначаются. Обязательным требованием, однако, является последовательное расположение адресов, используемых для одного экземпляра. Если это требование выполнить не удастся, происходит полное переназначение адресов переменных. При этом могут образоваться неиспользуемые участки памяти. В этом случае для более рационального использования областей памяти и удаления неиспользуемых сегментов следует выполнить операцию оптимизации.

Комментарии

Может быть введен комментарий длиной до 30 000 символов.

Создание нескольких экземпляров

Вызов одного и того же экземпляра

Допускается вызывать один и тот же экземпляр в разных местах программы. Экземпляр, вызываемый из разных мест программы, использует одни и те же внутренние переменные.

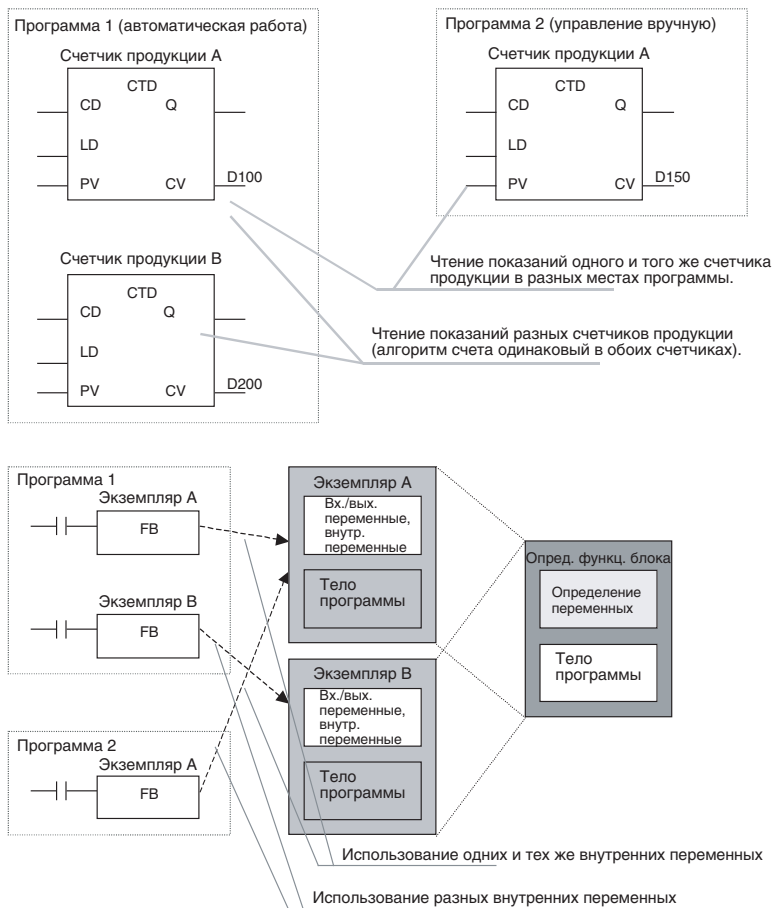
Создание нескольких экземпляров

В программе может быть создано множество экземпляров одного и того же определения функционального блока. В этом случае значения внутренних переменных в каждом экземпляре будут отличаться.

Пример: подсчет количества изделий А и изделий В

Подготовьте определение функционального блока под названием «СТД» (Вычитающий счетчик) и настройте счетчики для подсчета количества изделий А и изделий В. Создаются программы двух типов: одна — для работы в автоматическом режиме, другая — для работы в ручном режиме. Требуемый режим работы может быть выбран пользователем.

В данном случае будет создано несколько экземпляров одного функционального блока. При этом один и тот же экземпляр будет вызываться в разных сегментах программы.



2-3-2 Характеристики параметров

В следующей таблице перечислены данные, которые пользователь может вводить во входные и выходные параметры экземпляра функционального блока.

Параметр	Допустимые данные
Входные параметры	<p>Значения (см. примеч. 1), адреса и символьные имена программы (глобальные и локальные символы) (см. примеч. 2).</p> <p>Примечание. Фактический объем данных, передаваемый во входную переменную из входного параметра определяется типом данных входной переменной (даже если в параметре указывается адрес, само значение адреса не передается).</p> <p>Примечание. Настройка входных параметров обязательна. Если хотя бы один из входных параметров не будет задан, произойдет критическая ошибка и входные параметры в ПЛК переданы не будут.</p>
Выходные параметры	Адреса, символьные имена программы (глобальные и локальные символы) (см. примеч. 2)
Входные-выходные параметры	Адреса, символьные имена программы (глобальные и локальные символы)

Примечание (1) В следующей таблице перечислены способы ввода значений в параметры.

Тип данных входной переменной	Содержание	Размер	Способ ввода значения в параметр	Диапазон значений
BOOL	Логическое значение	1 бит	P_Off, P_On	0 (ЛОЖЬ), 1 (ИСТИНА)
INT	Целое	16 бит	Положительное значение: целое число со знаком «&» или «+» спереди Отрицательное значение: целое число со знаком «-» спереди	-32 768...32 767
DINT	Двойное целое	32 бит		-2 147 483 648...2 147 483 647
LINT	Длинное (8 байтов) целое	64 бит		-9 223 372 036 854 775 808... 9 223 372 036 854 775 807
UINT	Целое без знака	16 бит	Положительное значение: целое число со знаком «&» или «+» спереди	&0...65 535
UDINT	Двойное целое без знака	32 бит		&0...4 294 967 295
ULINT	Длинное (8 байтов) целое без знака	64 бит		&0...18 446 744 073 709 551 615
REAL	Вещественное число	32 бит	Положительное значение: вещественное число (с десятичной запятой) со знаком «&» или «+» спереди Отрицательное значение: вещественное число (с десятичной запятой) со знаком «-» спереди	$-3,402823 \times 10^{38} \dots -1,175494 \times 10^{-38}$, 0, $1,175494 \times 10^{-38} \dots 3,402823 \times 10^{38}$
LREAL	Длинное вещественное число	64 бит		$-1,79769313486232 \times 10^{308} \dots$ $-2,22507385850720 \times 10^{-308}$, 0, $2,22507385850720 \times 10^{-308}$, $1,79769313486232 \times 10^{308}$
WORD	16-битовое значение	16 бит	Шестнадцатеричное число (макс. 4 разряда) со знаком «#» спереди Десятичное число со знаком «&» или «+» спереди	#0000...FFFF или &0...65 535
DWORD	32-битовое значение	32 бит	Шестнадцатеричное число (макс. 8 разрядов) со знаком «#» спереди Десятичное число со знаком «&» или «+» спереди	#00000000...FFFFFFFF или &0...4 294 967 295
LWORD	64-битовое значение	64 бит	Шестнадцатеричное число (макс. 16 разряда) со знаком «#» спереди Десятичное число со знаком «&» или «+» спереди	#0000000000000000... FFFFFFFFFFFFFFFF или &0...18 446 744 073 709 551 615

(2) Размер входных и выходных переменных функционального блока должен совпадать с размером символьных имен программы (глобальных и локальных), что отражено в следующей таблице.

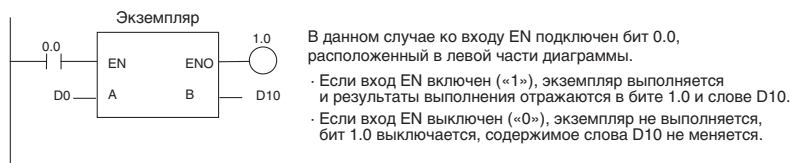
Размер	Тип данных переменной функц. блока	Тип данных символа программы (глобального или локального)
1 бит	BOOL	BOOL
16 бит	INT, UINT, WORD	INT, UINT, UINT BCD, WORD
32 бит	DINT, UDINT, REAL, DWORD	DINT, UDINT, UDINT BCD, REAL, DWORD
64 бит	LINT, ULINT, LREAL, LWORD	LINT, ULINT, ULINT BCD, LREAL, LWORD
Больше чем 1 бит	Не логический тип	CHANNEL, NUMBER (см. примеч.)

Примечание. Символ программы типа NUMBER может быть задан только для входного параметра. Вводимое значение при этом должно находиться в пределах диапазона, определяемого типом данных входной переменной функционального блока.

2-3-3 Порядок работы

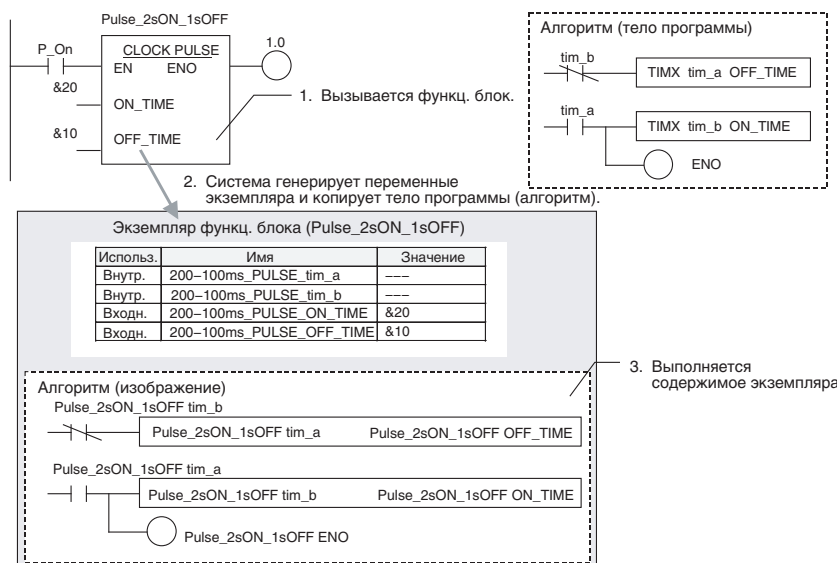
Вызов экземпляров

Экземпляр функционального блока может быть вызван в любом месте программы пользователя. Экземпляр выполняется, если включен его вход разрешения (EN).

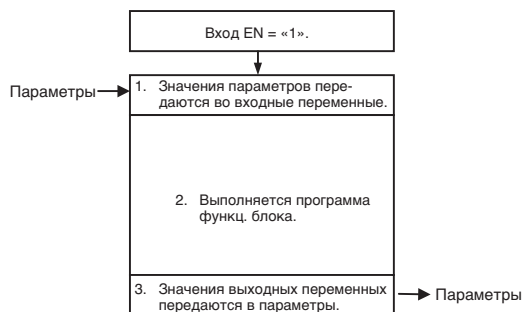


Порядок выполнения экземпляра

Вызываемый функциональный блок выполняется, если включен вход переменной EN. При вызове функционального блока система генерирует переменные экземпляра и создает копию программы функционального блока. После этого выполняется экземпляр.



- Выполнение экземпляра происходит в следующей последовательности:
1. Во входные переменные передается содержимое входных параметров.
 2. Выполняется программа функционального блока.
 3. В выходные параметры записывается содержимое выходных переменных.

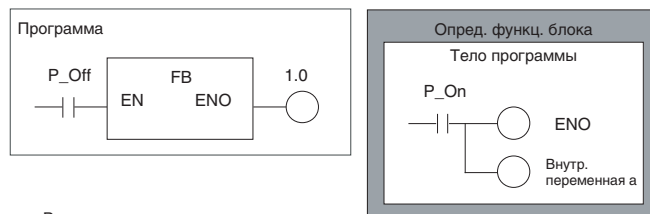


Во время своего выполнения программа не может обмениваться данными с параметрами.

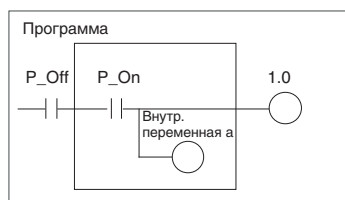
Кроме того, если содержание выходной переменной не изменяется в ходе выполнения программы, оно сохраняет свое прежнее значение.

Порядок работы, когда экземпляр не выполняется

Если вход входной переменной EN функционального блока выключен, вызов функционального блока не производится, поэтому внутренние переменные экземпляра сохраняют свои прежние значения. При выключенном входе EN также не изменяются значения выходных переменных.



Результаты выполнения: Выходная переменная 1.0 выключается, но внутренняя переменная сохраняет свое прежнее состояние.



Если бы данный участок программы был набран непосредственно в основной программе, а не в определении функционального блока, были бы выключены и бит 1.0, и переменная.

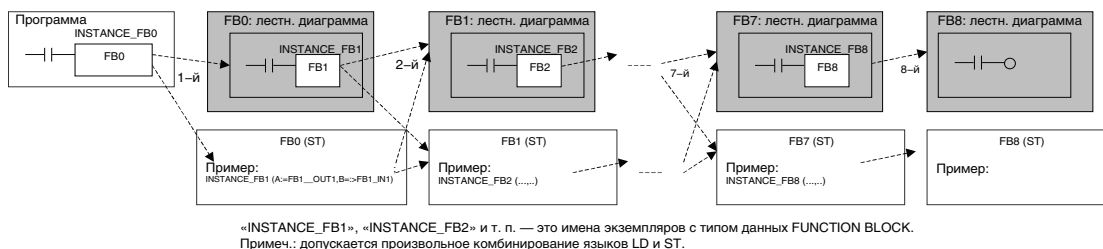
Предупреждение

При выключенном входе переменной EN экземпляр не выполняется. Из этого следует, что при выключенном входе EN не инициализируются команды таймеров и команды с различением фронта. Если в программе функционального блока присутствуют команды таймеров или команды с различением фронта, в качестве входного условия EN следует использовать флаг «всегда ВКЛ» (P_On) и включать входное условие такой команды в программу функционального блока.

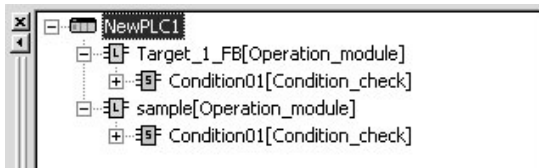
Вложение функциональных блоков

В CX-Programmer версии 6.0 и более поздних версий один функциональный блок может быть вызван из другого функционального блока (то есть поддерживается вложение). До 8 функциональных блоков могут быть вложены друг в друга (включая функциональный блок, вызываемый из программы).

Вложение функциональных блоков, помимо прочего, допускает произвольное комбинирование языков ST и LD. То есть один функциональный блок (вызываемый или вызывающий) может быть написан на языке ST, а другой функциональный блок — на языке LD.



Взаимосвязь между вложенными функциональными блоками может быть отображена в виде иерархического дерева в окне FB Instance Viewer (Окно обзора экземпляров функц. блоков) в CX-Programmer.



Если некоторый функциональный блок содержит вложенные функциональные блоки, при сохранении определения этого функционального блока в файл библиотеки функциональных блоков (.cxs) вместе с ним в этот файл также сохраняются определения всех вложенных в него функциональных блоков.

2-4 Ограничения при программировании

2-4-1 Ограничения при программировании на языке LD

Ниже описаны ограничения, которые необходимо учитывать при создании программы функционального блока на языке релейно-контактных схем.

Команды, не поддерживаемые в определениях функц. блоков

См. руководство *Справочное руководство по командам программируемых контроллеров* (Cat. No. W474)

Ограниченное использование параметра AT (не поддерживаемые области данных)

С помощью параметра AT не могут быть указаны адреса в перечисленных ниже областях памяти.

- Регистры указателей (как с прямой, так и с косвенной адресацией) и регистры данных.

Примечание. Следует вводить непосредственно адрес, не используя параметр AT.

- Косвенные адреса областей DM и EM (как в двоичном, так и в двоично-десятичном формате).

Прямая адресация памяти ввода/вывода в операндах команд

- Значения адресов (а не имена переменных) могут непосредственно вводиться в регистры указателей (с косвенной и прямой адресацией) и регистры данных.

Допускается ввод следующих значений в операнды команд:

Прямая адресация: IR0...IR15; косвенная адресация: ,IR0...,IR15; фиксированное смещение (пример): +5,IR0; смещение DR: DR0,IR0; автоматическое увеличение: ,IR0++; автоматическое уменьшение: -,IR0

- Для всех остальных областей памяти ввода/вывода прямая адресация в операндах команд не поддерживается.

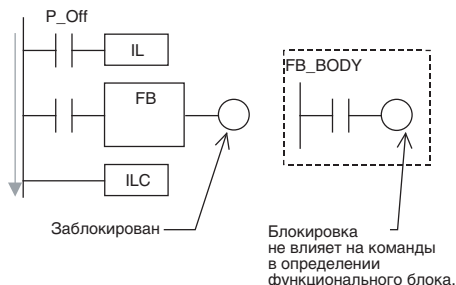
Ограничения для входных, выходных и входных-выходных переменных (не поддерживаемые области данных)

Для входных, выходных и входных-выходных переменных в качестве параметров не могут использоваться адреса следующих областей данных.

- Регистры указателей (как с прямой, так и с косвенной адресацией) и регистры данных.
- Косвенные адреса областей DM и EM (как в двоичном, так и в двоично-десятичном формате).

Ограничения при блокировке

Если вызов функционального блока осуществляется из заблокированного сегмента программы, содержимое определения функционального блока не выполняется. Блокированный функциональный блок ведет себя так же, как заблокированная подпрограмма.

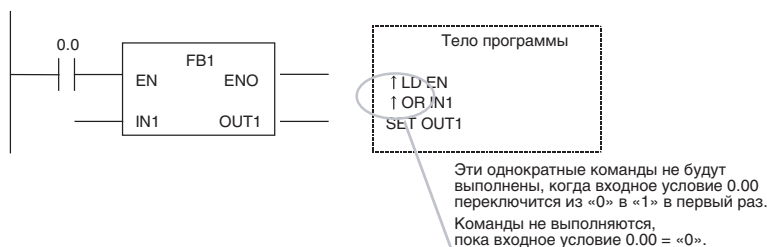


Команды с различием фронта в определениях функц. блоков

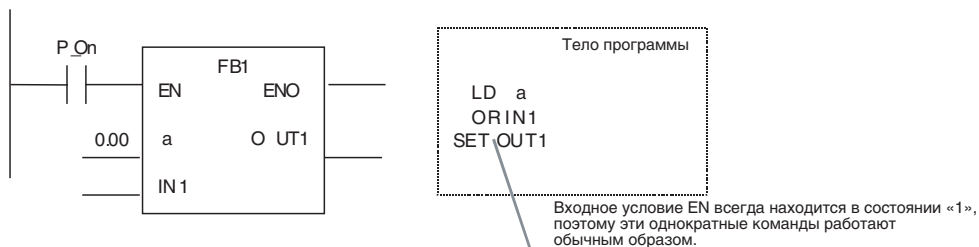
Поскольку при выключенном входе переменной EN экземпляр не выполняется, при использовании команды с различием фронта в определении функционального блока необходимо соблюдать некоторые предосторожности с учетом особенностей выполнения, которые описаны ниже. (Под командами с различием фронта понимаются команды DIFU, DIFD и любые команды с префиксом @ или %.)

- Пока вход переменной EN экземпляра функционального блока остается выключенным, условие выполнения сохраняет свое предыдущее состояние (последнее состояние, в котором оно находилось при включенном входе переменной EN), и команда с различием фронта поэтому не выполняется.
- Когда вход переменной EN экземпляра функционального блока переходит во включенное состояние, текущее состояние условия выполнения сравнивается не с состоянием в последнем цикле, а с состоянием, которое наблюдалось, когда вход переменной EN был включен, поэтому команда с различием фронта работает неправильно. (Однако если вход переменной EN остается включенным, команда с различием фронта начинает работать правильно при поступлении следующего положительного или отрицательного фронта.)

Пример:



Если в программе функционального блока присутствуют команды с различием фронта, в качестве входного условия EN следует обязательно использовать флаг «всегда ВКЛ» (P_On), и входное условие такой команды должно быть включено в программу функционального блока.



- При вводе числового десятичного значения в первый операнд перечисленных ниже команд должен использоваться префикс «#». MILH(517), MILR(518), MILC(519), DIM(631), MSKS(690), MSKR(692), CLI(691), FAL(006), FALS(007), TKON(820), TKOF(821)

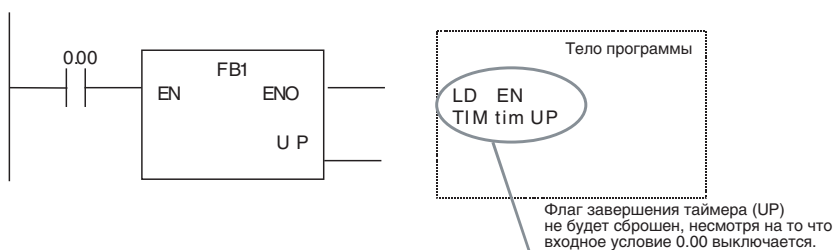
Примечание. Префикс «&» не поддерживается.

- Команды CNR(545), CNRX(547) (СБРОС ТАЙМЕРА/СЧЕТЧИКА) невозможно использовать для сброса одновременно нескольких таймеров и счетчиков внутри функционального блока. И в первом (номер таймера/счетчика 1), и во втором (номер таймера/счетчика 2) операндах должна указываться одна и та же переменная. Разные переменные в первом и втором операндах указаны быть не могут.

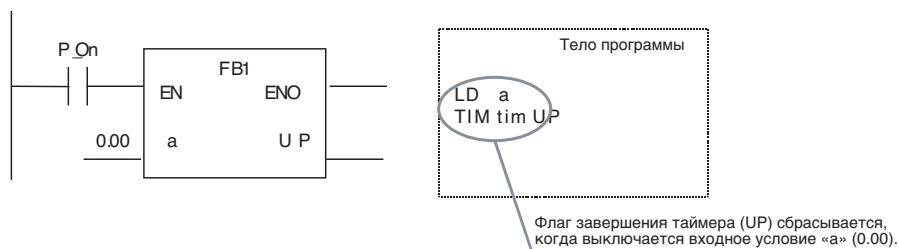
Команды таймеров в определениях функц. блоков

Поскольку при выключенном входе переменной EN экземпляр не выполняется, при использовании команды таймера в определении функционального блока необходимо соблюдать некоторые предосторожности в связи с описанными ниже особенностями выполнения.

Даже если вход переменной EN экземпляра выключается, команда таймера не инициализируется. Следовательно, флаг завершения таймера не будет выключен, даже если после начала работы таймера выключится вход переменной EN.



Если в программе функционального блока присутствуют команды таймеров, в качестве входного условия EN следует обязательно использовать флаг «всегда ВКЛ» (P_On), и входное условие такой команды должно быть включено в программу функционального блока.



- Если в нескольких местах одной программы используется один и тот же экземпляр, содержащий команду таймера, имеет место дублирование таймера с одним и тем же номером.

2-4-2 Ограничения при программировании на языке ST

Ограничения при использовании языка ST в функц. блоках

- Поддерживаются только следующие выражения и операторы:
 - выражения присваивания;
 - выражения выбора (CASE и IF);
 - выражения цикла (FOR, WHILE, REPEAT и EXIT);
 - выражения RETURN;
 - выражения вызова функциональных блоков;
 - арифметические операторы;
 - логические операторы;
 - операторы сравнения;
 - числовые функции;
 - арифметические функции;
 - стандартные функции для работы с текстовыми строками;
 - функции для работы с числовыми значениями в формате текстовых строк;
 - расширенные функции OMRON.
 - Комментарии

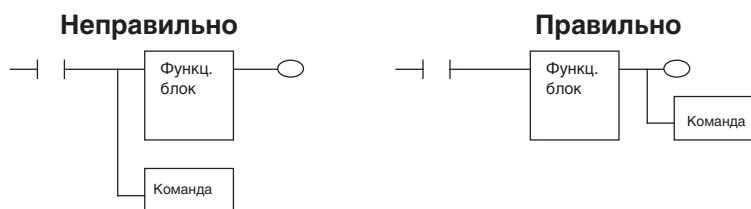
Дополнительную информацию см. в РАЗДЕЛ 5 Характеристики языка структурированного текста в Часть 2: Структурированный текст (ST).

2-4-3 Ограничения при программировании

Ограничения в выборе местоположения экземпляров функциональных блоков

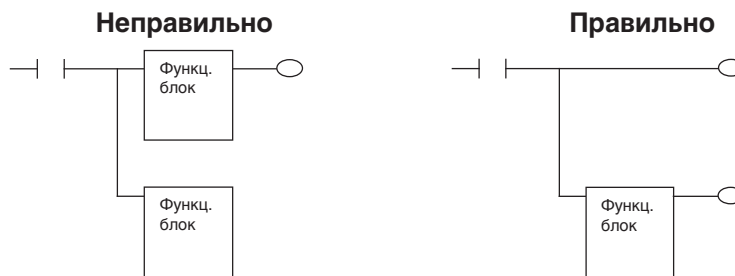
Никаких ответвлений слева от экземпляра

Не допускается наличие ответвлений слева от экземпляра. Ответвления с правой стороны допускаются.



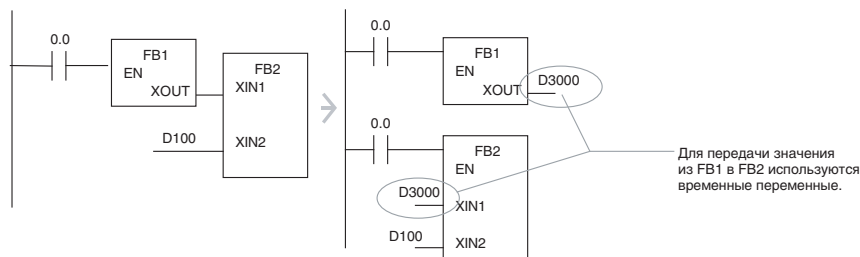
В одной цепи только один экземпляр

В одной цепи лестничной диаграммы может находиться не более одного экземпляра функционального блока.



Никаких соединений между функциональными блоками

Между входом одного функционального блока и выходом другого функционального блока не может быть прямого соединения. Для передачи состояния выполнения с выхода первого функционального блока на вход второго функционального блока следует зарегистрировать переменную.



Загрузка отдельных задач

Индивидуальная загрузка задач, содержащих функциональные блоки, невозможна (считывание возможно).

Отображение на консоли программирования

Если программа пользователя, созданная с помощью CX-Programmer и загруженная в модуль ЦПУ, впоследствии считывается из модуля ЦПУ с помощью консоли программирования, вместо экземпляров на экране отображаются вопросительные знаки. (Имена экземпляров при этом не отображаются.)

Ограничения при онлайн-редактировании

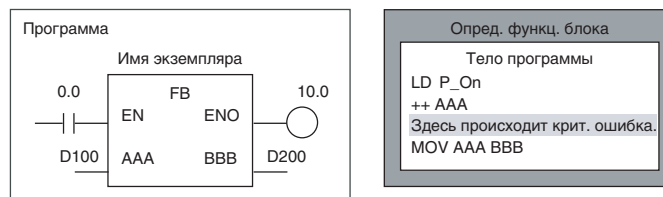
При онлайн-редактировании программы пользователя в модуле ЦПУ невозможно выполнение следующих операций.

- Изменение или удаление определений функциональных блоков (таблиц переменных или алгоритмов).
- Вставка экземпляров или изменение имен экземпляров.

Примечание. Возможны следующие операции: изменение входных/выходных параметров экземпляра, удаление экземпляров и изменение команд за пределами экземпляров.

Ограничения при возникновении ошибок

Если во время выполнения определения функционального блока в модуле ЦПУ возникает критическая ошибка, выполнение прикладной программы прекращается в той точке, где происходит ошибка.



В приведенном выше примере команда MOV AAA BBB не будет выполнена, и выходная переменная D200 сохранит значение, которое в ней содержалось до выполнения функционального блока.

Запрет доступа к областям экземпляров функц. блоков

Для того чтобы экземпляр функционального блока мог быть выполнен, для всех его внутренних, входных, выходных и входных-выходных переменных должны быть выделены адреса в тех или иных областях памяти модуля ЦПУ.

Модули ЦПУ серии CJ2

Область экземпляров функциональных блоков	Исходное значение начального адреса	Исходное значение размера	Допустимые области данных
Без сохранения	H512	896	CIO, WR, HR, DM, EM (см. примеч.)
Сохранение	H1408	128	HR, DM, EM (см. примеч.)
Таймер	T3072	1024	TIM
Счетчик	C3072	1024	CNT

Примечание.

Принудительная установка/сброс возможны, если указаны следующие банки памяти EM:

CJ2H-CPU64(-EIP)/-CPU65(-EIP)	EM, банк 3
CJ2H-CPU66(-EIP)	EM, банки 6...9
CJ2H-CPU67(-EIP)	EM, банки 7...E
CJ2H-CPU68(-EIP)	EM, банки 11...18

Модули ЦПУ серии CS/CJ версии 3.0 или выше и контроллеры NSJ

Область экземпляров функциональных блоков	Исходное значение начального адреса	Исходное значение размера	Допустимые области данных
Без сохранения	H512	896	CIO, WR, HR, DM, EM
Сохранение	H1408	128	HR, DM, EM
Таймер	T3072	1024	TIM
Счетчик	C3072	1024	CNT

Контроллеры движения FQM1

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	5000	5999	1000	CIO, WR, DM
Сохран.	Нет			
Таймеры	T206	T255	50	TIM
Счетчики	C206	C255	50	CNT

Модули ЦПУ серии CP

Область экземпляров функциональных блоков	Исходное значение начального адреса	Исходное значение размера	Допустимые области данных
Без сохранения	H512	896	CI0, WR, HR, DM (см. примеч.)
Сохранение	H1408	128	HR, DM (см. примеч.)
Таймер	T3072	1024	TIM
Счетчик	C3072	1024	CNT

Примечание.

Область DM в ЦПУ CP1L-L

Адрес	CP1L-L
D0000...D9999	Есть
D10000...D31999	Нет
D32000...D32767	Есть

Если прикладная программа содержит команду, которая обращается к адресу памяти в области экземпляра функционального блока, при попытке выполнения любой из следующих операций CX-Programmer выдаст сообщение об ошибке.

- При проверке программы пользователем путем выбора команды **Program – Compile (Программа — Компилировать)** в меню Program (Программа) или команды **Compile All Programs (Компилировать все программы)** в меню PLC (ПЛК).
- При попытке записи программы путем онлайн-редактирования (запись невозможна).

Ограничения в связи с указанием структур данных в качестве параметров при вложении функциональных блоков

При вызове одного функционального блока из другого (т. е. при вложении функциональных блоков) отдельные члены структуры данных не могут указываться в качестве параметров для вложенного функционального блока. Должна быть указана вся структура данных целиком.

Ограниченное применение функции копирования с приращением адресов

При использовании функционального блока в выбранном сегменте программы невозможно использовать функцию копирования с приращением адресов.

2-5 Указания по использованию функциональных блоков

Данный раздел содержит указания и рекомендации по использованию функциональных блоков при работе в программном обеспечении CX-Programmer.

2-5-1 Выбор типа данных для переменной

Целочисленные данные (данные длиной в 1, 2 или 4 слова)

Для выполнения операций над одиночными числовыми значениями длиной в 1, 2 или 4 слова должны использоваться следующие типы данных.

- INT и UINT
- DINT и DINT
- LINT и ULINT

Примечание.

Используйте целые значения со знаком, если это не приводит к выходу используемых чисел за допустимый диапазон.

**Группы слов
(данные длиной в 1, 2
или 4 слова)**

Для выполнения операций над группами значений (не составляющих единое число) длиной в 1, 2 или 4 слова должны использоваться следующие типы данных.

- WORD
- DWORD
- LWORD

Текстовые строки

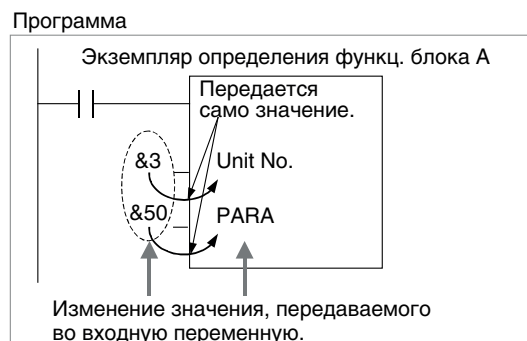
Для выполнения операций над текстовыми строками должен использоваться следующий тип данных.

- STRING

2-5-2 Определение типа переменной (входные, выходные, входные-выходные, внешние и внутренние)

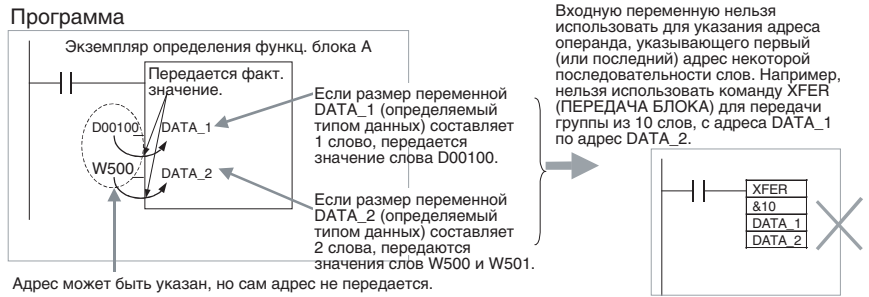
Использование входной переменной для изменения передаваемого значения

Входная переменная используется для того, чтобы при вставке очередного экземпляра некоторого функционального блока в программу в этот экземпляр можно было передать требуемое значение (а не сам адрес).



В отношении входных переменных существуют два следующих ограничения.

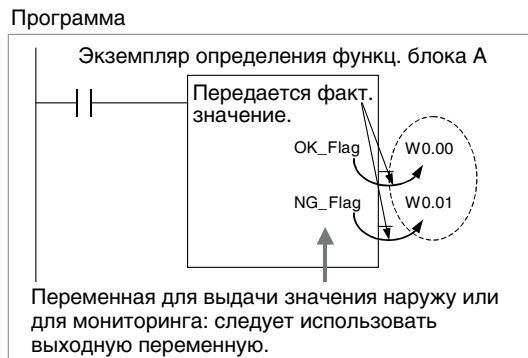
- Во входном параметре может быть указан адрес ячейки памяти, однако во входную переменную функционального блока передается не само значение адреса, а содержащиеся по этому адресу данные, при чем объем передаваемых данных определяется типом данных входной переменной. Из этого следует, что если в операнде некоторой команды программы функционального блока указывается первое или последнее слово последовательности слов, входную переменную в качестве такого операнда использовать нельзя. В этом случае необходимо прибегнуть к одному из следующих вариантов: использовать внутреннюю переменную с заданным параметром AT; указать первый или последний элемент входной-выходной переменной, определенной как массив (и указать во входном параметре адрес первого слова) (возможно в CX-Programmer версии 7.0 и выше); указать первый или последний элемент внутренней переменной, определенной как массив; использовать внешнюю переменную (как описано в разделе 2-5-4 *Определение входных-выходных и внутренних переменных в качестве массива*).



- Значения всех входных параметров передаются во входные переменные функционального блока одновременно, до начала выполнения программы функционального блока (а не одновременно с выполнением команд программы функционального блока). Поэтому, если требуется, чтобы значение параметра передавалось во входную переменную непосредственно во время выполнения команды программы функционального блока, вместо входной переменной следует использовать внутреннюю или внешнюю переменную.

Получение значений из выходных переменных и их мониторинг

Назначение выходной переменной состоит в том, чтобы передать за пределы функционального блока результат выполнения экземпляра функционального блока, вставленного в прикладную программу.

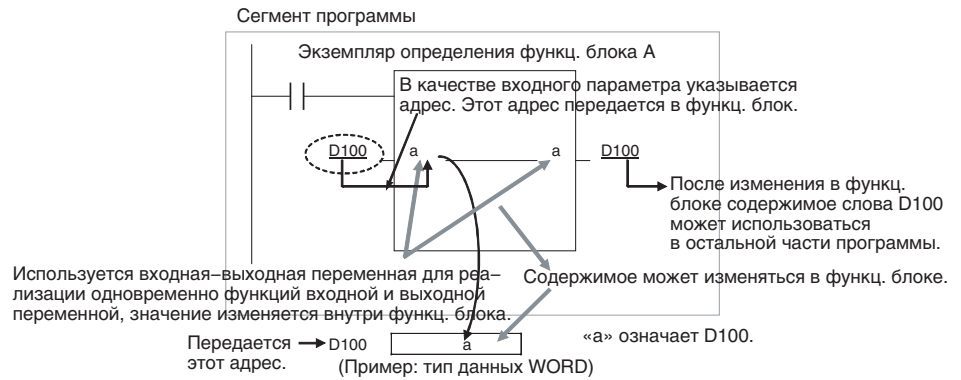


В отношении выходных переменных существуют следующие ограничения.

- Значения, содержащиеся в выходных переменных, передаются в выходные параметры все вместе по завершении выполнения программы функционального блока.

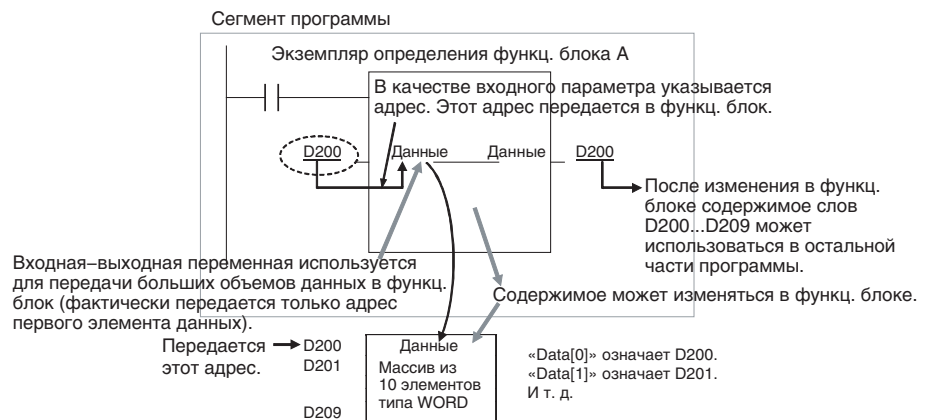
Использование входных-выходных переменных для возврата в выходные параметры результатов обработки значений, переданных во входные параметры

Входная-выходная переменная совмещает в себе функции входного и выходного параметров. В целом использование входной-выходной переменной внутри функционального блока ничем не отличается от использования других переменных, за исключением того, что при ее обработке в функциональный блок передается адрес, заданный в качестве параметра входной-выходной переменной.



Использование входных-выходных переменных-массивов для передачи больших объемов данных

В отличие от входных и выходных переменных, входная-выходная переменная может быть определена как массив. В случае определения входной-выходной переменной в качестве массива внутри функционального блока может использоваться группа адресов указанного размера, начинающаяся с адреса, который задан в качестве входного параметра для этой входной-выходной переменной. Входные-выходные переменные, сконфигурированные как массивы, позволяют передавать в функциональный блок большие объемы данных.



Внешние переменные: флаги условий, тактовые импульсы, биты вспомогательной области, глобальные символы программы.

Флаги условий (напр., флаг «Всегда ВКЛ» или флаг «Равно»), тактовые импульсы (напр., бит тактовых импульсов с периодом 1,0 с), системные биты вспомогательной области (напр., флаг первого цикла), а также используемые в программе глобальные символы — все это внешние переменные, определенные в системе.

Внутренние переменные: автоматически распределяемые переменные и переменные, требующие настройки параметра АТ

Переменные, которые не определены как входные, выходные, входные-выходные или внешние переменные, являются внутренними переменными. Внутренние переменные, в свою очередь, подразделяются на переменные с автоматически назначаемыми («внутренними») адресами и переменные, требующие указания конкретного адреса с помощью параметра АТ (напр., адреса входов/выходов или адреса, зарезервированные за специальными модулями ввода/вывода). К переменным, которые могут быть определены как массив, относятся входные-выходные переменные и внутренние переменные. Более подробно о ситуациях, требующих настройки параметра АТ или определения массива, можно прочитать в разделе 2-5-3 Использование параметра АТ для внутренних переменных и разделе 2-5-4 Определение входных-выходных и внутренних переменных в качестве массива.

2-5-3 Использование параметра АТ для внутренних переменных

В описанных ниже ситуациях внутренние переменные обязательно должны использоваться с параметром АТ.

- Используются адреса, зарезервированные для базовых модулей ввода/вывода, специальных модулей ввода/вывода или модулей шины ЦПУ, и эти адреса закреплены за глобальными символами, которые невозможно использовать в качестве внешних переменных (например, из-за нестабильности данных, заданных для глобальных символов).

Примечание. При указании регистров указателей для определения пространства адресов специального модуля ввода/вывода для первого адреса выделяемой области памяти должен указываться параметр АТ (подробное описание см. в 2-5-5 *Указание адресов для специальных модулей ввода/вывода*).

- Используются биты вспомогательной области, не зарегистрированные заранее в качестве внешних переменных, и эти биты закреплены за глобальными символами, которые не используются в качестве внешних переменных.
- Для команды SEND(090) указывается первое адресуемое слово удаленного узла, для команды RECV(098) указывается первое исходное слово локального узла.
- В операнде команды указывается первое или последнее слово последовательности слов, при этом в качестве операнда не может быть указана переменная-массив (например, не может быть указано количество элементов массива).

2-5-4 Определение входных-выходных и внутренних переменных в качестве массива

Использование переменных-массивов для указания первого или последнего слова в многословных операндах

Если в операнде команды указывается первое или последнее слово некоторого диапазона слов (см. примеч.), команда работает в соответствии с адресом, заданным с помощью параметра АТ или назначенным автоматически. (Из чего следует, что тип данных и количество элементов переменной-массива не влияют на работу команды.) Обязательно указывайте переменную с параметром АТ или переменную-массив, количество элементов которой совпадает с объемом данных, обрабатываемых командой.

Примечание. Например, при указании первого исходного слова или первого адресуемого слова для команды XFER(070) (ПЕРЕДАЧА БЛОКА), при указании первого исходного слова для команды SEND(090) или при указании управляющих данных для ряда поддерживаемых команд.

Подробные сведения смотрите в разделе 2-6 *Меры предосторожности в отношении команд с операндами, указывающими первое или последнее слово группы слов*. Порядок настройки переменной-массива описан ниже.

Если переменная-массив является входной-выходной переменной, во входном параметре для нее должен быть указан первый адрес обрабатываемой совокупности слов.

Ниже описан порядок действий для внутренних переменных.

- 1,2,3...**
1. Подготовьте внутреннюю переменную-массив с требуемым числом элементов.

Примечание. Проследите, чтобы объем данных, обрабатываемых командой, соответствовал количеству элементов в массиве. Сведения об объемах данных для каждой команды можно найти в разделе 2-7 *Поддерживаемые команды и ограничения в отношении операндов*.

2. Используя команду MOV в программе функционального блока, запишите требуемое значение в каждый элемент массива.
3. Укажите для операнда первый (или последний) элемент переменной-массива. Благодаря этому станет возможным указание первого (или последнего) адреса диапазона слов.

Ниже приведено несколько примеров.

Оперирование одной строкой данных, занимающей несколько слов

В этом примере используется массив, который содержит имя каталога и имя файла (операнд S2) для команды FREAD.

- Таблица переменных
Входная-выходная или внутренняя переменная, тип данных = WORD, массив из 10 элементов, имена переменных = filename[0]...filename[9]
- Подготовка данных и операции внутри функционального блока
 - Входные-выходные переменные:
Задайте во входном параметре адрес первого слова данных (пример: D100). Данные (#5C31, #3233, #0000 и др.) записываются в слова D100...D109 заранее в основной программе пользователя.

FREAD (опущено) (опущено) read_num[0] (опущено) ← В операнде команды указывается первый элемент массива.

- Внутренние переменные:
Задайте значения элементов массива с помощью программы функционального блока на языке LD.

```
MOV #5C31 file_name[0] }
MOV #3233 file_name[1] } ← Задаются значения каждого элемента массива.
MOV #0000 file_name[2] }
FREAD (опущено) (опущено) file_name[0] (опущено) ← В операнде команды указывается первый элемент массива.
```

Оперирование управляющими данными, занимающими несколько слов

В этом примере используется массив, который содержит значение количества слов и первое исходное слово (операнд S1) для команды FREAD.

- Таблица переменных
Входная-выходная или внутренняя переменная, тип данных = DINT, массив из 3 элементов, имена переменных = read_num[0]...read_num[9]
- Подготовка данных и операции внутри функционального блока
 - Входные-выходные переменные:
Задайте во входном параметре адрес первого слова данных (пример: D200). Значения записываются в слова D200...D205 заранее в основной программе пользователя.

FREAD (опущено) read_num[0] (опущено) (опущено) ← В операнде команды указывается первый элемент массива.

- Внутренние переменные:
Задайте значения элементов массива с помощью программы функционального блока на языке LD.
- Программа на языке LD

```
MOVL &100 read_num[0] (No._of_words) }
MOVL &0 read_num[1] (1st_source_word) } ← Задается значение каждого элемента массива.
```

```
FREAD (опущено) read_num[0] (опущено) (опущено) ← В операнде команды указывается первый элемент массива.
```

Оперирование блоком прочитанных данных, занимающим несколько слов

Необходимо заранее определить допустимый объем читаемых данных и подготовить массив, способный вместить максимальное количество данных. В этом примере используется массив, в который записываются данные, прочитанные с помощью команды FREAD (операнд D).

- Таблица переменных
Входная-выходная или внутренняя переменная, тип данных = WORD, массив из 100 элементов, имена переменных = read_data[0]...read_data[99]
- Подготовка данных и операции внутри функционального блока
 - Входные-выходные переменные:
Задайте во входном параметре адрес первого слова прочитанных данных (пример: D200).

```
FREAD (опущено) (опущено) (опущено) read_data[0]
```

- Внутренние переменные:

```
FREAD (опущено) (опущено) (опущено) read_data[0]
```

Деление с использованием целочисленных переменных-массивов (только для языка LD)

Для сохранения результата выполнения команды ДЕЛИТЬ ДВОИЧНЫЕ СЛОВА СО ЗНАКОМ (/) в лестничной диаграмме можно использовать массив, состоящий из двух элементов. Выполненная команда возвращает значения D (частное) и D+1 (остаток). Данный способ можно использовать для обращения к величине остатка, получаемого в результате операции деления в программе на языке LD.

Примечание.

Если программа создается на языке ST, использовать массив для получения результата операции деления не требуется. Язык ST не позволяет напрямую вычислить значение остатка. Для вычисления остатка необходимо использовать следующую формулу:

Остаток = делимое - (делитель × частное)

2-5-5 Указание адресов для специальных модулей ввода/вывода

Для указания адресов, отведенных для специальных модулей ввода/вывода, используются регистры указателей IR0...IR15 (косвенная адресация с постоянным смещением адреса), при этом за основу принимается значение номера модуля, которое вводится в определение функционального блока в качестве входного параметра (примеры приведены ниже).

Примечание.

Подробную информацию об использовании регистров указателей в функциональных блоках см. в разделе 2-5-6 *Использование регистров указателей*.

Примеры

Пример 1: указание области CIO внутри функционального блока (то же для области DM)

Специальные модули ввода/вывода

Переменные: номер модуля используется в качестве входной переменной, во внутренней переменной с помощью параметра AT указывается первый адрес используемой области памяти: CIO 2000.

Программы: соблюдается порядок действий, представленный ниже.

1,2,3...

1. Номер модуля (входная переменная) умножается на &10, создается смещение номера модуля (внутренняя переменная с типом данных DINT).
2. С помощью команды MOVR(560) (ПЕРЕДАТЬ В РЕГИСТР) в регистр указателя (напр., IR0) записывается адрес физической памяти ввода/вывода, соответствующий первому адресу используемой области CIO (внутренняя переменная, AT = CIO 2000).
3. К адресу физической памяти ввода/вывода в регистре указателя (напр., IR0) добавляется смещение номера модуля.

Пример 2: указание адресуемого бита в области CIO (напр., CIO слова n+a, бит b)

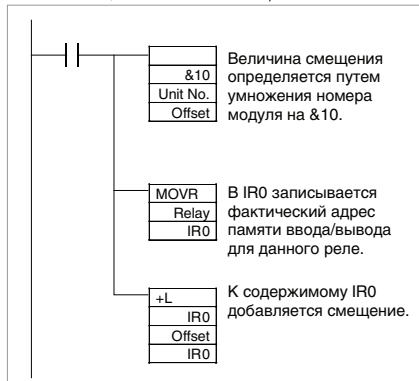
Программы: используется один из следующих способов.

- Адреса слов: используется косвенная адресация и указывается константа для смещения содержимого регистра указателя (напр., +a,IR0).
- Адреса битов: используется команда, допускающая указание адреса бита в пределах слова (напр., &b во втором операнде команды SETB (при записи) и команды TST (при чтении)).

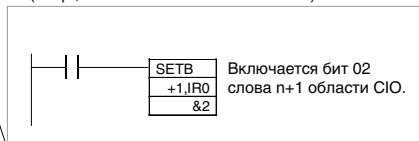
Пример: специальные модули ввода/вывода



1) Укажите n — первое слово области CIO (n = CIO 2000 + номер модуля x 10)
 Используемые константы:
 Номер модуля (Unit No.) (входная переменная, тип данных INT)
 Смещение (Offset) (внутренняя переменная, тип данных DINT)
 Реле (Relay) (внутренняя переменная, тип данных WORD, массив из 400 элементов, значение AT = 2000)



2) Укажите адресуемый бит в области CIO (напр., бит 02 слова n+1 области CIO)

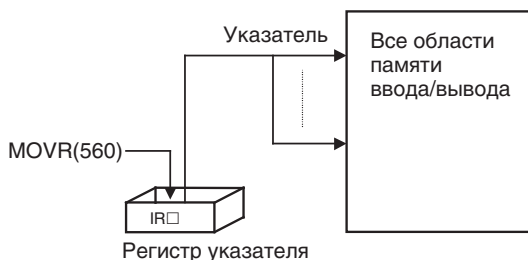


2-5-6 Использование регистров указателей

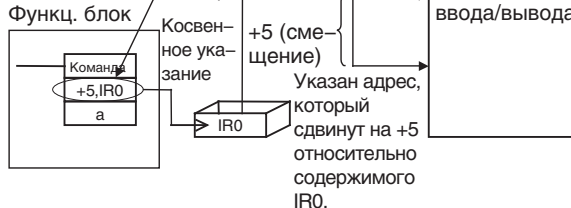
Регистры указателей IR0...IR15 служат для указания физических адресов памяти ввода/вывода. Регистры указателей могут использоваться внутри функциональных блоков для прямого указания адресов. В этом случае вместо имен переменных вводятся названия регистров IR0...IR15 (прямое обращение к регистру: IR0...IR15; косвенное обращение к памяти посредством регистра указателя: ,IR0...,IR15).

Примечание.

Записав в регистр указателя адрес физической ячейки памяти ввода/вывода с помощью команды MOVR(560), этот адрес далее можно косвенно указывать в командах общего назначения с помощью данного регистра указателя. За счет этого возможно динамическое обращение к любой области памяти ввода/вывода.



Пример: вместо имени переменной указывается +5,IR0 (в качестве смещения указывается константа).

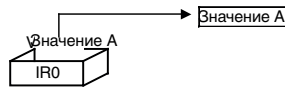


Примечание

- (1) Если какой-либо из регистров указателей (IR0...IR15) будет использован одновременно в нескольких функциональных блоках или одновременно в функциональном блоке и основной программе, программы будут выполняться неправильно, мешая работе друг друга. Указанной проблемы при использовании регистров указателей (IR0...IR15) можно избежать следующим образом: предусмотреть в программе сохранение текущего значения регистра указателя непосредственно перед запуском функционального блока (или перед использованием регистра указателя) и возврат прежнего сохраненного значения в регистр указателя по завершении работы функционального блока (или после использования регистра указателя).

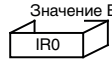
Пример. Запуск функц. блока (или до использования регистра указателя):

1. Сохранение содержимого IR (напр., A).



Внутри функционального блока:

2. Использование регистра IR.



При запуске функц. блока (или до использования регистра указателя):

3. Возвращение в IR прежнего сохраненного значения (напр., A).

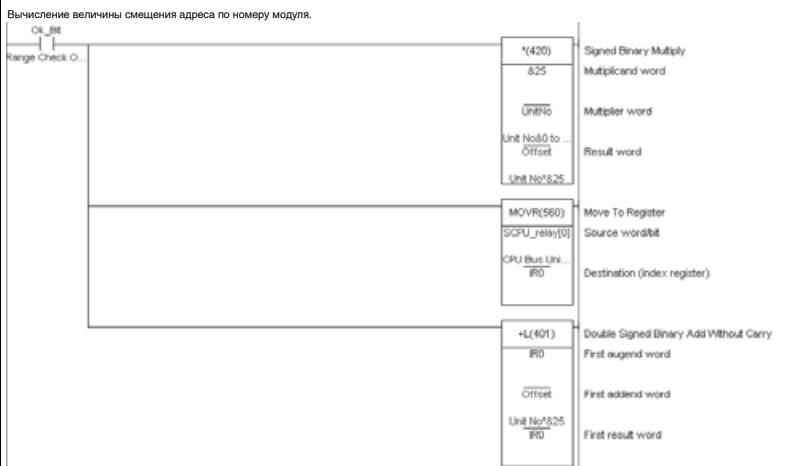


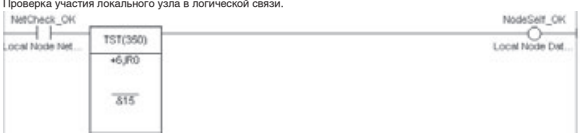
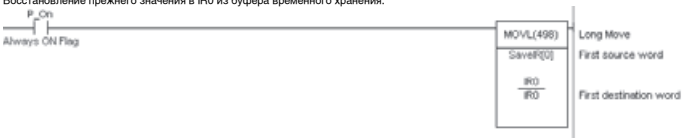
(2) Перед использованием регистра указателя в него обязательно должно быть записано требуемое значение. При использовании регистров с незадаанными значениями программа будет работать непредсказуемым образом.

Примеры использования

Ниже приведено несколько примеров применения регистров указателей IR0...IR15 в функциональных блоках.

Пример	Подробное описание
<p>Сохранение содержимого регистра указателя перед использованием регистра указателя</p> <p>Сохранение содержимого IR0 в буфер для временного хранения.</p>	<p>В случае использования регистра указателя в программе функционального блока предусматривается сохранение текущего значения регистра указателя при запуске функционального блока (или перед использованием регистра указателя), для того чтобы по завершении работы функционального блока (или после использования регистра указателя) в регистр указателя можно было вернуть его первоначальное значение.</p> <p>Пример: сохранение содержимого регистра указателя IR0 путем его записи в переменную <i>SaveIR[0]</i> (внутренняя переменная, тип данных DINT, 1 элемент массива).</p>

Пример	Подробное описание
<p>Использование регистров указателей</p> <p>1) Помещение значения в регистр указателя (запись физического адреса ввода/вывода, соответствующего первому слову области CIO n).</p> <p>Вычисление величины смещения адреса по номеру модуля.</p> 	<p>Пример: в регистр IR0 записывается адрес физической памяти ввода/вывода, соответствующий первому слову области CIO, выделенной для модуля шины ЦПУ. Адрес первого слова области CIO определяется по формуле: $1500 + \text{номер модуля} \times 25$, где номер модуля — это номер модуля шины ЦПУ (&0...&15), который передается в функциональный блок в качестве входного параметра.</p> <p>Выполнение:</p> <p>Предполагается, что номера модулей &0...&15 уже введены (вне функционального блока) в переменные <i>UnitNo</i> (входные переменные, тип данных INT).</p> <ol style="list-style-type: none"> 1. Значение <i>UnitNo</i> умножается на &25, результат записывается в переменную <i>Offset</i> (внутренняя переменная, тип данных DINT). 2. В регистр указателя IR0 записывается адрес физической памяти ввода/вывода для переменной <i>SCPU_Relay</i> (внутренняя переменная, тип данных WORD (если требуется, ее можно определить как массив с 400 элементами (см. примеч.), параметр AT = 1500)). <p>Примечание.</p> <p>Определение переменной <i>SCPU_Relay</i> в качестве массива позволит указать адрес CIO $1500 + (UnitNo \times \&25) + 2$, например, в виде <i>SCPU_relay [2]</i>. Это также относится к примеру 2 ниже.</p> <ol style="list-style-type: none"> 3. К адресу физической памяти ввода/вывода регистра указателя IR0 добавляется значение переменной <i>Offset</i> (переменная $UnitNo \times \&25$).

Пример	Подробное описание
<p>2) Указание константы для смещения содержимого регистра указателя (указание бита в диапазоне от CIO n+0 до n+24)</p> 	<p>Выполняется описанный выше шаг 1 для записи адреса физической памяти ввода/вывода слова CIO 1500 + (UnitNo × &25) в регистр указателя IR0. Адрес слова, таким образом, указывается путем смещения содержимого IR0 на фиксированную величину.</p> <p>Например, запись «+2,IR0» обозначает следующий адрес: CIO 1500 + (UnitNo × &25) + 2.</p> <p>Примечание. Адрес CIO 1500 + (UnitNo × &25) + 2 также может быть указан как <i>SCPU_relay [2]</i>, если переменная <i>SCPU_relay</i> будет определена как массив.</p> <p>Для указания адресов битов следует использовать команды, допускающие непосредственное указание адресов битов в пределах слов (например, второй операнд команд TST(350/351)/SETB(532)).</p> <p>Пример: переменная <i>NodeSelf_OK</i> включается (т. е. переходит в состояние «1»), если включена переменная <i>NetCheck_OK</i> (внутренняя переменная, тип данных BOOL) и включен бит 15 слова по адресу: IR0 (CIO 1500 + UnitNo × &25) +6, где «+6» — величина смещения адреса.</p>
<p>Возврат прежнего значения в регистр указателя</p> 	<p>По завершении выполнения этого функционального блока (или после использования регистра указателя) в регистр указателя возвращается его первоначальное содержимое.</p> <p>Пример: в регистр IR0 возвращается его прежнее значение, сохраненное в переменную <i>SaveR[0]</i> перед запуском функционального блока (или перед использованием регистра указателя).</p>

2-6 Меры предосторожности в отношении команд с операндами, указывающими первое или последнее слово группы слов

Ниже поясняются меры предосторожности, которые должны соблюдаться в том случае, когда в программе функционального блока, создаваемой на языке релейно-контактных схем, используется команда, в операнде которой указывается не одно слово данных, а первое или последнее слово некоторой совокупности слов данных.

Если операнд команды ссылается на первое или последнее слово некоторой последовательности слов, команда выполняется в соответствии с адресом, заданным с помощью параметра AT или назначенным автоматически (либо в соответствии с определением внешней переменной). Из этого следует, что тип данных переменной и количество элементов массива не влияют на работу команды. Следует использовать либо переменную с параметром AT, либо переменную-массив, количество элементов которой совпадает с объемом данных, обрабатываемых командой.

Примечание. В том случае когда в операнде команды должно быть указано первое или последнее слово группы слов, следует обязательно использовать переменную с параметром AT (или внешнюю переменную) либо переменную, размер которой совпадает с объемом данных, обрабатываемых командой. Необходимо соблюдать следующие указания.

- 1,2,3...**
1. Если будет указана переменная, не являющаяся массивом, отличающаяся по размеру и не имеющая параметра AT, CX-Programmer выдаст сообщение об ошибке при компиляции программы.
 2. При указании переменной-массива должны соблюдаться следующие меры предосторожности.

Фиксированный объем данных, обрабатываемых командой

Количество элементов массива должно точно соответствовать объему данных, обрабатываемых командой. В противном случае CX-Programmer выдаст сообщение об ошибке при компиляции.

Не фиксированный объем данных, обрабатываемых командой

Количество элементов массива должно быть равно или должно превышать объем данных, указанный другим операндом.

Размер указывается другим операндом: константа

CX-Programmer сигнализирует ошибку при компиляции.

Размер указывается другим операндом: переменная

Даже если количество элементов массива не соответствует объему данных, указанному другим операндом (переменной), CX-Programmer не выдает сообщение об ошибке при компиляции (отображается лишь предупреждение).

В частности, если количество элементов массива окажется меньше размера (количества элементов данных), указанного другим операндом (например, если размер операнда команды равен 16, а в фактической таблице переменных зарегистрировано 10 элементов данных), команда произведет чтение или запись того количества элементов данных области памяти, которое указано ее операндом. (Так, в описанном выше случае будут прочитаны или записаны не только 10 элементов данных, которые зарегистрированы в фактической таблице переменных, но и 6 слов, расположенных за ними.) Если эта же область используются другими командами (в том числе внутренними переменными функциональных блоков), система может работать непредсказуемым образом, что может стать причиной несчастного случая с тяжелыми последствиями.

В операнде команды, указывающем первый (или последний) адрес некоторого диапазона слов, нельзя использовать переменную, размер которой расходится с объемом (количеством элементов) данных, обрабатываемым этой командой. Если переменная не является массивом, ее тип данных должен соответствовать объему данных команды. Если переменная является массивом, количество элементов массива должно соответствовать объему данных команды. При несоблюдении этих требований будут возникать ошибки, описанные ниже.

Если для *W* указано значение &20, для *S* указана переменная-массив *a* с 10 элементами типа WORD, а для *D* указана переменная-массив *b* с 10 элементами типа WORD:

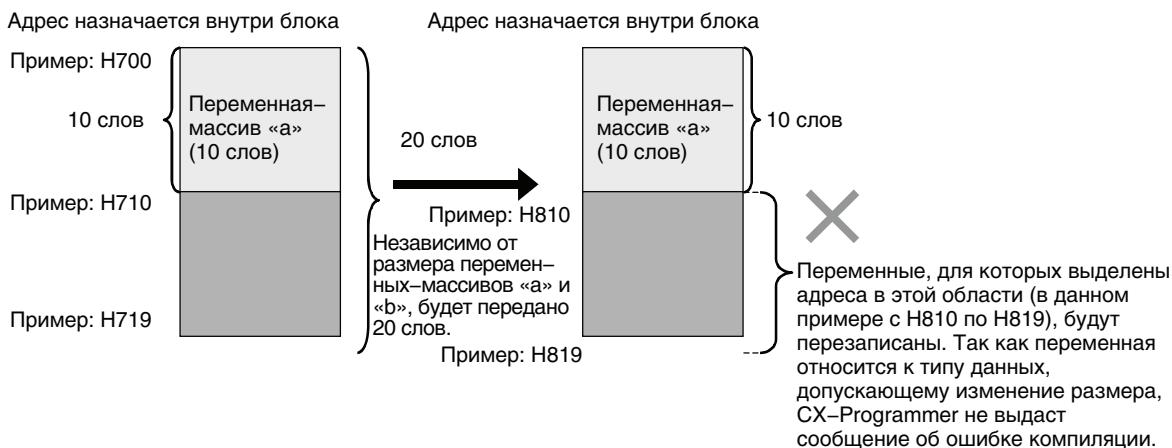
XFER &20 a[0] b[0]

Несмотря на то что переменные-массивы *a*[0] и *b*[0] оба содержат по 10 слов, команда XFER(070) выполнит передачу содержимого 20 слов в соответствии со значением операнда *W*. Другими словами, команда XFER(070) произведет чтение/запись не только в отношении выделенного количества элементов массива, но и в отношении еще 10 слов памяти ввода-вывода, расположенных по порядку за этими элементами (см. рисунок ниже).

Например, если для переменной *a*[10 элементов] автоматически выделены слова в диапазоне H700...H709, а для переменной *b*[10 элементов] автоматически выделены слова в диапазоне H800...H809, команда XFER(070) передаст содержимое слов H700...H719 в слова H800...H819. Если при этом для другой переменной (напр., *c*) были автоматически выделены слова в диапазоне H810...H819, содержимое этих слов будет перезаписано непредусмотренными значениями, нормальный ход выполнения программы будет нарушен. Если требуется передать содержимое 20 слов данных, для обеих переменных массивов *a* и *b* должно быть указано количество элементов, равное 20.

XFER &20 a[0] b[0]

Переменные *a* и *b* являются массивами, содержащими по 10 элементов с типом данных WORD: чтобы передать 20 слов, для обоих массивов должно быть указано количество элементов, равное 20.



Размер указывается другим операндом: переменная

Даже если количество элементов массива не совпадает с объемом (т. е. количеством элементов данных, обрабатываемых командой), который указан другим операндом (переменной), CX-Programmer не выдаст сообщение об ошибке при компиляции программы. Команда будет выполнена в соответствии с объемом, который указан операндом, независимо от количества элементов в переменной-массиве.

В частности, если количество элементов массива окажется меньше размера (количества элементов данных), указанного другим операндом (переменной), при выполнении команды может нарушиться содержимое других переменных, что может привести к непредсказуемым последствиям.

2-7 Поддерживаемые команды и ограничения в отношении операндов

Поддерживаемые команды

Информацию о возможности использования той или иной команды программирования, поддерживаемой модулями ЦПУ серии CS/CJ/NSJ, модулями ЦПУ серии CP и модулями серии FQM1, можно найти в справке по командам в программе CX-Programmer и в справочном руководстве по командам программирования для используемой модели ПЛК.

Ограничения в отношении операндов

- Перед применением какой-либо команды, в операндах которой указывается первое или последнее слово последовательности слов, обязательно ознакомьтесь с информацией в *Разделе 2-6 Меры предосторожности в отношении команд с операндами, указывающими первое или последнее слово группы слов.*
- Если в операнде команды указывается первое или последнее слово последовательности слов, для передачи данных в переменные или получения данных из переменных невозможно использовать входные параметры. Следует использовать параметр AT или входную-выходную либо внутреннюю переменную, являющуюся массивом.
 - Если используется входная/выходная переменная-массив, во входном параметре следует указать первое слово.
 - Если используется внутренняя переменная-массив, следует подготовить переменную-массив с требуемым числом элементов, записать необходимые значения в элементы массива с помощью программы функционального блока, а в операнде указать первый или последний элемент переменной массива.
- Используйте переменную с заданным параметром AT для операндов, указывающих адрес памяти ввода-вывода, расположенный на удаленном узле в сети.

2-8 Характеристики функциональных блоков в модулях ЦПУ

Приведенная ниже таблица содержит технические характеристики функциональных блоков, которые используются в модулях ЦПУ серии CS/CJ и серии CP. Информацию о других характеристиках можно найти в соответствующих руководствах по эксплуатации модулей ЦПУ серии CS/CJ и серии CP.

2-8-1 Технические характеристики

Модули ЦПУ CJ2H

Параметр		Характеристики				
Модель		CJ2H-CPU68 (-EIP)	CJ2H-CPU67 (-EIP)	CJ2H-CPU66 (-EIP)	CJ2H-CPU65 (-EIP)	CJ2H-CPU64 (-EIP)
Входы/выходы		2560				
Объем программы (кол-во шагов)		400K	250K	150K	100K	50K
Память данных		32K слов (для областей DM и EM возможна побитовая адресация)				
Расширенная память данных		32K слов × 25 банков E0_00000...E18_32767	32K слов × 15 банков E0_00000...EE_32767	32K слов × 10 банков E0_00000...E9_32767	32K слов × 4 банка E0_00000...E3_32767	32K слов × 4 банка E0_00000...E3_32767
	Область, допускающая принуд. уст./ сброс	EM 11...EM 18	EM 7...EM E	EM 6...EM9	EM3	EM3
Функциональные блоки	Максимальное количество определений	1024				
	Максимальное количество экземпляров	2048				
Области исходного кода/ комментариев/ указатели программ	Таблицы символов/ комментариев/ указатели программы	3,5 Мбайт (см. примеч.)				

Примечание. Объем памяти, используемой для хранения данных, не ограничен. Однако суммарный объем памяти областей исходного кода и комментариев составляет 3,5 Мбайт.

Модули ЦПУ CJ2M

Параметр		Характеристики				
Модель		CJ2M-CPU11/31	CJ2M-CPU12/32	CJ2M-CPU13/33	CJ2M-CPU14/34	CJ2M-CPU15/35
Входы/выходы		2560				
Объем программы (кол-во шагов)		5 K	10 K	20 K	30 K	60 K
Память данных		32K слов				
Расширенная память данных		32K слов × 1 банк E0_00000...E0_32767			32K слов × 4 банка E0_00000...E3_32767	

Параметр		Характеристики	
Функциональные блоки	Максимальное количество определений	256	2048
	Максимальное количество экземпляров	256	2048
Области исходного кода/комментариев	Таблицы символов/комментарии/указатели программы	1 Мбайт (см. примеч.)	

Примечание. Объем памяти, используемой для хранения данных, не ограничен. Однако суммарный объем памяти областей исходного кода и комментариев составляет 1 Мбайт.

Область, используемая для функциональных блоков

Область, используемая для функциональных блоков в модулях ЦПУ CJ2, зависит от модели модуля ЦПУ, что отражено в следующей таблице. В модулях ЦПУ CJ2M для функциональных блоков предусмотрена специальная область памяти, называемая областью программ функциональных блоков. В модулях ЦПУ CJ2H такая область отсутствует.

Модуль ЦПУ	Модель	Область, используемая для функц. блоков
CJ2H	CJ2H-CPU6□-EIP CJ2H-CPU6□	Для функц. блоков используется область памяти программ пользователя.
CJ2M	CJ2M-CPU3□ CJ2M-CPU1□	Для функц. блоков используется область памяти программ функциональных блоков. Если объема области программ функц. блоков недостаточно, используется область программ пользователя.

Модули ЦПУ CS1-H

Параметр		Характеристики								
Модель		CS1H-CPU67H	CS1H-CPU66H	CS1H-CPU65H	CS1H-CPU64H	CS1H-CPU63H	CS1G-CPU45H	CS1G-CPU44H	CS1G-CPU43H	CS1G-CPU42H
Входы/выходы		5120					1280	960		
Объем программы (кол-во шагов)		250K	120K	60K	30K	20K	60K	30K	20K	10K
Память данных		32K слов								
Расширенная память данных		32K слов × 13 банков E0_00000... E6_32767	32K слов × 7 банков E0_00000... E6_32767	32K слов × 3 банка E0_00000... E2_32767	32K слов × 1 банк E0_00000... E0_32767	32K слов × 3 банка E0_00000... E2_32767	32K слов × 1 банк E0_00000...E0_32767			
Функциональные блоки	Мак. кол-во определений	1024	1024	1024	1024	128	1024	1024	128	128
	Мак. кол-во экземпляров	2048	2048	2048	2048	256	2048	2048	256	256
Модуль памяти комментариев (версия 4.0 или выше).	В сумме для всех файлов (Кбайт)	2048	2048	1280	1280	1280	1280	704	704	704

Параметр		Характеристики								
Внутр. память комментариев (версия 3.0 или выше).	Память программ функц. блоков (Кбайт)	1664	1664	1024	512	512	1024	512	512	512
	Файлы комментариев (Кбайт)	128	128	64	64	64	64	64	64	64
	Файлы указателей программы (Кбайт)	128	128	64	64	64	64	64	64	64
	Таблицы переменных (Кбайт)	128	128	128	64	64	128	64	64	64

Модули ЦПУ CJ1-H

Параметр		Характеристики							
Модель		CJ1H-CPU67H/ CPU67H-R	CJ1H-CPU66H/ CPU66H-R	CJ1H-CPU65H/ CPU65H-R	CPU64H-R	CJ1G-CPU45H	CJ1G-CPU44H	CJ1G-CPU43H	CJ1G-CPU42H
Входы/выходы		2560				1280		960	
Объем программы (кол-во шагов)		250K	120K	60K	30K	60K	30K	20K	10K
Память данных		32K слов							
Расширенная память данных		32K слов × 13 банков E0_00000... E6_32767	32K слов × 7 банков E0_00000... E6_32767	32K слов × 3 банка E0_00000... E2_32767	32K слов × 1 банк E0_00000... E2_32767	32K слов × 3 банка E0_00000... E2_32767	32K слов × 1 банк E0_00000...E0_32767		
Функциональные блоки	Макс. кол-во определений	1024	1024	1024	1024	1024	1024	128	128
	Макс. кол-во экземпляров	2048	2048	2048	2048	2048	2048	256	256
Модуль памяти комментариев (версия 4.0 или выше).	В сумме для всех файлов (Кбайт)	2048	2048	1280	1280	1280	1280	1280	704

Параметр		Характеристики							
Внутр. память комментариев (версия 3.0 или выше).	Память программ функц. блоков (Кбайт)	1664	1664	1024	512	1024	512	512	512
	Файлы комментариев (Кбайт)	128	128	64	64	64	64	64	64
	Файлы указателей программы (Кбайт)	128	128	64	64	64	64	64	64
	Таблицы переменных (Кбайт)	128	128	128	64	128	64	64	64

Модули ЦПУ CJ1M

Параметр	Характеристики					
	Модули с внутренними функциями ввода/вывода			Модули без внутренних функций ввода/вывода		
Модель	CJ1M-CPU23	CJ1M-CPU22	CJ1M-CPU21	CJ1M-CPU13	CJ1M-CPU12	CJ1M-CPU11
Входы/выходы	640	320	160	640	320	160
Объем программы (кол-во шагов)	20К	10К	5К	20К	10К	5К
Количество стоек расширения	Макс. 1	Расширение не поддерживается		Макс. 1	Расширение не поддерживается	
Память данных	32К слов					
Расширенная память данных	Нет					
Время начала выдачи импульсов	46 мкс (без разгона/торможения) 70 мкс (с разгоном/торможением)		63 мкс (без разгона/торможения) 100 мкс (с разгоном/торможением)	---		
Количество запланированных прерываний	2		1	2		1
Выходы ШИМ	2		1	Нет		
Макс. значение номера подпрограммы	1024		256	1024		256
Макс. значение номера перехода в команде JMP	1024		256	1024		256
Внутренние входы	10 точек • 4 входа прерывания (захват импульса) • 2 входа скоростных счетчиков (50 кГц — двухканальные квадратурные или 100 кГц — одноканальные)			---		
Внутренние выходы	6 точек • 2 импульсных выхода (100 кГц) • 2 выхода ШИМ		6 точек • 2 импульсных выхода (100 кГц) • 1 выход ШИМ	---		

Параметр		Характеристики	
		Модули с внутренними функциями ввода/вывода	Модули без внутренних функций ввода/вывода
Функциональные блоки	Макс. кол-во определений	128	
	Макс. кол-во экземпляров	256	
Модуль памяти комментариев (версия 4.0 или выше).	В сумме для всех файлов (Кбайт)	704	
Внутр. память комментариев (версия 3.0 или выше).	Память программ функц. блоков (Кбайт)	256	
	Файлы комментариев (Кбайт)	64	
	Файлы указателей программы (Кбайт)	64	
	Таблицы переменных (Кбайт)	64	

Модули ЦПУ CP1H

Параметр	Модели X	Модели XA	Модели Y
Модель	CP1H-X40DR-A CP1H-X40DT-D CP1H-X40DT1-D	CP1H-XA40DR-A CP1H-XA40DT-D CP1H-XA40DT1-D	CP1H-Y20DT-D
Макс. число входов/выходов	320 точек (40 встроенных точек + 40 точек/стойка расширения x 7 стоек)		300 точек (20 встроенных точек + 40 точек/стойка расширения x 7 стоек)
Объем программы (кол-во шагов)	20К		
Память данных	32К слов		
Количество подключаемых модулей расширения и модулей расширения входов/выходов	7 модулей (модули расширения и модули расширения входов/выходов серии CP)		
Функциональные блоки	Макс. кол-во определений	128	
	Макс. кол-во экземпляров	256	

Параметр		Модели X	Модели XA	Модели Y
Внутренняя память комментариев	Память программ функц. блоков (Кбайт)	256		
	Файлы комментариев (Кбайт)	64		
	Файлы указателей программы (Кбайт)	64		
	Таблицы переменных (Кбайт)	64		

Модули ЦПУ CP1L

Параметр		Модели M			Модели L		
Модель		CP1L-M60D□-□	CP1L-M40D□-□	CP1L-M30D□-□	CP1L-L20D□-□	CP1L-L14D□-□	CP1L-L10D□-□
Макс. число входов/выходов		180 точек (60 встр. точек + 40 точек/стойка расшир. x 3 стоек)	160 точек (40 встр. точек + 40 точек/стойка расшир. x 3 стоек)	150 точек (30 встр. точек + 40 точек/стойка расшир. x 3 стоек)	60 точек (20 встр. точек + 40 точек/стойка расшир. x 1 стойка)	54 точек (14 встр. точек + 40 точек/стойка расшир. x 1 стойка)	10 точек
Объем программы (кол-во шагов)		10K			5K		
Память данных		32K слов (D00000...D32767)			10K слов (D00000...D09999 и D32000...D32767)		
Количество подключаемых модулей расширения и модулей расширения входов/выходов		3 модуля (модули расширения и модули расширения входов/выходов серии CP)			1 модуль (модуль расширения или модуль расширения входов/выходов серии CP)		Нет
Функциональные блоки	Макс. кол-во определений	128					
	Макс. кол-во экземпляров	256					
Внутренняя память комментариев	Память программ функц. блоков (Кбайт)	256					
	Файлы комментариев (Кбайт)	64					
	Файлы указателей программы (Кбайт)	64					
	Таблицы переменных (Кбайт)	64					

Контроллеры серии NSJ

Модель		NSJ5-TQ0□-G5D, NSJ5-SQ0□-G5D, NSJ8-TV0□-G5D, NSJ10-TV0□-G5D, NSJ12-TS0□-G5D,	NSJ5-TQ0□-M3D, NSJ5-SQ0□-M3D, NSJ8-TV0□-M3D
Макс. число входов/выходов		1280	640
Объем программы (кол-во шагов)		60K	20K
Память данных		32K слов	
Расширенная память данных		32K слов × 3 банка E0_00000...E2_32767	Нет
Функциональ- ные блоки	Макс. кол-во определений	1024	128
	Макс. кол-во экземпляров	2048	256
Внутренняя память комментариев	Память программ функц. блоков (Кбайт)	1024	256
	Файлы комментариев (Кбайт)	64	64
	Файлы указателей программы (Кбайт)	64	64
	Таблицы переменных (Кбайт)	128	64

Контроллеры движения FQM1

Параметр		Модуль координирования	Модули управления движением	
Модель		FQM1-CM002	FQM1-MMA22	FQM1-MMP22
Макс. число входов/выходов		344 точки (24 встр. точки + 320 точек в базовых модулях вх./вых.)	20 встр. точек	
Объем программы (кол-во шагов)		10К		
Память данных		32К слов		
Функциональные блоки	Макс. кол-во определений	128		
	Макс. кол-во экземпляров	256		
Внутренняя память комментариев	Память программ функц. блоков (Кбайт)	256		
	Файлы комментариев (Кбайт)	64		
	Файлы указателей программы (Кбайт)	64		
	Таблицы переменных (Кбайт)	64		

2-8-2 Особенности работы команд таймеров

В окне настройки параметров ПЛК предусмотрен параметр *Apply the same spec as T0-2047 to T2048-4095 (Применить к T2048...4095 настройки T0...2047)*. В данном разделе описано влияние этого параметра на работу таймеров.

Параметр выбран

Если данный параметр выбран, все таймеры функционируют одинаковым образом, независимо от номера таймера, что отражено в следующей таблице.

Работа таймеров с номерами T0000...T4095

Обновление	Описание
При выполнении команды	Текущее значение обновляется каждый раз, когда выполняется команда. Если текущее значение = 0, флаг завершения установлен (=1). Если текущее значение не равно 0, флаг завершения сброшен (=0).
После завершения выполнения всех задач	Все текущие значения обновляются один раз в каждом цикле.
Каждые 80 мс	Если длительность цикла превышает 80 мс, все текущие значения обновляются один раз через каждые 80 мс.

Параметр не выбран (по умолчанию)

Если данный параметр не выбран, команды таймеров с номерами T0000...T2047 обновляются не так, как команды таймеров с номерами T2048...T4095, что отражено в двух таблицах ниже. Сказанное справедливо и для модулей ЦПУ, не поддерживающих функциональные блоки. (Подробные сведения см. в описании отдельных команд в *Справочном руководстве по командам серии CS/CJ*.)

Работа таймеров с номерами T0000...T2047

Обновление	Описание
При выполнении команды	Текущее значение обновляется каждый раз, когда выполняется команда. Если текущее значение = 0, флаг завершения установлен (=1). Если текущее значение не равно 0, флаг завершения сброшен (=0).
После завершения выполнения всех задач	Все текущие значения обновляются один раз в каждом цикле.
Каждые 80 мс	Если длительность цикла превышает 80 мс, все текущие значения обновляются один раз через каждые 80 мс.

Работа таймеров с номерами T2048...T4095

Обновление	Описание
При выполнении команды	Текущее значение обновляется каждый раз, когда выполняется команда. Если текущее значение = 0, флаг завершения установлен (=1). Если текущее значение не равно 0, флаг завершения сброшен (=0).
После завершения выполнения всех задач	Текущее значение не обновляется.
Каждые 80 мс	Даже если длительность цикла превышает 80 мс, текущие значения не обновляются.

В случае использования номеров таймеров, назначаемых для переменных функциональных блоков по умолчанию (T3072...T4095), для обеспечения согласованной работы всех таймеров следует выбрать параметр *Apply the same spec as TO-2047 to T2048-4095 (Применить к T2048...4095 настройки TO...2047)*.

2-9 Количество шагов в программах функциональных блоков и время выполнения экземпляра

2-9-1 Количество шагов в программах функциональных блоков

Количество шагов, используемое функц. блоками

При использовании функциональных блоков некоторый объем памяти программ (количество шагов) расходуется для нужд двух следующих компонентов функциональных блоков.

1. Определения функциональных блоков
2. Экземпляры определений функциональных блоков, создаваемые в программах

Таким образом, чем больше экземпляров определений функциональных блоков создается в программе, тем больший объем памяти (количество шагов) используется.

Способ определения количества шагов программы при использовании функциональных блоков

Информация данного раздела относится только к модулям ЦПУ серии CP с версией модуля 1.0 и выше, модулям ЦПУ серии CS/CJ с версией модуля 3.0 и выше, контроллерам NSJ и контроллерам движения FQM1.

Для расчета приблизительного количества шагов программы при создании определений функциональных блоков и вставке их экземпляров в программу пользователя модуля ЦПУ можно воспользоваться следующей формулой.

Количество шагов
 = количество экземпляров × (размер секции вызова m + размер секции обмена вх./вых. параметрами n × количество параметров) + количество шагов команд в функц. блоке p (см. примеч.)

Примечание.

Если в программе пользователя используется несколько экземпляров одного и того же функционального блока, второй, третий и все последующие экземпляры функционального блока не увеличивают значение количества шагов команд в определении функционального блока (p). Поэтому в приведенной выше формуле количество шагов команд функционального блока (p) не умножается на количество экземпляров.

Следующая таблица относится только к модулям ЦПУ серии CP с версией модуля 1.0 и выше, модулям ЦПУ серии CS/CJ с версией модуля 3.0 и выше, контроллерам NSJ и контроллерам движения FQM1.

Содержание			Количество шагов
Да	Секция вызова	---	57 шагов
п	Секция обмена вх./вых. параметрами (В скобках указан тип данных.)	Вх. переменная или вых. переменная размером в 1 бит (BOOL)	6 шагов
		Вх. переменная или вых. переменная размером в 1 слово (INT, UINT или WORD)	6 шагов
		Вх. переменная или вых. переменная размером в 2 слова (DINT, UDINT, DWORD или REAL)	6 шагов
		Вх. переменная или вых. переменная размером в 4 слова (LINT, ULINT, LWORD или LREAL)	18 шагов
		Входные-выходные переменные	6 шагов
р	Количество шагов команд в функц. блоке	Общее количество шагов команд (такое же, как и у обычной программы пользователя) + 27 шагов.	

Пример:

Входные переменные длиной в 1 слово (INT): 5

Выходные переменные длиной в 1 слово (INT): 5

Определение функционального блока: 100 шагов

Количество шагов в 1 экземпляре = $57 + (5 + 5) \times 6$ шагов + 100 шагов + 27 шагов = 244 шагов

Если программа создается на языке ST, действительное количество шагов рассчитать невозможно. Количество шагов команд для каждого определения функционального блока можно найти в окне свойств определения функционального блока.

2-9-2 Время выполнения экземпляра функционального блока

Информация данного раздела относится только к модулям ЦПУ серии CP с версией модуля 1.0 и выше, модулям ЦПУ серии CS/CJ с версией модуля 3.0 и выше, контроллерам NSJ и контроллерам движения FQM1. Для количественной оценки влияния выполнения экземпляра на длительность цикла прикладной программы при создании определений функциональных блоков и вставке их экземпляров в программу пользователя модуля ЦПУ можно воспользоваться следующей формулой.

Влияние экземпляра функц. блока на время цикла
 = время запуска (A)
 + время обмена входными/выходными параметрами (B)
 + время выполнения команд функц. блока (C)

Длительность каждого из этапов A, B и C указана в следующей таблице.

Работа			Модель модуля ЦПУ						
			CJ1H-CPU6□H-R CJ2H-CPU6□(-EIP)	CJ2M-CPU□□	CS1H-CPU6□H CJ1H-CPU6□H	CS1G-CPU4□H CJ1G-CPU4□H NSJ	CJ1M-CPU□□	CP1H-X□□□-□ CP1H-XA□□□-□ CP1H-Y□□□-□	CP1L-M□□□-□ □ CP1L-L□□□-□
A	Время запуска	Время запуска (не включает обмен вх. и вых. параметрами)	3,3 мкс	7,4 мкс	6,8 мкс	8,8 мкс	15,0 мкс	15,0 мкс	320,4 мкс
B	Время обмена входными и выходными параметрами (В скобках указан тип данных.)	Вх. или вых. переменная разм. в 1 бит (BOOL)	0,24 мкс	0,88 мкс	0,4 мкс	0,7 мкс	1,0 мкс	1,0 мкс	59,52 мкс
		Вх. или вых. переменная разм. в 1 слово (INT, UINT, WORD)	0,19 мкс	0,88 мкс	0,3 мкс	0,6 мкс	0,8 мкс	0,8 мкс	13,16 мкс
		Вх. или вых. переменная разм. в 2 слова (DINT, UDINT, DWORD, REAL)	0,19 мкс	1,2 мкс	0,5 мкс	0,8 мкс	1,1 мкс	1,1 мкс	15,08 мкс
		Вх. или вых. переменная разм. в 4 слова (LINT, ULINT, LWORD, LREAL)	0,38 мкс	2,96 мкс	1,0 мкс	1,6 мкс	2,2 мкс	2,2 мкс	30,16 мкс
		Вх.-вых. переменная	0,114 мкс	0,4 мкс	0,4 мкс	0,5 мкс	1,2 мкс	(Не поддержив.)	(Не поддержив.)
C	Время выполнения команд функц. блока	Общее время выполнения команд (такое же, как и у обычной программы пользователя)							

Пример: CJ1H-CPU67H-R

Входные переменные длиной в 1 слово (INT): 3

Выходные переменные длиной в 1 слово (INT): 2

Общее время выполнения команд функц. блока: 10 мкс

Время выполнения 1 экземпляра = 3,3 мкс + (3 + 2) × 0,19 мкс + 10 мкс = 14,25 мкс

Примечание.

Если одно и то же определение функционального блока используется в нескольких местах программы, время выполнения возрастает пропорционально количеству экземпляров.

РАЗДЕЛ 3

Создание функциональных блоков

Данный раздел описывает порядок действий по созданию функциональных блоков в программе CX-Programmer.

3-1	Общий порядок действий	98
3-2	Порядок действий	100
3-2-1	Создание проекта	100
3-2-2	Создание нового определения функционального блока	100
3-2-3	Создание функциональных блоков: подробное описание	104
3-2-4	Создание экземпляров определений функциональных блоков	117
3-2-5	Ввод параметров функционального блока с помощью клавиши «Ввод»	120
3-2-6	Настройка областей экземпляров функциональных блоков	123
3-2-7	Выяснение внутренних адресов переменных	125
3-2-8	Копирование и редактирование определений функциональных блоков	127
3-2-9	Выяснение исходного определения функционального блока по экземпляру	128
3-2-10	Отображение сведений об уровнях вложения и другой информации об экземплярах	128
3-2-11	Отображение используемой памяти	129
3-2-12	Компиляция определений функциональных блоков (проверка программы)	130
3-2-13	Вывод на печать определения функционального блока	131
3-2-14	Защита определений функциональных блоков с помощью пароля	132
3-2-15	Сравнение функциональных блоков	135
3-2-16	Сохранение и повторное использование файлов функциональных блоков	135
3-2-17	Загрузка программ в модуль ЦПУ и их считывание из модуля ЦПУ	136
3-2-18	Мониторинг и отладка функциональных блоков	136
3-2-19	Редактирование определений функциональных блоков в режиме онлайн	145

3-1 Общий порядок действий

В данном разделе описан порядок выполнения следующих операций над функциональными блоками: создание, сохранение в файлы, загрузка в модуль ЦПУ, мониторинг, отладка.

Создание функциональных блоков

Создание проекта	<p>Подробное описание смотрите в разделе 3-2-1 <i>Создание проекта</i>.</p> <p>Создание нового проекта</p> <p>1,2,3...</p> <ol style="list-style-type: none"> 1. Запустите CX-Programmer и выберите команду New (Создать) в меню File (Файл). 2. Выберите модель в поле <i>Device type (Тип устройства)</i>: CS1G-H, CS1H-H, CJ1G-H, CJ1H-H, CJ1M или CP1H, CP1L, NSJ или FQM1-CM (ММА/ММР). <p>Повторное использование существующего проекта CX-Programmer</p> <p>1,2,3...</p> <ol style="list-style-type: none"> 1. Запустите CX-Programmer и откройте существующий файл проекта (.sxp), созданный с помощью программы CX-Programmer версии 4.0 или ниже, выбрав требуемый файл с помощью меню File (Файл). 2. Выберите модель в поле <i>Device type (Тип устройства)</i>: CS1G-H, CS1H-H, CJ1G-H, CJ1H-H, CJ1M или CP1H, CP1L, NSJ или FQM1-CM (ММА/ММР).
Создание определения функц. блока	<p>Подробное описание смотрите в разделе 3-2-2 <i>Создание нового определения функционального блока</i>.</p> <p>1,2,3...</p> <ol style="list-style-type: none"> 1. Выберите <i>Function Blocks (Функциональные блоки)</i> в рабочей области проекта и щелкните правой кнопкой мыши. 2. Выберите пункт Insert Function Block – Ladder (Вставить функциональный блок – LD) или Insert Function Block – Structured Text (Вставить функциональный блок – ST) в контекстном меню.
Определение функционального блока	<p>Подробное описание смотрите в разделе 3-2-3 <i>Создание функциональных блоков: подробное описание</i>.</p> <p>Регистрация переменных до ввода программы на языке LD или ST</p> <p>1,2,3...</p> <ol style="list-style-type: none"> 1. Зарегистрируйте переменные в таблице переменных. 2. Создайте программу на языке LD или ST. <p>Регистрация переменных по мере необходимости в процессе ввода программы на языке LD или ST</p> <p>1,2,3...</p> <ol style="list-style-type: none"> 1. Создайте программу на языке LD или ST. 2. Регистрируйте переменные в таблице переменных, когда это требуется.
Создание экземпляра определения функц. блока	<p>Подробное описание смотрите в разделе 3-2-4 <i>Создание экземпляров определений функциональных блоков</i>.</p> <p>Вставка экземпляров в окне сегмента лестничной диаграммы с последующим вводом имени экземпляра</p> <p>1,2,3...</p> <ol style="list-style-type: none"> 1. Расположите курсор в том месте программы, где требуется создать экземпляр функционального блока, и нажмите клавишу F. 2. Введите имя экземпляра. 3. Выберите определение функционального блока, экземпляр которого требуется создать.

Регистрация имен экземпляров в таблице глобальных символов и выбор имени экземпляра при вставке экземпляра

- 1,2,3...
1. Выберите для переменной в таблице глобальных символов тип данных *Function Block*.
 2. Нажмите клавишу **F** в окне Ladder Section (Сегмент лестничной диаграммы).
 3. Выберите зарегистрированное имя экземпляра в раскрывающемся списке в поле *FB Instance* (Экземпляр функц. блока).

Назначение внешних входов и выходов функциональному блоку

Подробное описание смотрите в разделе 3-2-5 *Ввод параметров функционального блока с помощью клавиши «Ввод»*.

- 1,2,3...
1. Расположите курсор в позиции входной или выходной переменной и нажмите клавишу **P**.
 2. Введите адрес источника для входной переменной или адрес назначения для выходной переменной.

Выбор областей памяти для функц. блока (областей экземпляров)

Подробное описание смотрите в разделе 3-2-6 *Настройка областей экземпляров функциональных блоков*.

- 1,2,3...
1. Выделите экземпляр и выберите команду **Function Block/SFC Memory – Function Block/SFC Memory Allocation** (Память функц. блоков/SFC – Адреса памяти для функц. блоков/SFC) в меню PLC (ПЛК).
 2. Задайте области памяти, которые должны использоваться для функционального блока.

Вывод на печать, сохранение и повторное использование файлов функциональных блоков**Компиляция определения функц. блока и его сохранение в файл библиотеки**

Подробное описание смотрите в 3-2-12 *Компиляция определений функциональных блоков (проверка программы)* и 3-2-16 *Сохранение и повторное использование файлов функциональных блоков*.

- 1,2,3...
1. Компилируйте сохраненный функциональный блок.
 2. Выведите функциональный блок на печать.
 3. Сохраните функциональный блок в виде файла определения функционального блока (.sxf).
 4. Откройте сохраненный файл в другом проекте ПЛК.

Загрузка программы в ПЛК

См. 3-2-17 *Загрузка программ в модуль ЦПУ и их считывание из модуля ЦПУ*.

Мониторинг и отладка функционального блока

См. 3-2-18 *Мониторинг и отладка функциональных блоков*.

3-2 Порядок действий

3-2-1 Создание проекта

Создание новых проектов в CX-Programmer

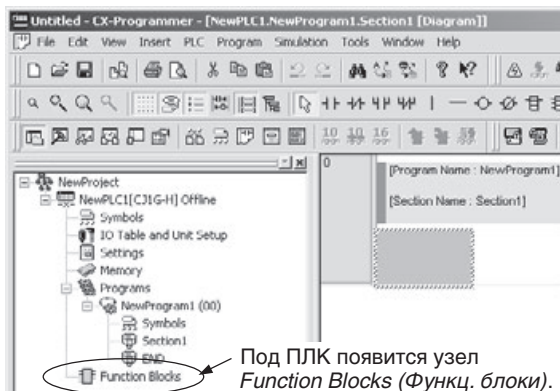
- 1,2,3...
1. Запустите CX-Programmer и выберите команду **New (Создать)** в меню File (Файл).
 2. В диалоговом окне Change PLC (Изменение ПЛК) в поле *Device Type* выберите серию ЦПУ, поддерживающую функциональные блоки. Такие серии перечислены в следующей таблице.

Серия	ЦПУ
CJ2H	CPU68/67/66/65/64/68-EIP/67-EIP/66-EIP/65-EIP/64-EIP
CJ2M	CPU11/12/13/14/15/31/32/33/34/35
CS1G-H	CPU42H/43H/44H/45H
CS1H-H	CPU63H/64H/65H/66H/67H
CJ1G-H	CPU42H/43H/44H/45H
CJ1H-H	CPU65H/66H/67H/64H-R/65H-R/66H-R/67H-R
CJ1M	CPU11/12/13/21/22/23
CP1H	CP1H-XA/X/Y
CP1L	CP1L-M/L
NSJ	G5D (используется для NSJ5-TQ0□-G5D, NSJ5-SQ0□-G5D, NSJ8-TV0□-G5D, NSJ10-TV0□-G5D и NSJ12-TS0□-G5D) M3D (используется для NSJ5-TQ0□-M3D, NSJ5-SQ0□-M3D и NSJ8-TV0□-M3D).
FQM1-CM	FQM1-CM002
FQM1-MMA	FQM1-MMA22
FQM1-MMP	FQM1-MMP22

3. Нажмите кнопку Settings (Настройка) и выберите модель ЦПУ в поле *CPU Type*. Подробную информацию о настройке других параметров можно найти в руководстве *CX-Programmer — Руководство по работе (W446)*.

3-2-2 Создание нового определения функционального блока

- 1,2,3...
1. После того как проект создан, в рабочей области проекта (также называемой «дерево проекта») отображается значок *Function Blocks* (Функциональные блоки).



Создание определений функциональных блоков

2. Значки определений функциональных блоков, вставляемых в программу, отображаются под значком Function Blocks.

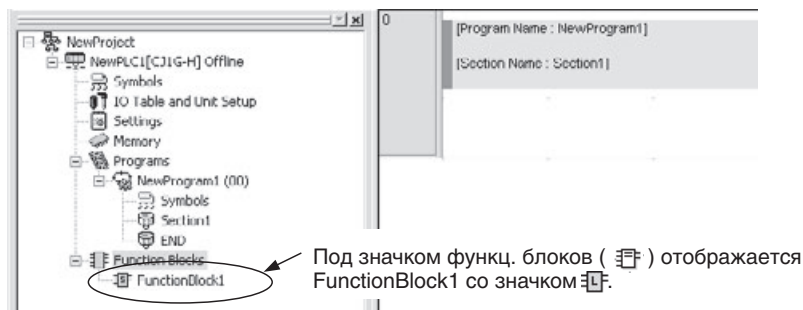
Программа определения функционального блока может создаваться на языке релейно-контактных схем (LD) или на языке структурированного текста (ST).



Создание (Вставка) определений функциональных блоков на языке LD

1. Щелкните правой кнопкой мыши по узлу **Function Blocks (Функциональные блоки)** в рабочей области проекта и выберите **Insert Function Blocks – Ladder (Вставить функциональный блок – LD)** в контекстном меню. (Или выберите пункт **Function Block – Ladder (Функц. блок – LD)** в меню Insert (Вставка).)

Создание (Вставка) определений функциональных блоков на языке ST

1. Щелкните правой кнопкой мыши по узлу **Function Blocks (Функциональные блоки)** в рабочей области проекта и выберите **Insert Function Blocks – Structured Text (Вставить функциональный блок – ST)** в контекстном меню. (Или выберите пункт **Function Block – Structured Text (Функц. блок – ST)** в меню Insert (Вставка).)

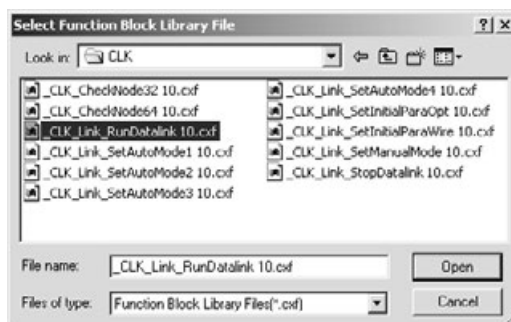


2. Функциональный блок под названием FunctionBlock1 будет автоматически вставлен либо со значком  — для языка LD (по умолчанию), либо со значком  — для языка ST. Этот значок содержит определение вновь созданного (или вставленного) функционального блока.
3. При создании определения функционального блока ему автоматически присваивается имя FunctionBlock□, где □ — порядковый номер. Это имя может быть изменено пользователем. Длина имени не должна превышать 64 символа.


Использование файлов библиотеки функц. блоков OMRON

Для вставки файлов библиотеки функциональных блоков OMRON (.cxf) в создаваемый проект необходимо выполнить следующие действия.

1. Щелкните правой кнопкой мыши по узлу **Function Blocks (Функциональные блоки)** в рабочей области проекта и выберите **Insert Function Blocks – Library File (Вставить функциональный блок – Файл библиотеки)** в контекстном меню. (Или выберите пункт **Function Block – Library File (Функц. блок – Файл библиотеки)** в меню Insert (Вставка).)
2. Отобразится показанное ниже диалоговое окно Select Function Block Library File (Выбор файла библиотеки функциональных блоков).



Примечание. Папка с файлами библиотеки, отображаемая в диалоговом окне Function Block Library File по умолчанию, может быть задана пользователем. Выберите **Tools – Options (Сервис – Настройка)**, откройте вкладку **General (Общие)** и выберите отображаемую по умолчанию папку в поле **OMRON FB library storage location (Расположение библиотеки функц. блоков OMRON)**.

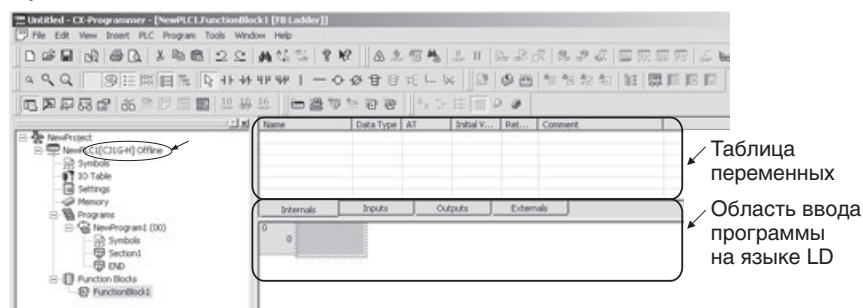
- Укажите папку, в которой располагается файл библиотеки функциональных блоков OMRON, выберите требуемый файл библиотеки и щелкните кнопку **Open (Открыть)**. Определение функционального блока, извлеченное из файла библиотеки, будет вставлено после значка .

Определения функциональных блоков

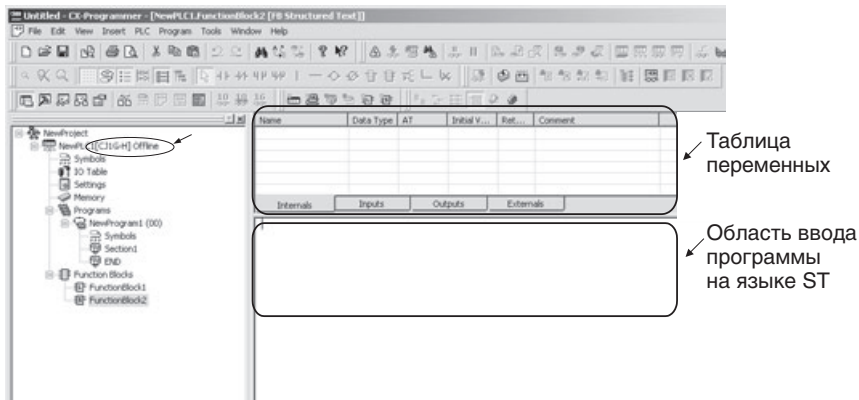
Создание определений функциональных блоков

Двойной щелчок по вновь созданному значку Function Block 1 (или щелчок правой кнопкой и выбор команды **Open (Открыть)** в контекстном меню) приведет к отображению одного из следующих окон. В верхней части окна находится таблица переменных для переменных, используемых в функциональном блоке. В нижней части окна находится программа на языке LD или ST.

Релейно-контактная схема (LD)



Структурированный текст (ST)



Как следует из приведенных выше рисунков, определение функционального блока состоит из таблицы переменных и программы на языке LD или ST. Программа реализует алгоритм функционального блока, а таблица переменных обеспечивает возможность взаимодействия с функциональным блоком.

Таблица переменных (интерфейс)

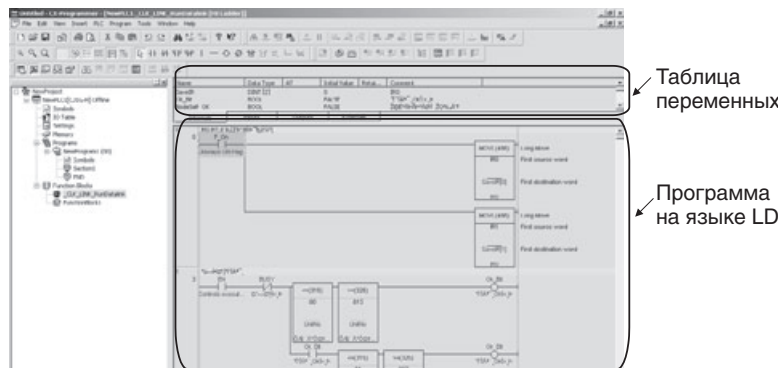
На этом этапе таблица переменных пуста, так как переменные еще не определены.

Программа на языке LD или ST (алгоритм)

- Программа функционального блока на языке LD может содержать практически те же команды, что и обычная программа (некоторые команды, однако, не поддерживаются). Ограничения в отношении использования команд указаны в разделе 2-4 *Ограничения при программировании*.
- Язык программирования ST, используемый для ввода программы функционального блока в виде структурированного текста, соответствует стандарту МЭК 61131-3.

Использование файлов библиотеки функц. блоков OMRON

Дважды щелкните по функциональному блоку, вставленному из файла библиотеки (или щелкните правой кнопкой и выберите **Open (Открыть)** в контекстном меню). В верхней части окна отобразится таблица переменных этого функционального блока, а в нижней части окна — его программа на языке LD. Ни таблица переменных, ни программа не будут доступны для редактирования.



Примечание. По умолчанию определения функциональных блоков для файлов библиотеки функц. блоков OMRON (.cxf) не отображаются. Для того чтобы они отображались, в свойствах функционального блока должен быть выбран параметр **Display the inside of FB (Отобразить внутр.**

содерж. функц. блока). (Выберите файл библиотеки функц. блоков OMRON в рабочей области проекта, щелкните правой кнопкой мыши, выберите **Properties (Свойства)**, после чего установите флажок **Display the inside of FB** на вкладке General (Общие).)

3-2-3 Создание функциональных блоков: подробное описание

В процессе создания функционального блока регистрируются переменные и создается программа функционального блока. Может использоваться любой из следующих подходов.

- Сначала регистрируются переменные, затем вводится программа на языке LD или ST.
- Переменные регистрируются по мере необходимости в процессе ввода программы на языке LD или ST.

Регистрация переменных до создания программы

Регистрация переменных в таблице переменных

Переменные классифицируются по типу использования. Для каждого типа в таблице переменных предусмотрена отдельная вкладка: Internals (внутренние переменные), Inputs (входные переменные), Outputs (выходные переменные), Input-Output (входные-выходные переменные) и Externals (внешние переменные).

Для регистрации или отображения переменной того или иного типа требуется открыть соответствующую вкладку в таблице переменных.

1,2,3...

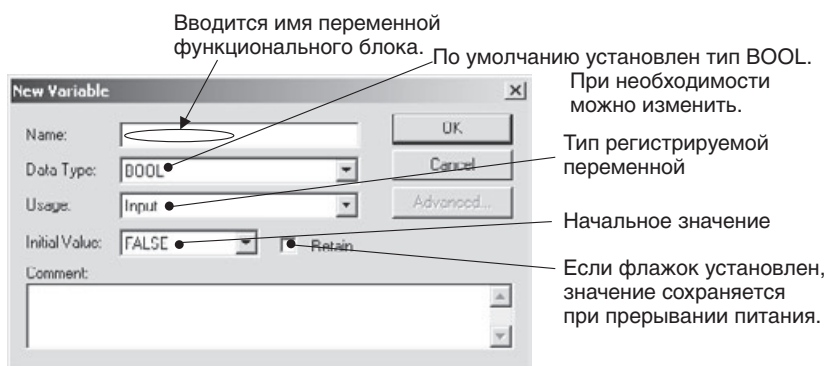
1. В таблице переменных откройте вкладку, соответствующую типу переменной, которую вы хотите зарегистрировать (см. примеч.). Расположите указатель мыши над вкладкой, вызовите контекстное меню щелчком правой кнопки и выберите в нем одну из следующих команд:

- **Insert Variable (Вставить переменную)** — для добавления переменной в последнюю строку.
- **Insert Variable – Above (Вставить переменную – Над строкой)** или **Below (Под строкой)** — для добавления переменной в строку, расположенную над или под выбранной строкой таблицы.

Примечание. Требуемую вкладку таблицы (т. е. тип переменной) также можно выбрать непосредственно при вставке переменной, указав тип переменной (N: внутренние, I: входные, O: выходные, E: внешние, P: вх.-вых).

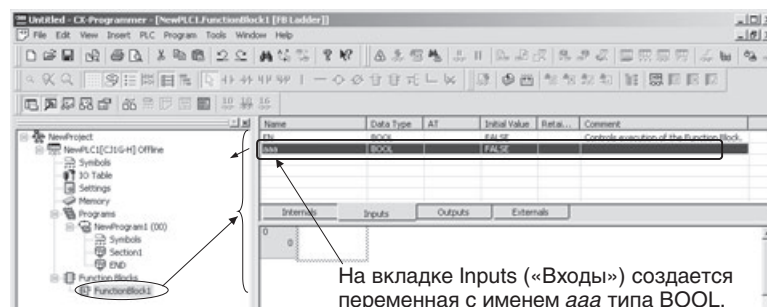
Отобразится показанное ниже диалоговое окно New Variable (Новая переменная).

- **Name (Имя):** введите имя переменной.
- **Data Type (Тип данных):** выберите тип данных.
- **Usage (Использование):** выберите тип переменной.
- **Initial Value (Исходное значение):** выберите исходное значение переменной в начале работы.
- **Retain (Сохран.):** выберите, должно ли сохраняться прежнее значение переменной при включении питания или переключении модуля ЦПУ из режима «Программирование» или «Мониторинг» в режим «Выполнение». Если опция **Retain (Сохранение)** выбрана не будет, значение переменной при выполнении указанных выше действий будет сбрасываться.



- Примечание**
- (a) Для вставки внешней переменной, созданной пользователем, может быть отображена таблица глобальных символов. В таблице глобальных символов должна быть зарегистрирована переменная с таким же именем.
 - (b) Системные внешние переменные включены в таблицу внешних переменных по умолчанию.
2. Например, введите «aaa» в качестве имени переменной и щелкните кнопку **ОК**.

Как показано ниже, на вкладке Inputs (Входные) таблицы переменных будет создана переменная с именем aaa типа BOOL.



- Примечание**
- (1) Зарегистрированную переменную можно выделить, щелкнув по любой ячейке строки переменной, кроме ячейки *Name* (*Имя*). Выделенную строку (отображаемую в инверсном режиме) можно перетащить мышью в требуемую позицию.
 - (2) После ввода переменной вкладку, на которой зарегистрирована переменная, можно изменить, дважды щелкнув кнопкой мыши и изменив параметр в поле *Usage* (*Использование*) (N: внутренние, I: входные, O: выходные, E: внешние, P: вх.-вых). Переменную также можно скопировать или переместить из одной вкладки таблицы в другую. Выделите переменную, вызовите контекстное меню щелчком правой кнопки мыши и выберите команду **Copy** (**Копировать**) или **Cut** (**Вырезать**), после чего выберите **Paste** (**Вставить**).
 - (3) Если для переменной задается параметр AT (указывается конкретный адрес), для нее все равно должно вводиться имя переменной.
 - (4) Ниже перечислены буквенные обозначения адресов памяти ввода-вывода в ПЛК, которые нельзя использовать в качестве имен переменных в таблице переменных функционального блока.

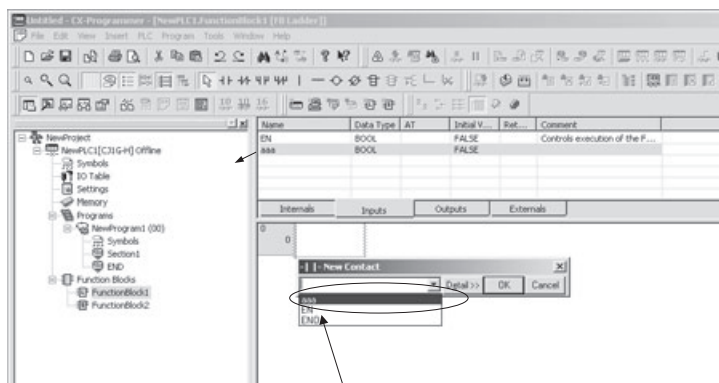
- A, W, H, HR, D, DM, E, EM, T, TIM, C или CNT вместе с числовым значением

Создание программы

Использование языка РКС (LD)

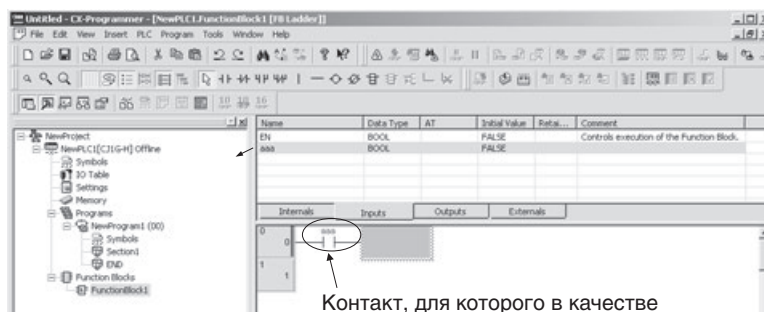
1,2,3...

1. Нажмите клавишу **C** и выберите зарегистрированную ранее переменную *aaa* в раскрывающемся меню в диалоговом окне New Contact (Создание контакта).



Нажмите клавишу **C** и выберите зарегистрированную ранее переменную *aaa* в раскрывающемся меню в диалоговом окне New Contact («Создание контакта»).

2. Щелкните по кнопке **OK**. В схему будет вставлен контакт, операндом которого будет являться внутренняя переменная *aaa* функционального блока (тип переменной: внутренняя).



Контакт, для которого в качестве операнда указана внутренняя переменная *aaa* функционального блока.

Во всем остальном ввод лестничной диаграммы ничем не отличается от ввода обычной программы CX-Programmer.

Примечание.

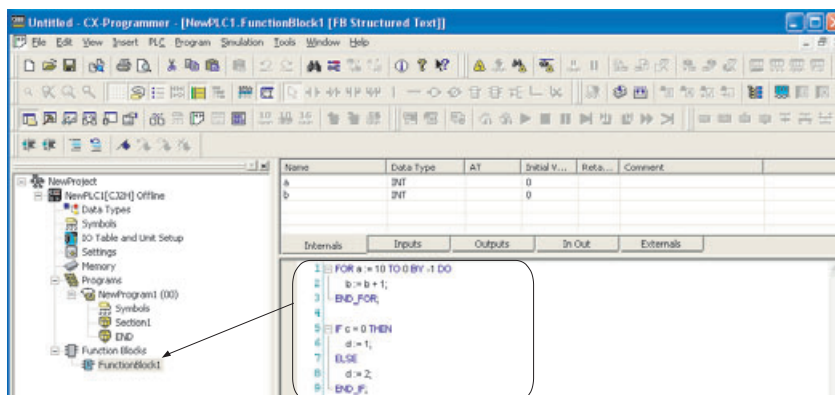
В операнды команд программы функционального блока не могут вводиться непосредственно адреса ячеек памяти. Могут вводиться только адреса регистров указателей (IR) и регистров данных (DR) (не в качестве переменных) в следующем формате: DR0...DR5, IR0...IR15 (прямое обращение к регистрам) и ,IR0...,IR15 (косвенное обращение к адресам памяти).

Использование структурированного текста (ST)

Программу на языке ST (см. примеч.) можно либо сразу вводить в окне ввода программы в CX-Programmer, либо набрать ее в обычном текстовом редакторе, после чего скопировать и вставить в окно ввода программы с помощью команды *Paste (Вставить)* в меню Edit (Правка).

Примечание.

Язык структурированного текста (ST) соответствует стандарту МЭК 61131-3. Подробные сведения смотрите в разделе **РАЗДЕЛ 5 Характеристики языка структурированного текста в Часть 2: Структурированный текст (ST)**.



Программа на языке ST вводится непосредственно или вставляется из текстового редактора.

Примечание

- (1) При вводе программы допускается использовать знаки «пробела» и «табуляции». На выполнение программы они не влияют.
- (2) Как при непосредственном наборе программы на языке ST в окне ввода программы, так и после вставки программы из текстового редактора все зарезервированные ключевые слова автоматически отображаются шрифтом синего цвета, комментарии отображаются зеленым цветом, ошибки — красным цветом, а все остальное — черным цветом.
- (3) Для того чтобы изменить размер или цвет шрифта, выберите команду **Options (Настройка)** в меню Tools (Сервис), после чего щелкните кнопку **ST Font (Шрифт ST)** на вкладке Appearance (Вид). В данном окне можно поменять шрифт, размер шрифта (по умолчанию установлено 8 точек) и его цвет.
- (4) Информацию о характеристиках языка структурированного текста содержит **РАЗДЕЛ 5 Характеристики языка структурированного текста** в *Часть 2: Структурированный текст (ST)*.

Регистрация переменных по мере необходимости

Переменные, используемые в программе на языке LD или ST, необязательно регистрировать заранее. Их можно зарегистрировать как по завершении работы над программой, так и непосредственно при ее написании.

Использование языка РКС (LD)

При вводе не зарегистрированного ранее имени переменной в программу на языке LD автоматически отображается диалоговое окно регистрации переменной. Таким образом, при создании программы на языке LD переменная регистрируется, когда она впервые вводится в программу.

Соблюдайте порядок действий, представленный ниже.

1,2,3...

1. Нажмите клавишу **C** и введите не зарегистрированное ранее имя переменной, например *aaa*, в диалоговом окне New Contact (Создание контакта).

Примечание. В операнды команд программы функционального блока не могут вводиться непосредственно адреса ячеек памяти. Могут вводиться только адреса регистров указателей (IR) и регистров данных (DR) (не в качестве переменных) в следующем формате: DR0...DR5, IR0...IR15 (прямое обращение к регистрам) и ,IR0...,IR15 (косвенное обращение к адресам памяти).

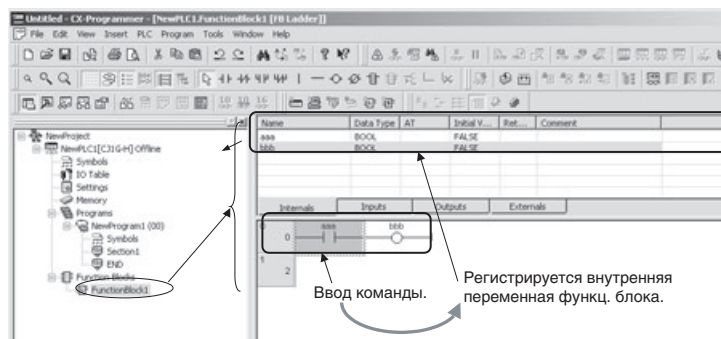
- Щелкните по кнопке **ОК**. Отобразится диалоговое окно New Variable (Создание переменной). Для некоторых команд диалоговое окно New Variable отображается отдельно для каждого операнда команды.



Задается тип данных и другие параметры (кроме имени).

Для всех входных переменных параметры первоначально (по умолчанию) имеют следующие значения:

- Usage (Использование): Internal (Внутр.)
 - Data Type (Тип данных): BOOL — для контактов, WORD — для регистров (слов)
 - Initial Value (Исходное значение): значение по умолчанию, соответствующее типу данных
 - Retain (Сохран.): не выбрано
- Установите требуемые значения параметров и щелкните кнопку **ОК**.
 - Как видно из рисунка ниже, зарегистрированная переменная отобразится в таблице переменных, расположенной над программой.



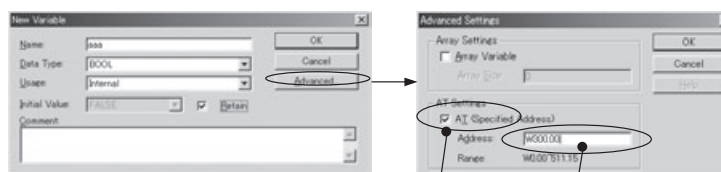
- Если тип или свойства введенной переменной оказались неверны, дважды щелкните по переменной в таблице переменных и внесите необходимые корректировки.

■ Справочная информация

Параметр AT (назначение конкретного адреса)

С помощью параметра AT переменной может быть назначен конкретный адрес, принадлежащий базовому модулю ввода/вывода, специальному модулю ввода/вывода или модулю шины ЦПУ, либо адрес вспомогательной области, не зарегистрированный в CX-Programmer. Несмотря на назначение переменной конкретного адреса, для нее все равно должно быть введено имя. Порядок действий описан ниже.

- 1,2,3...**
- Завершив ввод имени переменной в диалоговом окне New Variable (Новая переменная), щелкните кнопку **Advanced (Дополнительно)**. Отобразится диалоговое окно Advanced Settings (Дополнительные параметры).
 - Установите флажок *AT (Specified Address) (Указанный адрес)* в поле *AT Settings (Назначение адреса)* и введите требуемый адрес.



Выберите AT. Введите адрес.

Даже если переменной с помощью параметра AT назначен определенный адрес, в программе функционального блока для ее обозначения используется имя (как и для переменных, для которых адреса не указаны).

Например, если для переменной с именем *Restart* с помощью параметра AT указан адрес A50100, в операнде команды должно быть указано имя *Restart*.

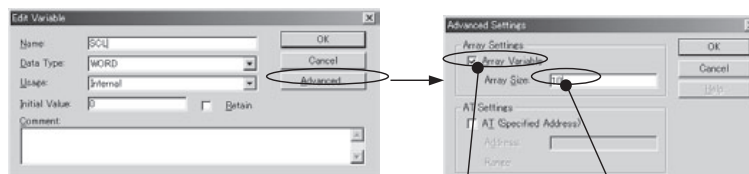
Определение переменной в качестве массива

Переменная может быть определена как массив. В этом случае множеством однотипных переменных можно оперировать как одним объектом.

Для определения переменной в качестве массива соблюдайте следующий порядок действий.

1,2,3...

1. Завершив ввод имени переменной в диалоговом окне *New Variable* (Новая переменная), щелкните кнопку **Advanced** (Дополнительно). Отобразится диалоговое окно *Advanced Settings* (Дополнительные параметры).
2. Установите флажок *Array Variable* (Переменная-массив) в поле *Array Settings* (Определение массива) и введите максимальное количество элементов массива.



Выберите *Array Variable* («Переменная-массив»). Введите число элементов.

Когда в программу определения функционального блока вводится имя переменной, являющейся массивом, после имени переменной автоматически отображаются квадратные скобки, внутри которых должен быть указан номер элемента массива.

Допустим, к примеру, что создана переменная-массив под именем PV с максимальным количеством элементов 3. В этом случае в качестве операндов команд могут указываться следующие переменные: PV[0], PV[1] и PV[2].

Номер элемента массива может быть указан одним из трех следующих способов.

- Может быть указано непосредственно числовое значение. Пример: PV[1] (в программе на языке LD или ST).
- Может быть указана переменная. Пример: PV[a], где «a» — имя переменной типа INT (в программе на языке LD или ST).

- Может быть указано выражение. Пример: PV[a+b] или PV[a+1], где «a» и «b» — имена переменных типа INT (только в программе на языке ST).

Использование структурированного текста (ST)

Если программа создается на языке структурированного текста и в нее вводится не зарегистрированное ранее имя переменной, диалоговое окно регистрации переменной при этом не вызывается. Следует обязательно зарегистрировать все новые переменные в таблице переменных. Это можно сделать либо сразу при вводе переменных в программу, либо по завершении создания программы. (Расположите указатель мыши на вкладке, соответствующей типу регистрируемой переменной, вызовите контекстное меню и выберите команду ***Insert Variable (Вставить переменную)***.)

Примечание. Информацию о характеристиках языка структурированного текста содержит **РАЗДЕЛ 5 Характеристики языка структурированного текста** в **Часть 2: Структурированный текст (ST)**.

Копирование фрагментов существующей программы на языке LD в программу функционального блока

Одну или несколько цепей лестничной диаграммы пользователя можно скопировать и вставить в лестничную диаграмму функционального блока. Имеются, однако, указанные ниже ограничения.

Входной операнд команды: только адрес

Адреса в таблице переменных определения функционального блока не регистрируются. После вставки программы используемые в операндах адреса отображаются красным цветом. Следует дважды щелкнуть по программе и ввести в операндах имена переменных.

Примечание. Для регистров указателей (IR) и регистров данных (DR) никаких изменений после вставки производить не требуется, они работают в том виде, в каком они записаны в операндах.

Входной операнд команды: адрес и комментарий к входу/выходу

На вкладке **Symbols (Символы) диалогового окна **Options (Настройки)** (меню **Tools (Сервис)**) выбран параметр **Automatically generate symbol name (Автоматически генерировать имена символов)**.**

Автоматически генерируются имена символов программы пользователя (только в таблице глобальных символов) вида: AutoGen_ + адрес (если выбор данного параметра будет отменен, имена символов будут удалены из таблицы).

Пример 1: для адреса 100.01 будет сгенерировано имя символа AutoGen_100_01.

Пример 2: для адреса D0 будет сгенерировано имя символа AutoGen_D0.

Если сегменты лестничной диаграммы пользователя копируются и вставляются в программу определения функционального блока в своем первоначальном виде, символы в таблице символов определения функционального блока регистрируются автоматически (одновременно с копированием и вставкой сегментов), при этом им автоматически присваиваются имена вида AutoGen_адрес, а в поле **Comment (Комментарий)** передаются комментарии ко входам-выходам. Благодаря этой функции сегменты существующих лестничных диаграмм могут быть легко применены в создаваемых функциональных блоках со своими адресами и комментариями ко входам/выходам.

Примечание. К адресам регистров указателей (IR) и регистров данных (DR) префикс AutoGen_ не добавляется, и их невозможно зарегистрировать в таблице глобальных символов исходной программы.

На вкладке Symbols (Символы) диалогового окна Options (Настройки) (меню Tools (Сервис)) не выбран параметр Automatically generate symbol name (Автоматически генерировать имена символов).

Адреса и комментарии ко входам/выходам в таблицах переменных определений функциональных блоков не регистрируются. Адреса, используемые в операндах, отображаются шрифтом красного цвета. Комментарии ко входам/выходам утрачиваются. Дважды щелкните по каждой команде и введите в ее операндах имена символов.

Для регистров указателей (IR) и регистров данных (DR) никаких изменений после вставки производить не требуется, они работают в том виде, в каком они записаны в операндах.

Входной операнд команды: символ

Символы программы пользователя автоматически регистрируется в качестве внутренних переменных в таблице переменных определения функционального блока. Имеются, однако, указанные ниже ограничения.

Адреса

Адреса символов не регистрируются. С помощью параметра AT необходимо указать такой же адрес.

Типы данных символов

При вставке сегмента программы пользователя в определение функционального блока типы данных символов преобразуются в соответствии со следующей таблицей.

Тип данных символа в программе пользователя	→	Тип данных переменной после вставки в программу функц. блока
CHANNEL	→	WORD
NUMBER	→	Переменная не регистрируется, значение (число) непосредственно вставляется в операнд в качестве константы.
UINT BCD	→	WORD
UDINT BCD	→	DWORD
ULINT BCD	→	LWORD

Символы с типом данных CHANNEL, NUMBER, UINT BCD, UDINT BCD или ULINT BCD не могут быть, однако, скопированы в таблице символов (но не в программе) и вставлены в таблицу переменных определения функционального блока.

Примечание. Символы с автоматически сгенерированными именами (AutoGen_ + адрес) не могут быть скопированы в таблице глобальных символов и вставлены в таблицу символов определения функционального блока.

Преобразование существующей программы на языке LD в определение функционального блока

Одна или несколько цепей лестничной диаграммы пользователя могут быть преобразованы в лестничную диаграмму функционального блока.

Примечание. Данная функция была создана специально для того, чтобы определения функциональных блоков можно было генерировать автоматически из уже существующих программ, написанных на языке LD. Эта функция,

однако, не создает готовые к использованию определения. Получив с помощью данной функции определение функционального блока, следует обязательно ознакомиться с содержанием предупреждающих сообщений в диалоговом окне FB Variable Allocation (Распределение переменных функц. блока) и в окне вывода информации, тщательно проверить созданную программу и внести в нее необходимые изменения.

1,2,3...

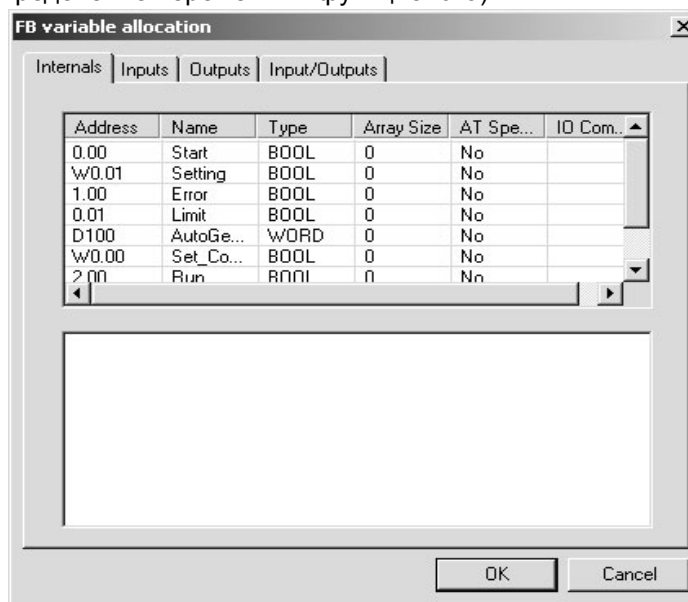
1. Щелкните правой кнопке мыши по одной или нескольким выделенным цепям лестничной диаграммы (программы пользователя) и выберите команду *Function Block (ladder) generation (Сгенерировать функциональный блок (LD))*.

Примечание.

При наличии в программе любого структурного типа данных команда *Function Block (ladder) generation* в контекстном меню будет недоступна для выбора.



2. Откроется показанное ниже диалоговое окно FB Variable Allocation (Распределение переменных функц. блока).



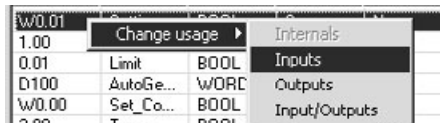
Адреса операндов, используемые в командах выделенных сегментов прикладной программы, будут автоматически распределены согласно представленной ниже таблице и в соответствии с предназначением операндов.

Применение за пределами выбранных сегментов программы	Применение внутри выбранных сегментов программы			
	Не используются	Исп. во входном сегменте	Исп. в выходном сегменте	Исп. во входном и выходном сегментах
Не используются (см.примеч.)	---	Внутр. переменная	Внутр. переменная	Внутр. переменная
Используются	---	Входная переменная	Выходная переменная	Вх.-вых. переменная

Примечание. Даже если адрес принадлежит входному или выходному операнду, он все равно считается «неиспользуемым» и преобразуется во внутреннюю переменную, если он не используется за пределами выделенных сегментов программы (и независимо от того, используется ли он внутри выделенных сегментов).

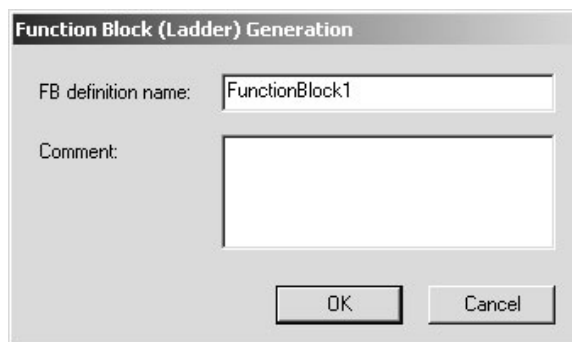
Примечание. Адресам, не имеющим символьных имен, автоматически присваиваются имена вида: AutoGen_адрес. Параметр АТ (принудительно назначенный адрес) автоматически удаляется.

3. При необходимости тип использования каждой переменной можно изменить. Щелкните правой кнопкой мыши по переменной и выберите требуемый тип переменной (внутренняя, входная, выходная или входная-выходная переменная) в меню **Change usage (Изменить тип использования)**.



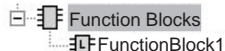
Если требуется, можно изменить имя или комментарий для любой переменной в таблице переменных (дважды щелкнув по ней). Кроме того, могут быть изменены параметры, связанные с определением переменной в качестве массива и назначением ей конкретного адреса.

4. Щелкните по кнопке **ОК**. Будет отображено показанное ниже диалоговое окно Function Block (Ladder) Generation (Генерация функционального блока (LD)).

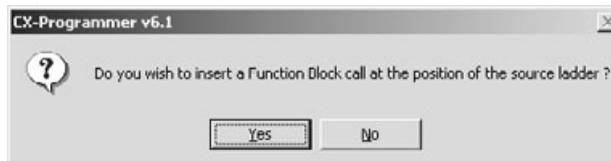


Введите имя определения функционального блока и комментарий, затем щелкните кнопку **ОК**.

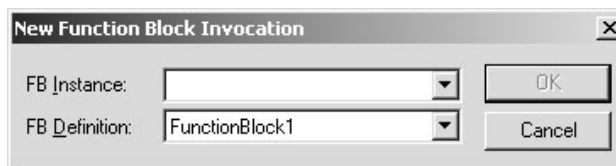
- Сгенерированное в соответствии с заданными параметрами определение функционального блока будет отображено на дереве проекта вместе с другими функциональными блоками.



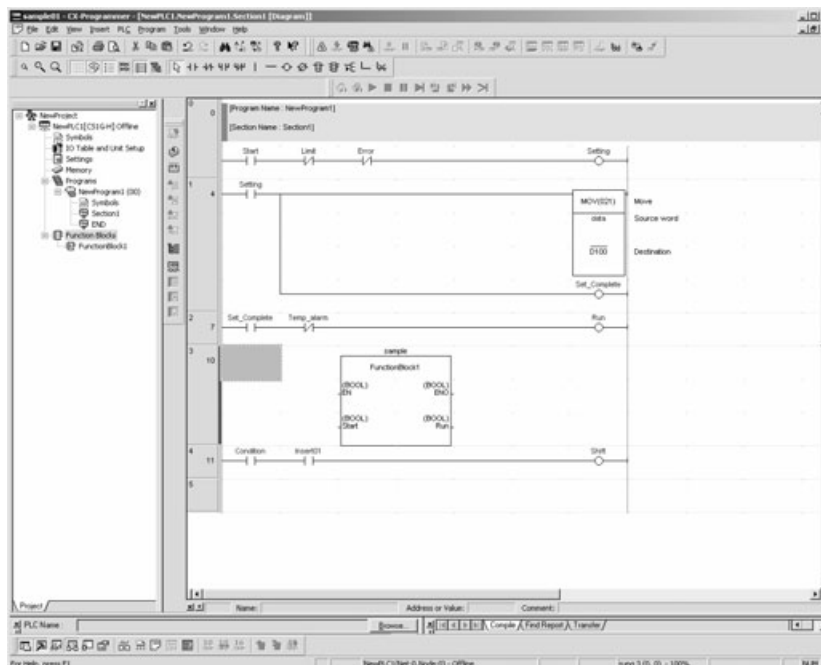
- Вслед за этим отобразится показанное ниже диалоговое окно, предлагающее вставить экземпляр созданного определения функционального блока под исходными сегментами программы.



- Щелкните кнопку **Yes (Да)**, чтобы вставить экземпляр, либо кнопку **No (Нет)**, если вставлять экземпляр не требуется.
- В случае щелчка по кнопке Yes (Да) отобразится показанное ниже диалоговое окно New Function Block Invocation (Новый вызов функционального блока).



Введите имя экземпляра функционального блока и щелкните кнопку **OK**. Под исходными сегментами программы будет вставлен экземпляр определения функционального блока.



Примечание.

- Введите входные условия и параметры для вставленного экземпляра. Функция генерации определений функциональных блоков весьма удобна для преобразования уже существующих и проверенных на деле лестничных диаграмм в функциональные блоки. Предназначение адресов в выделенных сегментах программы тщательно анализируется

(в том числе анализируется их использование за пределами выделенной части программы), на основании чего максимально точно определяется тип переменной (внутренняя, входная, выходная или входная-выходная), в которую преобразуется каждый адрес. Однако если сегменты программы не содержат адресов (в операндах используются только символы), они не могут быть преобразованы. Для использования таких сегментов в программе функционального блока их следует скопировать и вручную вставить в программу создаваемого определения функционального блока. Подробное описание смотрите в разделе *Копирование фрагментов существующей программы на языке LD в программу функционального блока* на стр. 110.

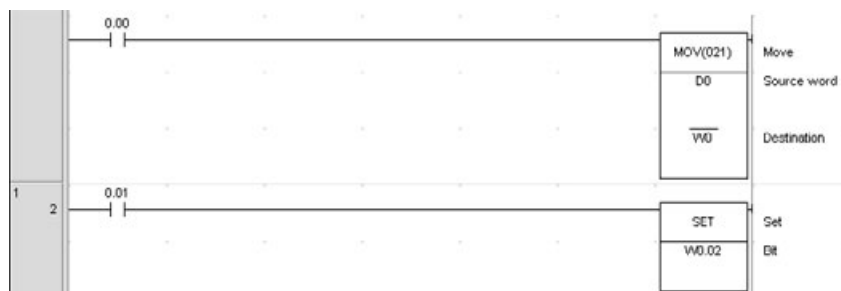
Сегменты программы, требующие корректировки перед генерацией определения функционального блока

В описанных ниже случаях автоматическая генерация определений функциональных блоков возможна лишь после внесения ряда изменений в цепи лестничной диаграммы.

Одновременное использование адреса бита и адреса слова

Адрес бита и адрес слова будут зарегистрированы как разные переменные. Чтобы этого не произошло, программу необходимо предварительно видоизменить.

Пример: MOV(021) для W0 и SET для W0.02



В данном случае в команде вместо адреса бита можно указать адрес слова. В новом варианте программы в командах MOV(021) и SETB(532) используется один и тот же адрес W0, а номер бита в команде SETB(532) указывается с помощью литерала &2.



Сегменты программы, требующие корректировки после генерации определения функционального блока

В описанных ниже случаях после генерации определения функционального блока некоторые операнды необходимо переопределить как массивы.

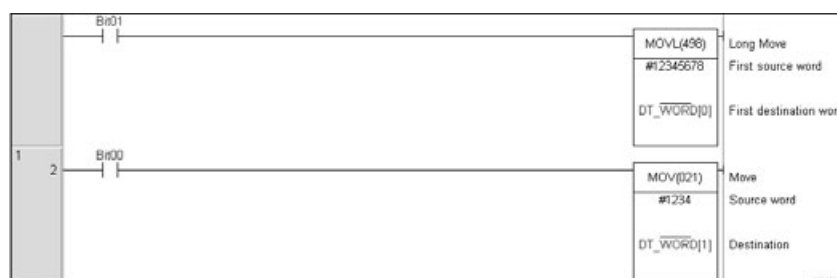
Команды с операндами из нескольких слов, некоторые из которых изменяются другой командой

Пример: для команды MOVL(498) в качестве первого слова указано слово D0, а для команды MOV(021) указано слово D1.



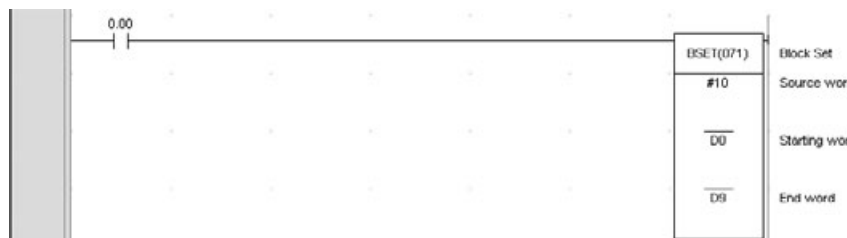
Как видно из рисунка ниже, после формирования определения функционального блока вместо двух разных переменных используется одна переменная-массив.

Пример: переменная DT_WORD определена как массив с двумя элементами типа WORD. В команде MOV(L)498 указывается DT_WORD[0], а в команде MOV(021) указывается DT_WORD[1].



Команды с двумя операндами, указывающими начальное и конечное слова

Пример: для команды BSET(071) указывается группа слов D0...D9



Как показано на рисунке ниже, после формирования определения функционального блока вместо десяти разных переменных в команде используется одна переменная-массив с десятью элементами.

Пример: переменная DT_WORD определена как массив с 10 элементами типа WORD. В качестве первого операнда указывается первый элемент массива DT_WORD[0], а в качестве второго операнда указывается последний элемент массива DT_WORD[9].



Операнды, размер которых зависит от других операндов

Пример: передача пяти слов данных с помощью команды XFER(070). Для первого исходного слова указан адрес D0, для первого слова назначения указан адрес D100.



Из приведенного ниже рисунка видно, что после формирования определения функционального блока в качестве операндов команды указываются первые элементы двух разных массивов.

Пример: переменные DT_WORD1 и DT_WORD2 определяются как массивы, содержащие по 5 элементов типа WORD. В первом операнде команды XFER(070) в качестве первого слова указывается первый элемент массива DT_WORD1[0], а во втором операнде в качестве первого слова указывается первый элемент массива DT_WORD2[0].



3-2-4 Создание экземпляров определений функциональных блоков

Если определение функционального блока зарегистрировано в таблице глобальных символов, для создания экземпляров можно использовать любой из описанных ниже способов.

Способ 1: выбрать определение функционального блока, вставить его в программу и ввести новое имя экземпляра. Экземпляр будет автоматически зарегистрирован в таблице глобальных символов.

Способ 2: выбрать в таблице глобальных символов тип данных «FUNCTION BLOCK», указать используемое определение функционального блока и ввести имя экземпляра, чтобы зарегистрировать экземпляр.

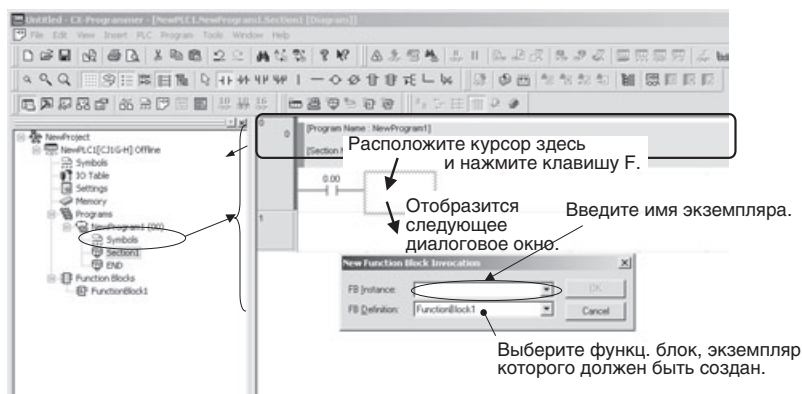
Примечание. В случае использования языка ST функциональный блок также можно вызвать следующим образом: указать для переменной тип данных «FUNCTION BLOCK», использовать требуемое имя экземпляра, ввести выражение вызова функционального блока.

Способ 1: использование клавиши F в окне сегмента лестничной диаграммы и ввод имени экземпляра

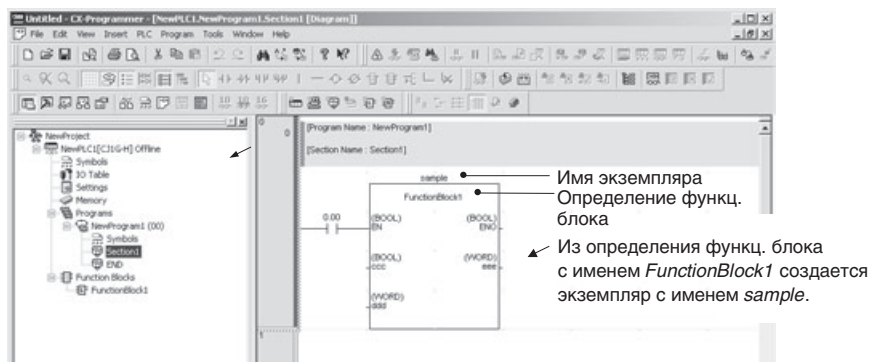
- 1,2,3...
1. Находясь в окне Ladder Section («Сегмент лестничной диаграммы»), расположите указатель мыши в том месте программы, где требуется создать экземпляр функционального блока, и нажмите клавишу **F**. (Либо выберите пункт **Function Block Invocation (Вызов функционального блока)** в меню Insert (Вставка).) Откроется диалоговое окно New Function Block Invocation (Новый вызов функционального блока).

В программе на языке ST для вызова функционального блока также можно указать для переменной тип данных «FUNCTION BLOCK», использовать требуемое имя экземпляра и ввести выражение вызова функционального блока. После имени экземпляра в скобках следует указать аргументы (для передачи значений входных переменных из вызывающего функционального блока во входные переменные в вызываемом функциональном блоке) и возвращаемые значения (для передачи значений выходных переменных из вызываемого функционального блока в выходные переменные в вызывающем функциональном блоке). В качестве имени экземпляра может быть установлена любая внутренняя переменная с типом данных «FUNCTION BLOCK».

2. Введите имя экземпляра, выберите функциональный блок, экземпляр которого вы хотите создать, и щелкните кнопку **OK**.



3. Например, введите для экземпляра в поле **FB Instance (Экземпляр функц. блока)** имя **sample**, в поле **FB Definition (Определение функц. блока)** выберите **FunctionBlock1** и щелкните **OK**. В результате для определения функционального блока с именем **FunctionBlock1** будет создан экземпляр с именем **sample**.



Созданный экземпляр будет автоматически зарегистрирован в таблице глобальных символов с именем экземпляра *sample* и типом данных *FUNCTION BLOCK*.

■ Способ 2: предварительная регистрация имени экземпляра в таблице глобальных символов

Имя экземпляра, заранее зарегистрированное в таблице глобальных символов, выбирается из таблицы глобальных символов для создания других экземпляров.

- 1,2,3...
1. Для программы на языке LD: выберите тип данных *Function block* в таблице глобальных символов, введите имя экземпляра и зарегистрируйте экземпляр.
Для программы на языке ST: выберите тип данных *Function block*, используйте имя экземпляра и примените описанное ниже выражение вызова функционального блока:
Введите имя экземпляра (имя любой внутренней переменной с типом данных «Function block»), указав в скобках необходимые аргументы (а именно: значения входных переменных вызываемого функционального блока, которые должны быть переданы во входные переменные вызываемого функционального блока) и возвращаемые значения (а именно: значения выходных переменных вызываемого функционального блока, которые должны быть возвращены в выходные переменные вызываемого функционального блока).
 2. Нажмите клавишу **F** в окне Ladder Section (Сегмент лестничной диаграммы). Будет отображено диалоговое окно Function Block Invocation (Вызов функционального блока).
 3. Выберите зарегистрированное ранее имя экземпляра в раскрывающемся меню в поле *FB Instance (Экземпляр функц. блока)*. В результате будет создан экземпляр.

Ограничения

При создании экземпляров необходимо принимать во внимание следующие ограничения. Подробное описание смотрите в разделе 2-4 *Ограничения при программировании*.

- В одной цепи лестничной диаграммы может быть создан только один функциональный блок.
- В цепи не допускается наличие ответвлений слева от экземпляра.
- Экземпляр не должен подсоединяться непосредственно к левой шине, между ним и левой шиной обязательно должно находиться входное условие (EN).

Примечание. После внесения изменений во входные или выходные переменные определения функционального блока отрезок левой шины слева от каждого экземпляра, созданного из этого определения функционального блока, отображается красным цветом, сигнализируя ошибку. Если это произошло, необходимо щелкнуть по функциональному блоку правой кнопкой мыши и выбрать команду **Update Invocation (Обновить вызов)**. Каждый экземпляр будет обновлен с учетом всех изменений, произведенных в определении функционального блока, после чего индикация ошибки (отображение отрезка левой шины красным цветом) прекратится.

3-2-5 Ввод параметров функционального блока с помощью клавиши «Ввод»

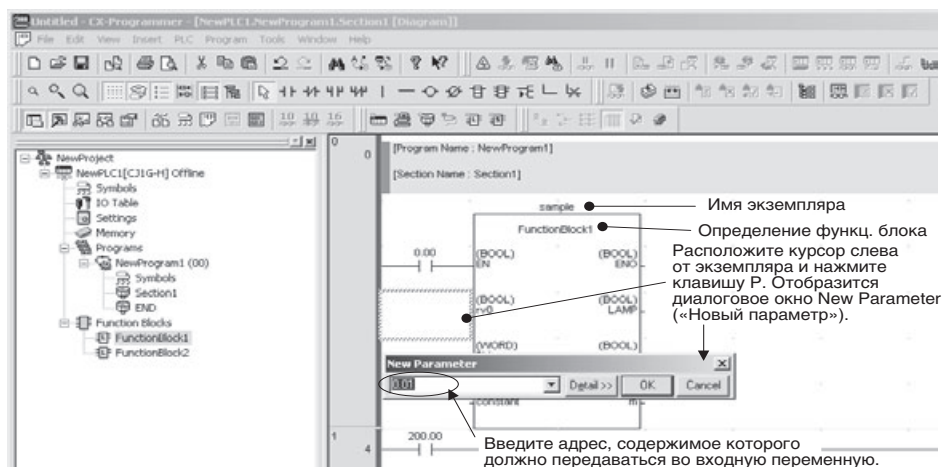
После того как функциональный блок создан, для его входных и выходных переменных требуется создать, соответственно, входные и выходные параметры, чтобы мог быть произведен обмен входными и выходными данными.

- В качестве входных параметров могут вводиться непосредственно значения, адреса, а также символьные имена программы (глобальные и локальные символы) (см. примеч. а).
- В качестве выходных параметров могут вводиться адреса и символьные имена программы (глобальные и локальные символы) (см. примеч. б).

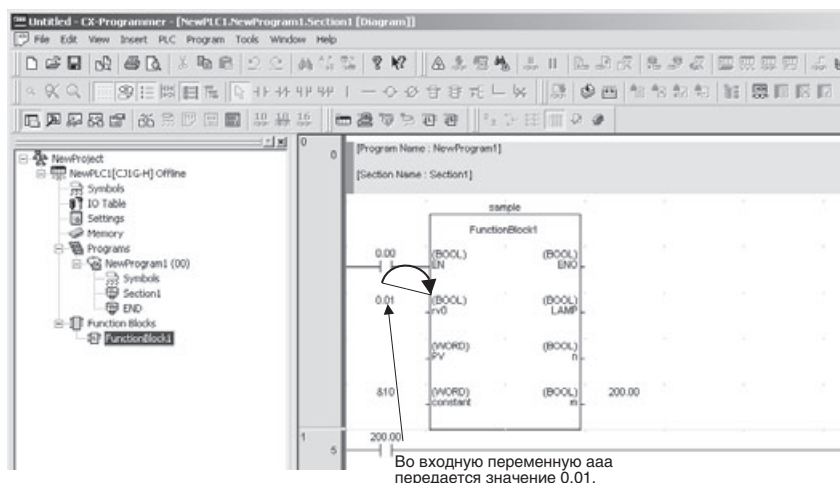
Примечание (а) Размер входной переменной функционального блока должен совпадать с размером символа программы (под размером понимается количество элементов данных).

(б) Размер выходной переменной функционального блока должен совпадать с размером символа программы (под размером понимается количество элементов данных).

- 1,2,3...
1. Входы экземпляра располагаются в его левой части, а выходы — в правой. Расположите указатель мыши в том месте, где должен быть задан параметр, и нажмите клавишу «Ввод». (Либо выберите пункт **Function Block Parameter (Параметр функционального блока)** в меню Insert (Вставка).) Отобразится показанное ниже диалоговое окно New Parameter (Новый параметр).



2. Укажите адрес источника, то есть адрес, содержащий данные, которые должны быть переданы во входную переменную. Также укажите адрес назначения, то есть адрес, по которому должны быть переданы данные из выходной переменной.



Примечание. Запишите необходимые данные во все входные параметры. Даже если всего один параметр останется незадаанным, линия шины программы слева от экземпляра будет отображаться красным цветом, сигнализируя ошибку. В этом состоянии программу загрузить в модуль ЦПУ невозможно.

Ввод значений в параметры

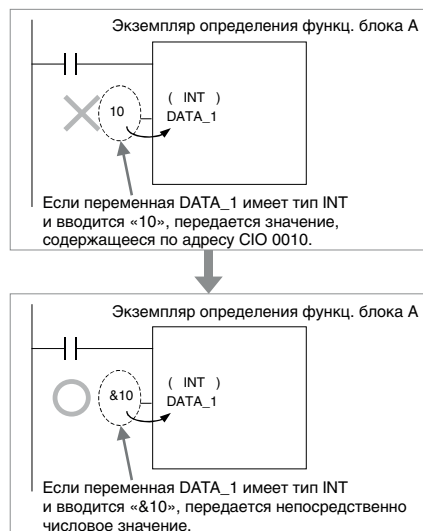
В следующей таблице перечислены все возможные варианты значений, которые могут вводиться в параметр.

Тип данных входной переменной	Содержание	Размер	Способ ввода	Диапазон значений
BOOL	Логическое значение	1 бит	P_Off, P_On	0 (ЛОЖЬ), 1 (ИСТИНА)
INT	Целое	16 бит	Положительное значение: целое число со знаком «&» или «+» спереди Отрицательное значение: целое число со знаком «-» спереди	-32768...+32767
DINT	Двойное целое	32 бит		-2147483648...+2147483647
LINT	Длинное (4 слова) целое	64 бит		-9223372036854775808... +9223372036854775807
UINT	Целое без знака	16 бит	Положительное значение: целое число со знаком «&» или «+» спереди	&0...65535
UDINT	Двойное целое без знака	32 бит		&0...4294967295
ULINT	Длинное (4 слова) целое без знака	64 бит		&0...18446744073709551615
REAL	Вещественное число	32 бит	Положительное значение: вещественное число (с десятичной запятой) со знаком «&» или «+» спереди Отрицательное значение: вещественное число (с десятичной запятой) со знаком «-» спереди	-3,402823 × 10 ³⁸ ...-1,175494 × 10 ⁻³⁸ , 0, +1,175494 × 10 ⁻³⁸ ... +3,402823 × 10 ³⁸
LREAL	Длинное вещественное число	64 бит		-1,79769313486232 × 10 ³⁰⁸ ... -2,22507385850720 × 10 ⁻³⁰⁸ , 0, +2,22507385850720 × 10 ⁻³⁰⁸ ... +1,79769313486232 × 10 ³⁰⁸
WORD	16-битовое значение	16 бит	Шестнадцатеричное число (макс. 4 разряда) со знаком «#» спереди Десятичное число со знаком «&» или «+» спереди	#0000...FFFF или &0...65 535

Тип данных входной переменной	Содержание	Размер	Способ ввода	Диапазон значений
DWORD	32-битовое значение	32 бит	Шестнадцатеричное число (макс. 8 разряда) со знаком «#» спереди Десятичное число со знаком «&» или «+» спереди	#00000000...FFFFFFFF или &0...4294967295
LWORD	64-битовое значение	64 бит	Шестнадцатеричное число (макс. 16 разряда) со знаком «#» спереди Десятичное число со знаком «&» или «+» спереди	#0000000000000000...FFFFFFFFFFFFFFFF или &0...18446744073709551615

Примечание. Если входная переменная не является переменной логического типа и в параметр вводится числовое значение без какого-либо префикса (напр., 20), в функциональный блок передается не само это значение, а значение, содержащееся по данному адресу области CIO (напр., CIO 0020). Для передачи числового значения его следует вводить с префиксом &, #, + или -.

Примеры программ:

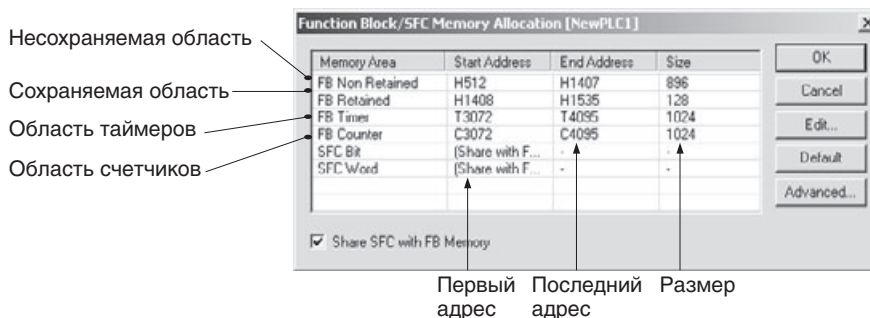


Если входная переменная является переменной логического типа (BOOL) и в параметр вводится числовое значение без какого-либо префикса (т.е., 0 или 1), в функциональный блок передается содержимое CIO 000000 (0.00) или CIO 000001 (0.01). Для передачи состояния «0» (ВЫКЛ) следует вводить P_Off, для передачи состояния «1» (ВКЛ) следует вводить P_On.

3-2-6 Настройка областей экземпляров функциональных блоков

Области памяти, адреса которых назначаются переменным в функциональных блоках, могут быть выбраны пользователем. Эти области памяти называются областями экземпляра функционального блока.

- 1,2,3...
1. Выберите экземпляр в окне сегмента лестничной диаграммы или в таблице глобальных символов, после чего выберите команду **Function Block/SFC Memory – Function Block/SFC Memory Allocation (Память функц. блоков/SFC – Адреса памяти для функц. блоков/SFC)** в меню PLC (ПЛК).
Будет отображено показанное ниже диалоговое окно Function Block/SFC Memory Allocation (Адреса памяти для функц. блоков/SFC).
 2. Задайте области памяти, которые должны использоваться для экземпляра функционального блока.



Границы и размер сохраняемых и несохраняемых областей памяти задаются в словах. Для областей таймеров и счетчиков указываются, соответственно, номера таймеров и счетчиков.

По умолчанию используются следующие значения:

Модули ЦПУ серии CJ2

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	H512	H1407	896	CIO, WR, HR, DM, EM (см. примеч.)
Сохран.	H1408	H1535	128	HR, DM, EM (см. примеч.)
Таймеры	T3072	T4095	1024	TIM
Счетчики	C3072	C4095	1024	CNT

Примечание. Принудительная установка/сброс возможны, если указаны следующие банки памяти EM:

CJ2H-CPU64(-EIP)/-CPU65(-EIP)	EM, банк 3
CJ2H-CPU66(-EIP)	EM, банки 6...9
CJ2H-CPU67(-EIP)	EM, банки 7...E
CJ2H-CPU68(-EIP)	EM, банки 11...18

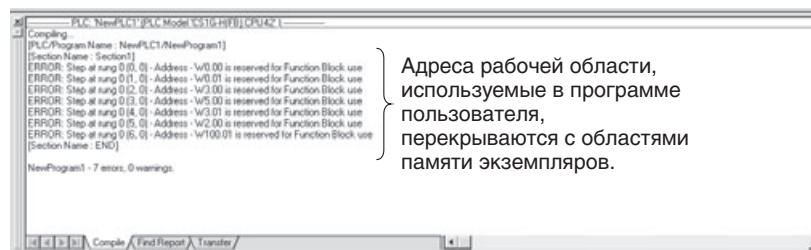
Модули ЦПУ серии CS/CJ версии 3.0 или выше и контроллеры NSJ

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран. (см. примеч. 1 и 3).	H512 (см. примеч. 2)	H1407 (см. примеч. 2)	896	CIO, WR, HR, DM, EM
Сохран. (см. примеч. 1)	H1408 (см. примеч. 2)	H1535 (см. примеч. 2)	128	HR, DM, EM
Таймеры	T3072	T4095	1024	TIM
Счетчики	C3072	C4095	1024	CNT

Примечание

- (1) Обращение к значениям битов возможно, даже если область DM или EM указана для несохраняемой области или сохраняемой области.
- (2) Для функциональных блоков выделяются слова области хранения в диапазоне от H512 до H1535. Эти слова нельзя указывать в качестве операндов команд в программе пользователя. Эти слова также нельзя указывать при настройке параметров AT для внутренних переменных.
- (3) Хотя слова H512...H1535 находятся в области хранения, содержаемое адресов, определенных как «несохраняемые», сбрасывается при выключении/включении питания, а также в начале работы ПЛК.
- (4) Для того чтобы адреса области памяти экземпляров не смешивались с адресами, которые используются прикладной программой, в качестве несохраняемой и сохраняемой областей памяти следует использовать адреса H512...H1535 (слова области хранения для функциональных блоков). Если этих слов окажется недостаточно, следует задействовать слова других областей памяти, не используемых в программе пользователя. В случае указания другой области памяти может произойти наложение на адреса, используемые в программе пользователя.

Если адресное пространство экземпляров функциональных блоков пересечется с адресным пространством программы пользователя, при компиляции будет выдано сообщение об ошибке. Это же сообщение об ошибке будет выдано при загрузке программы, при онлайн-редактировании или при проверке программы пользователем.



В случае возникновения ошибки из-за дублирования адресов следует либо перенастроить области памяти экземпляров функционального блока, либо использовать другие адреса в программе пользователя.

Контроллеры движения FQM1

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран. (см. примеч.)	5000	5999	1000	CIO, WR, DM
Сохран.	Нет			
Таймеры	T206	T255	50	TIM
Счетчики	C206	C255	50	CNT

Примечание. Обращение к состояниям битов возможно, даже если область DM указана для несохраняемой области.

Модули ЦПУ серии CP

Область экзempl. функц. блоков	Значение по умолчанию			Поддерживаемые области памяти
	Начальный адрес	Конечный адрес	Размер	
Не сохран.	H512	H1407	896	CIO, WR, HR, DM (см. примеч.)
Сохран.	H1408	H1535	128	HR, DM (см. примеч.)
Таймеры	T3072	T4095	1024	TIM
Счетчики	C3072	C4095	1024	CNT

Примечание. Область DM в ЦПУ CP1L-L

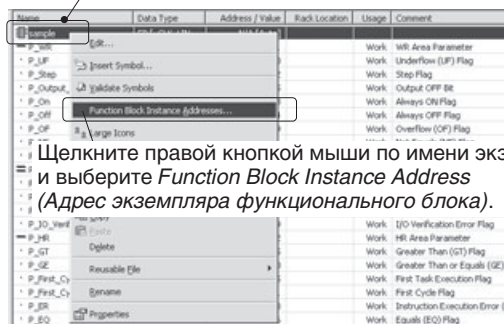
Адрес	CP1L-L
D0000...D9999	Есть
D10000...D31999	Нет
D32000...D32767	Есть

3-2-7 Выяснение внутренних адресов переменных

Описанный ниже порядок действий позволяет выяснить, какие именно адреса памяти ввода/вывода были автоматически назначены переменным для использования внутри экземпляра функционального блока.

- 1,2,3... 1. Выберите **View – Symbols – Global (Вид – Символы – Глобальные)**.
2. Выделите экземпляр в таблице глобальных символов, вызовите контекстное меню щелчком правой кнопки мыши и выберите команду **Function Block/SFC Memory Address (Адреса памяти функц. блока/SFC)**. (Либо выберите **Memory Allocation – Function Block/SFC Memory – Function Block/SFC Memory Address (Распределение памяти – Память функц. блоков/SFC – Адреса памяти для функц. блоков/SFC)** в меню PLC (ПЛК).)

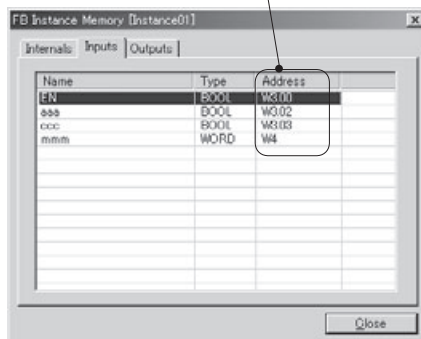
Пример: имя экземпляра отображается в таблице глобальных переменных (зарегистрировано автоматически).



Щелкните правой кнопкой мыши по имени экземпляра и выберите *Function Block Instance Address* (Адрес экземпляра функционального блока).

3. Отобразится диалоговое окно FB Interface Memory (Память интерфейса функц. блока), в котором можно посмотреть, какие именно адреса памяти ввода/вывода выделены для переменных данного экземпляра функционального блока.

Пример: адреса, используемые внутри функц. блока для входных переменных.



Способ определения адресов, используемых для переменных внутри экземпляра



Проверка состояния адресов, автоматически назначаемых переменным

Описанный ниже порядок действий позволяет определить количество адресов, уже назначенных переменным, и количество еще не занятых адресов в областях памяти, используемых для экземпляра функционального блока.

- 1,2,3... 1. Выберите экземпляр в окне сегмента лестничной диаграммы, щелкните правой кнопкой мыши и выберите команду **Memory Allocation – Function Block/SFC Memory – Function Block/SFC Memory Statistics (Распределение памяти – Память функц. блоков/SFC – Состояние памяти для функц. блоков/SFC)** в меню PLC (ПЛК).
2. Отобразится показанное ниже диалоговое окно Function Block/SFC Memory Statistics (Состояние памяти для функц. блоков/SFC) с количественными данными об использовании адресов.

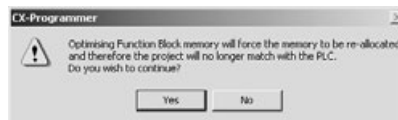
Memory Area	Available	Required	Remaining
FB Non Retained	896	27	869
FB Retained	120	0	120
FB Timer	1024	0	1024
FB Counter	1024	0	1024
SFC Bit	Share with F...	-	-
SFC Word	Share with F...	-	-

Общее количество слов в каждой области. Количество доступных для использования слов.
 Количество уже используемых слов.

Функция оптимизации памяти

При добавлении или удалении переменной происходит автоматическое перераспределение адресов между переменными в области памяти экземпляров. Адреса памяти, используемые для одного экземпляра, должны располагаться последовательно. Если после внесения изменений в переменные это требование соблюсти невозможно, для всех переменных выделяются другие последовательности адресов. В результате возникают неиспользуемые блоки адресов. Чтобы память расходовалась более рационально, можно воспользоваться функцией оптимизации памяти, которая перераспределяет память и устраняет неиспользуемые участки.

- 1,2,3... 1. Выделите экземпляр в окне сегмента лестничной диаграммы, щелкните правой кнопкой мыши и выберите команду **Memory Allocation – Function Block/SFC Memory – Optimize Function/SFC Memory (Распределение памяти – Память функц. блоков/SFC – Оптимизировать память функц. блоков/SFC)** в меню PLC (ПЛК). Будет отображено следующее диалоговое окно.



2. Щелкните по кнопке **ОК**. Распределение адресов в областях памяти экземпляров функционального блока будет оптимизировано.

3-2-8 Копирование и редактирование определений функциональных блоков

Для того чтобы скопировать или внести изменения в ранее созданное определение функционального блока, выполните следующие действия.

1. Выделите функциональный блок, который требуется скопировать, вызовите контекстное меню и выберите команду **Сору (Копировать)**.

2. Расположите указатель мыши на узле Function blocks в рабочей области проекта, вызовите контекстное меню щелчком правой кнопки мыши и выберите команду **Paste (Вставить)**.
3. Будет вставлено скопированное определение функционального блока (с тем же именем, что и у исходного определения, но со словом «сору» спереди).
4. Для изменения имени щелкните по функциональному блоку левой кнопкой мыши или щелкните правой кнопкой мыши и выберите **Rename (Изменить имя)** в контекстном меню.
5. Чтобы внести изменения в функциональный блок, щелкните по нему дважды кнопкой мыши.

3-2-9 Выяснение исходного определения функционального блока по экземпляру

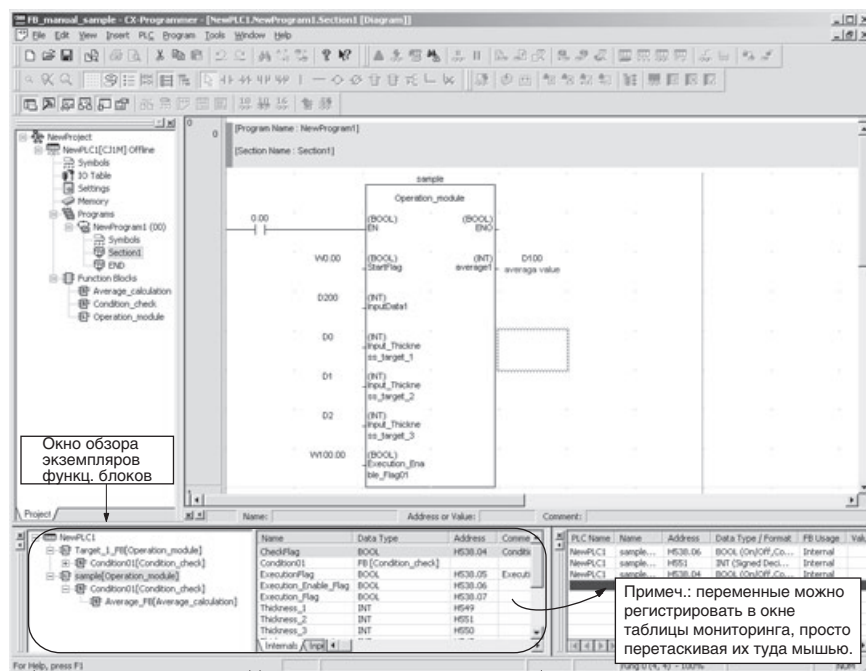
Для того чтобы выяснить, из какого именно определения функционального блока был создан тот или иной экземпляр, выполните следующие действия.

Щелкните дважды левой кнопкой мыши по экземпляру либо щелкните по экземпляру правой кнопкой мыши и выберите команду **To Lower Layer (На уровень ниже)** в контекстном меню. Отобразится определение функционального блока.

3-2-10 Отображение сведений об уровнях вложения и другой информации об экземплярах

Если в создаваемой программе имеются вложенные друг в друга функциональные блоки, можно воспользоваться командой **Windows – FB Instance Viewer (Окна – Обзор экземпляров функц. блоков)** в меню View (Вид), чтобы отобразить иерархию уровней вложения. Взаимосвязь между вложенными функциональными блоками отображается в виде древообразной структуры, на которой вызывающие функциональные блоки располагаются над вызываемыми функциональными блоками.

Окно обзора экземпляров функциональных блоков также предоставляет и другую информацию, например сведения об используемых переменных-массивах и о внутренних адресах, автоматически назначенных переменным (см. рисунок ниже). Переменные можно регистрировать в окне таблицы мониторинга, просто перетаскивая их туда мышью из списка переменных экземпляра.



В данной области отображается иерархия уровней вложения экземпляров (в скобках указываются имена определений функциональных блоков). На верхнем уровне располагается вызывающий блок, на нижнем — вызываемый. При наличии переменных типа «массив» или «таймер»/«счетчик» они также отображаются сразу под содержащим их экземпляром.

Здесь отображаются имена переменных, типы данных, адреса (выделенные внутренние адреса) и комментарии для переменных, используемых в экземпляре, который в данный момент выбран на дереве уровней слева.

Примеч.: переменные можно регистрировать в окне таблицы мониторинга, просто перетаскивая их туда мышью.

3-2-11 Отображение используемой памяти

Для функциональных блоков используются следующие области памяти.

Область памяти пользователя (UM)

В этой области содержится объектный код функциональных блоков.

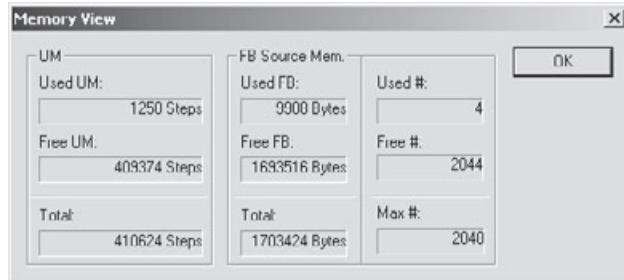
Память исходного кода функциональных блоков

В этой области хранится исходный код функциональных блоков, благодаря чему в CX-Programmer возможно отображение программ и таблиц переменных функциональных блоков.

В CX-Programmer предусмотрена возможность отображения объема памяти, используемого функциональными блоками. Порядок действия следующий:

1. Выберите **Memory View (Состояние памяти)** в меню View (Вид).
2. Отобразится показанное ниже диалоговое окно View Dialog (Состояние памяти).

Пример: CJ2H-CPU68



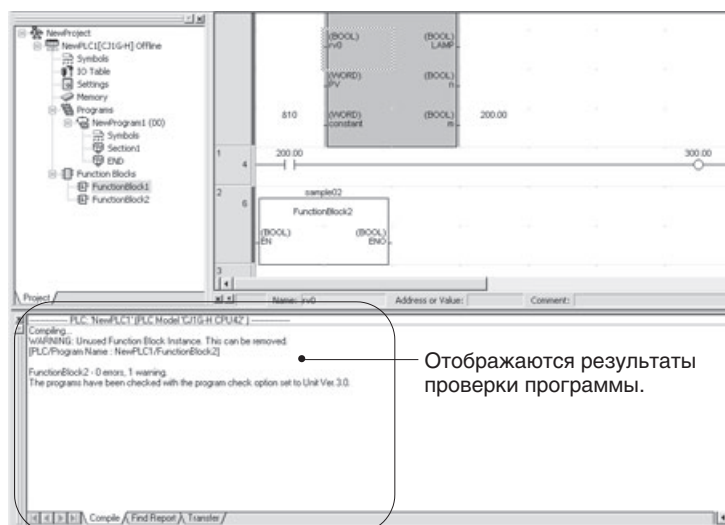
- Вид диалогового окна зависит от модели ПЛК. Более подробно функция отображения состояния памяти описана в руководстве *SX-Programmer — Руководство по работе* (Cat. No. W446).
- Способ вычисления количества шагов программы, занимаемого объектным кодом функциональных блоков, описан в разделе 2-9 *Количество шагов в программах функциональных блоков и время выполнения экземпляра*.

3-2-12 Компиляция определений функциональных блоков (проверка программы)

С целью проверки программы функционального блока можно выполнить компиляцию определения функционального блока. Выполните следующие действия.

- 1,2,3... Выделите определение функционального блока, щелкните правой кнопкой мыши и выберите **Compile (Компилировать)** в контекстном меню. (Другой способ: нажать клавиши **Ctrl + F7**.)

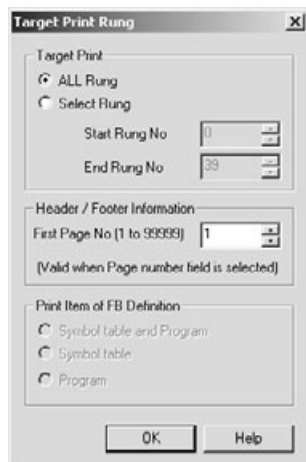
Функциональный блок будет скомпилирован, результаты проверки программы автоматически отобразятся на странице Compile Table (Таблица компиляции).



3-2-13 Вывод на печать определения функционального блока

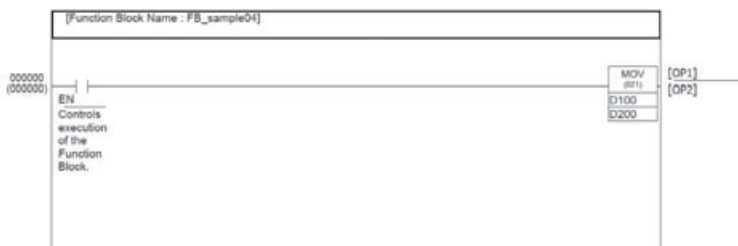
Чтобы вывести на печать определение функционального блока, выполните следующие действия.

- 1,2,3...
1. Двойным щелчком по определению функционального блока, который требуется вывести на печать, отобразите его таблицу переменных и программу, после чего выберите команду **Print (Печать)** в меню File (Файл). Откроется показанное ниже окно Target Print Rung (Выбор цепей для печати).



2. Выберите опцию **All Rung (Все цепи)** или **Select Rung (Выбранные цепи)**. В случае выбора опции Select Rung задайте номера первой и последней цепей для вывода на печать. Если ранее с помощью диалогового окна File (Файл) – Page Setup (Параметры страницы) в полях верхнего и нижнего колонтитулов был указан номер страницы, здесь может быть указан номер первой страницы.
3. Выберите компоненты функционального блока, которые должны быть выведены на печать.
 - Symbol table and program (Таблица символов и программа) (выбрано по умолчанию)
 - Symbol table (Таблица символов)
 - Program (Программа)
4. Щелкните кнопку **ОК**. Отобразится диалоговое окно Print (Вывод на печать). Настроив параметры принтера, количество печатаемых объектов и параметры страницы, щелкните кнопку ОК.
5. Ниже показан пример выведенной на печать таблицы переменных и программы функционального блока для языка LD.

Variable Type	Name	Type	Retained	AT	Initial Value	Comment
Inputs	EN	BOOL	No		FALSE	Controls execution of the Function Block.
Outputs	ENO	BOOL	No		FALSE	Indicates successful execution of the Function Block.



Примечание. Подробную информацию о настройке параметров печати можно найти в руководстве *CX-Programmer — Руководство по работе (W446)*.

3-2-14 Защита определений функциональных блоков с помощью пароля

Обзор

Доступ к определениям функциональных блоков в проекте может быть ограничен путем установки пароля. Может быть установлен один из двух следующих видов парольной защиты.

Парольная защита от записи и чтения

Данный вид парольной защиты запрещает как запись (изменение) определения функционального блока, так и отображение его содержимого на экране.

Для того чтобы установить защиту от чтения/записи, выберите *Prohibit writing and display (Запретить запись и отображение)* в поле *Protection type (Тип защиты)* в свойствах функционального блока. Данный вид защиты предотвращает непреднамеренное изменение/модификацию программы, а также не позволяет посторонним лицам незаконно знакомиться с содержимым программы.

Парольная защита только от записи

Данный вид парольной защиты запрещает внесение каких-либо изменений в определение функционального блока.

Для того чтобы установить защиту от записи, выберите *Prohibit writing (Запретить запись)* в поле *Protection type (Тип защиты)* в свойствах функционального блока. Данный вид защиты предотвращает непреднамеренное изменение/модификацию программы.

Установка парольной защиты

Эта операция может быть выполнена только в режиме офлайн.

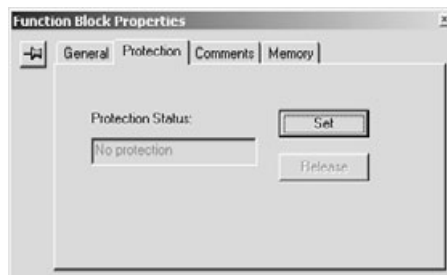
Парольная защита может быть применена как к отдельным определениям функциональных блоков, так и одновременно к нескольким определениям функциональных блоков.

Защита отдельного определения функционального блока

Для установки парольной защиты для отдельного определения функционального блока соблюдайте следующий порядок действий.

1,2,3...

1. В рабочей области проекта выделите определение функционального блока, щелкните правой кнопкой мыши и выберите **Properties (Свойства)** в контекстном меню. (Другой способ: выбрать команду **Properties (Свойства)** в меню View (Вид).)
2. Будет отображено диалоговое окно Function Block Properties (Свойства функционального блока). Откройте вкладку **Protection (Защита)** и щелкните кнопку **Set (Установить)**.






3. Отобразится диалоговое окно Function Block Protect Setting (Установка защиты для функционального блока). В поле *Protection Type (Тип защиты)* выберите требуемый уровень защиты.



В следующей таблице указаны функции, которые будут недоступны при установке парольной защиты соответствующего уровня.

Функция	Тип защиты	
	Защита от записи и отображения	Защита от записи
Отображение содержимого функционального блока	Запрещено	Разрешено
Вывод на печать содержимого функционального блока		Запрещено
Редактирование содержимого функционального блока		
Сохранение/загрузка файла библиотеки функциональных блоков	Разрешено	Разрешено

4. В поле *Password (Пароль)* диалогового окна Function Block Protect Setting введите пароль. Введите тот же пароль еще раз в поле проверки пароля и щелкните кнопку **Set (Установить)**.
Длина пароля не должна превышать 8 символов, допускается использовать только буквенно-цифровые символы.
5. Для определения функционального блока, защищенного паролем, на дереве проекта используется особый значок, вид которого зависит от типа установленной защиты:
 -  : Запрет записи и отображения (одинаковый для LD и ST)
 -  : Запрет записи (LD)
 -  : Запрет записи (ST)

Защита одновременно нескольких определений функциональных блоков

1,2,3...

Для установки парольной защиты одновременно для двух или большего числа определений функциональных блоков соблюдайте следующий порядок действий.

1. Щелкните правой кнопкой мыши по узлу **Function Blocks (Функциональные блоки)** в рабочей области проекта и выберите **Function Block Protection – Set (Защита функционального блока – Установить)** в контекстном меню.
2. Отобразится диалоговое окно Function Block Protection Collective Setting (Установка защиты для нескольких функциональных блоков). Выберите имена функциональных блоков, которые вы хотите защитить, выберите требуемый уровень защиты в поле *Protection Type (Тип защиты)*, введите пароль и щелкните кнопку **Set (Установить)**.



3. Для выбранных определений функциональных блоков будет установлена парольная защита.

Отмена парольной защиты

Эта операция может быть выполнена только в режиме офлайн.

Парольная защита может быть отменена (снята) как для отдельного определения функционального блока, так и одновременно для нескольких определений функциональных блоков.

Отмена парольной защиты для отдельного функционального блока

Для снятия парольной защиты с отдельного определения функционального блока соблюдайте следующий порядок действий.

1,2,3...

1. В рабочей области проекта выберите определение функционального блока, щелкните правой кнопкой мыши и выберите **Properties (Свойства)** в контекстном меню. (Другой способ: выбрать команду **Properties (Свойства)** в меню View (Вид).)
2. Будет отображено диалоговое окно Function Block Properties (Свойства функционального блока). Откройте вкладку **Protection (Защита)** и щелкните кнопку **Release (Отменить)**.
3. Отобразится диалоговое окно Function Block Protection Release (Отмена защиты для функционального блока). Введите пароль в поле *Password (Пароль)* и щелкните кнопку **Release (Отменить)**.
4. В случае ввода правильного пароля защита будет снята, и значок определения функционального блока на дереве проекта вернется к своему обычному виду.

Отмена парольной защиты одновременно для нескольких функциональных блоков

Для снятия парольной защиты одновременно с двух или большего числа определений функциональных блоков соблюдайте следующий порядок действий.

1,2,3...

1. Щелкните правой кнопкой мыши по узлу **Function Blocks (Функциональные блоки)** в рабочей области проекта и выберите **Function Block Protection – Release (Защита функционального блока – Отменить)** в контекстном меню.

2. Отобразится диалоговое окно Function Block Protection Collective Release (Отмена защиты для нескольких функциональных блоков). Выберите имена функциональных блоков, защиту с которых вы хотите снять, введите пароль и щелкните кнопку **Release (Отменить)**.
3. Если введенный пароль совпадет с паролями всех выбранных функциональных блоков, защита будет снята одновременно со всех определений функциональных блоков.

3-2-15 Сравнение функциональных блоков

Содержимое редактируемого функционального блока можно сравнить с содержимым функционального блока в ПЛК или в другом проекте, чтобы проверить их идентичность. Подробнее о функции сравнения программ можно прочитать в руководстве *CX-Programmer — Руководство по работе (W446)*.

3-2-16 Сохранение и повторное использование файлов функциональных блоков

Созданное определение функционального блока может быть индивидуально сохранено в файл библиотеки функциональных блоков (*.cxf) и впоследствии повторно использовано в других проектах.

Примечание

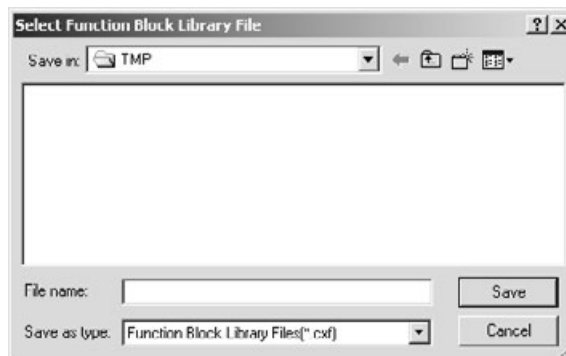
- (1) Прежде чем сохранять определение функционального блока в файл и использовать его в другом проекте, надо проверить его программу, выполнив компиляцию.
- (2) Если функциональный блок содержит вложенные функциональные блоки, при его сохранении в файл библиотеки функциональных блоков вместе с ним в этот файл также сохраняются определения всех вложенных (т. е. вызываемых) в него функциональных блоков.

Сохранение файла библиотеки функциональных блоков

Для сохранения определения функционального блока в файл библиотеки функциональных блоков используйте следующие действия.

1,2,3...

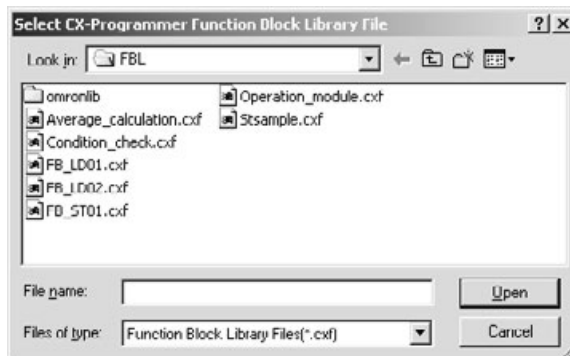
1. Выделите определение функционального блока, щелкните правой кнопкой мыши и выберите **Save Function Block to File (Сохранить функциональный блок в файл)** в контекстном меню. (Либо выберите **Function Block – Save Function Block to File (Функциональный блок – Сохранить функциональный блок в файл)** в меню File (Файл).)
2. Будет отображено следующее диалоговое окно. Введите имя файла. В качестве типа файла должно быть выбрано *Function Block Library Files (*.cxf)*.



Открытие файла библиотеки функц. блоков в другом проекте

Для открытия файла библиотеки функциональных блоков (*.cxf) в другом проекте соблюдайте следующий порядок действий.

- 1,2,3... 1. Щелкните правой кнопкой мыши по узлу **Function Blocks (Функциональные блоки)** в рабочей области проекта и выберите **Insert Function Block – From File (Вставить функциональный блок – Из файла)** в контекстном меню (или выберите **File – Function Block – Load Function Block from File (Файл – Функциональный блок – Загрузить функциональный блок из файла)**).
2. Будет отображено следующее диалоговое окно. Выберите файл библиотеки функциональных блоков (*.cxf) и щелкните кнопку **Open (Открыть)**.



3. Под значком **Function Blocks** будет автоматически вставлено определение функционального блока под названием **FunctionBlock1**.
4. Щелкните дважды по значку **FunctionBlock1**. Отобразятся таблица переменных и программа данного функционального блока.

3-2-17 Загрузка программ в модуль ЦПУ и их считывание из модуля ЦПУ

Прикладная программа, содержащая функциональные блоки, может быть загружена из CX-Programmer в модуль ЦПУ, с которым установлено соединение (т. е. в режиме онлайн). Также возможно считывание программы из подсоединенного модуля ЦПУ. Наконец, можно проверить, совпадает ли программа в CX-Programmer (на ПК) с программой в подключенном модуле ЦПУ. Если прикладная программа содержит функциональные блоки, индивидуальная загрузка задач в модуль ЦПУ невозможна (считывание возможно).

3-2-18 Мониторинг и отладка функциональных блоков

В данном разделе описаны возможные способы мониторинга и отладки программ функциональных блоков.

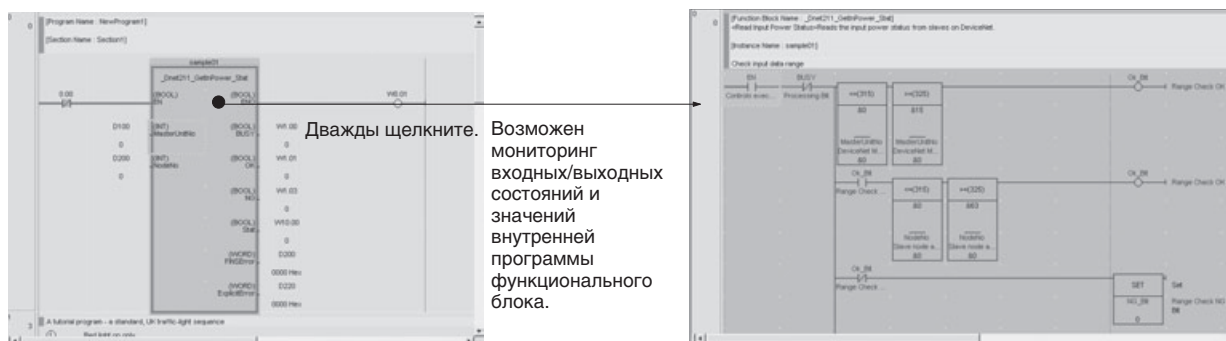
Мониторинг входов/выходов экземпляра в программе на языке LD

В CX-Programmer версии 6.0 и более поздних версий во время мониторинга прикладной программы также возможен мониторинг состояний битов и содержимого слов в окне программы экземпляра на языке LD. Для мониторинга входных и выходных битов и слов (I/O Bit Monitor) следует либо дважды щелкнуть по экземпляру функционального блока, либо щелкнуть правой кнопкой мыши по экземпляру и выбрать

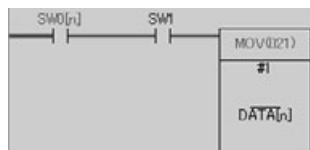
Monitor FB Ladder Instance (Мониторинг LD экземпляра функц. блока) в контекстном меню. В этом режиме можно наблюдать за значениями переменных и слов, изменять текущие значения, принудительно устанавливать/сбрасывать биты и контролировать переключение фронтов.

Примечание

- (1) В данном режиме невозможно изменять уставки таймеров/счетчиков.
- (2) В данном режиме невозможно изменять текущие значения и невозможно принудительно устанавливать/сбрасывать биты для входных-выходных переменных. Кроме того, если во входных-выходных переменных используются структуры данных, для членов логического типа (BOOL) этих структур данных невозможно отображать информацию о принудительно установленных/сброшенных состояниях (значки с изображением ключа).



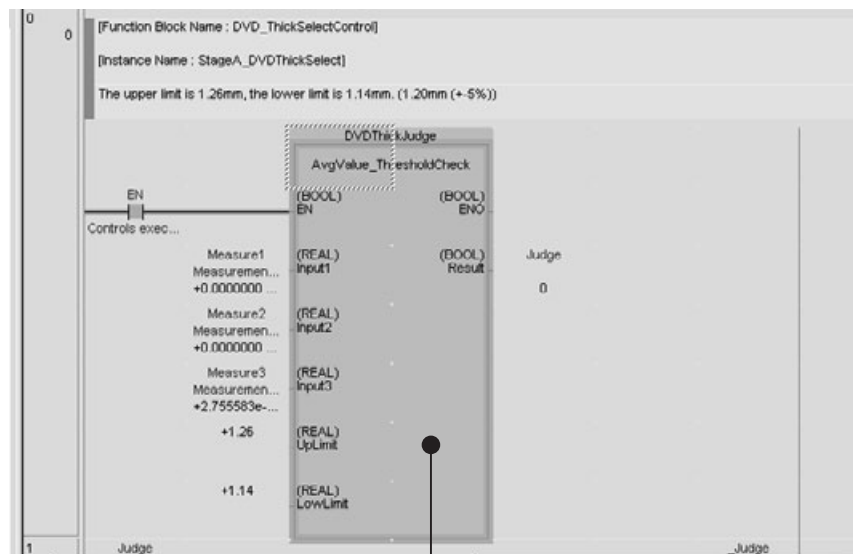
- (3) Если в функциональном блоке имеется переменная-массив и в аргументах этой переменной-массива используется символ, мониторинг текущего значения невозможен, если эта переменная-массив используется в качестве операнда во входном условии или в специальной команде. В этом случае входное условие или команда отображаются красным цветом.



Мониторинг переменных экземпляра в программе на языке ST

В CX-Programmer версии 6.1 и более поздних версий во время мониторинга прикладной программы возможен мониторинг программы экземпляра функционального блока в окне представления программы экземпляра на языке структурированного текста. Для мониторинга входных и выходных битов и слов (I/O Bit Monitor) следует либо дважды щелкнуть по экземпляру функционального блока, либо щелкнуть правой кнопкой мыши по экземпляру и выбрать **Monitor FB Instance (Мониторинг экземпляра функц. блока)** в контекстном меню.

Для возврата к первоначальному экземпляру щелкните правой кнопкой мыши по окну мониторинга программы на языке ST и выберите пункт **To Upper Layer (На уровень выше)** в контекстном меню.

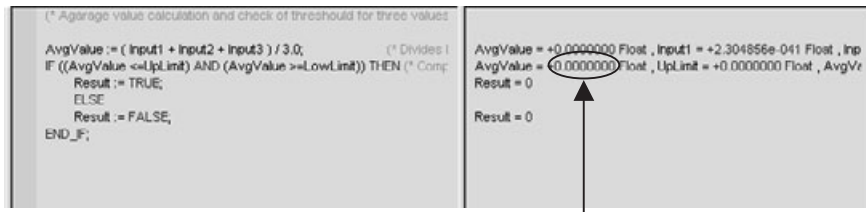


Щелкните правой кнопкой и выберите **To Upper Layer (Перейти на уровень выше)**.

Дважды щелкните по экземпляру. Отобразятся области мониторинга программы на языке ST и мониторинга переменных.

Левая часть: Окно мониторинга программы на языке ST

Правая часть: Окно мониторинга переменных программы на языке ST



Синим шрифтом отображаются текущие значения переменных.

В левой части окна (называемой окном мониторинга программы на языке ST) отображается программа на языке ST.

В правой части окна (называемой окном мониторинга переменных программы на языке ST или, просто, окном мониторинга переменных ST) отображаются значения переменных, используемых в программе на языке ST.

Здесь можно наблюдать за значениями переменных, изменять текущие значения, принудительно устанавливать/сбрасывать биты, копировать и вставлять переменные в окно таблицы мониторинга. (Эти операции подробно описаны ниже.)

Мониторинг переменных

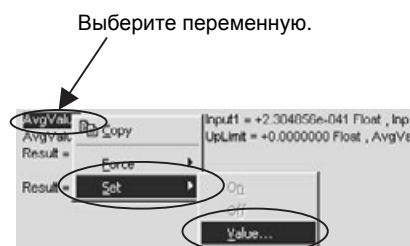
Значения переменных отображаются в окне мониторинга переменных ST шрифтом синего цвета.

Примечание

- (1) В случае использования модуля ЦПУ CJ2 невозможно получить текущее значение переменной типа TIMER, являющейся аргументом команды «0,1 мс ТАЙМЕР» или «0,01 мс ТАЙМЕР». Поэтому в окне мониторинга программы на языке ST на месте текущего значения отображается прочерк («-»).
- (2) Если текущее значение переменной типа TIMER, являющейся аргументом команды «0,1 мс ТАЙМЕР» или «0,01 мс ТАЙМЕР», используется где-либо еще кроме команд таймеров, корректное отображение текущего значения невозможно (поскольку оно будет недостоверным). Если текущее значение присваивается другой переменной, текущее значение последней также будет недостоверным.

Изменение текущих значений

Для изменения текущего значения выберите требуемую переменную в окне мониторинга переменных ST (при выделении отображается в инверсном виде), щелкните правой кнопкой мыши и выберите пункт **Set – Value (Задать – Значение)** в контекстном меню.



Отобразится диалоговое окно Set New Value (Ввод нового значения). Введите новое значение в поле *Value (Значение)*.

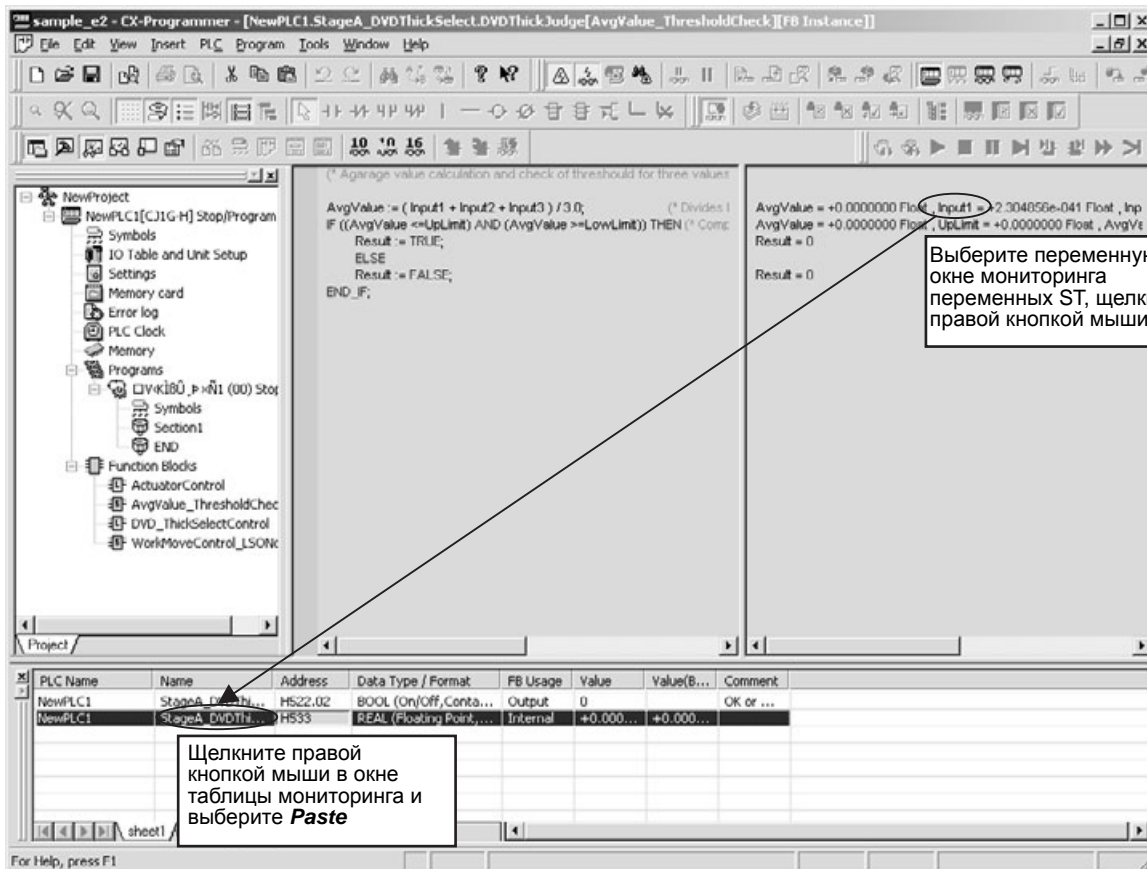
Принудительная установка/сброс битов

Чтобы выполнить принудительную установку/сброс бита или отменить принудительно установленные состояния, выберите требуемую переменную в окне мониторинга переменных ST (при выделении отображается в инверсном виде), щелкните правой кнопкой мыши и выберите пункт **Force – On (Принудительно – Включить)**, **Force – Off (Принудительно – Выключить)**, **Force – Cancel (Принудительно – Отменить)** или **Force – Cancel All Forces (Принудительно – Отменить все)** в контекстном меню.

Копирование и вставка в окне таблицы мониторинга

1,2,3...

1. Для того чтобы скопировать переменную и вставить ее в окно таблицы мониторинга, выберите требуемую переменную в окне мониторинга переменных ST (при выделении отображается в инверсном виде), щелкните правой кнопкой мыши и выберите пункт **Сору (Копировать)** в контекстном меню.
2. Правой кнопкой мыши щелкните по окну таблицы мониторинга и выберите пункт **Paste (Вставить)** в контекстном меню.



Отображение программы определения функционального блока

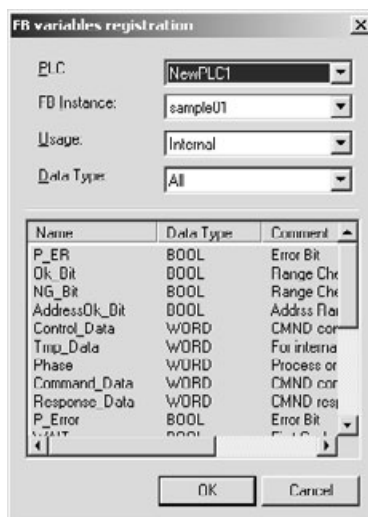
Во время мониторинга для экземпляра можно отобразить программу определения функционального блока. Для этого выполните следующие действия.

- 1,2,3... Щелкните правой кнопкой мыши по экземпляру и выберите **To Lower Layer (На уровень ниже)** в контекстном меню. Отобразится определение функционального блока.

Мониторинг переменных экземпляра в окне таблицы мониторинга

Для мониторинга экземпляра в окне таблицы мониторинга используйте следующий порядок действий.

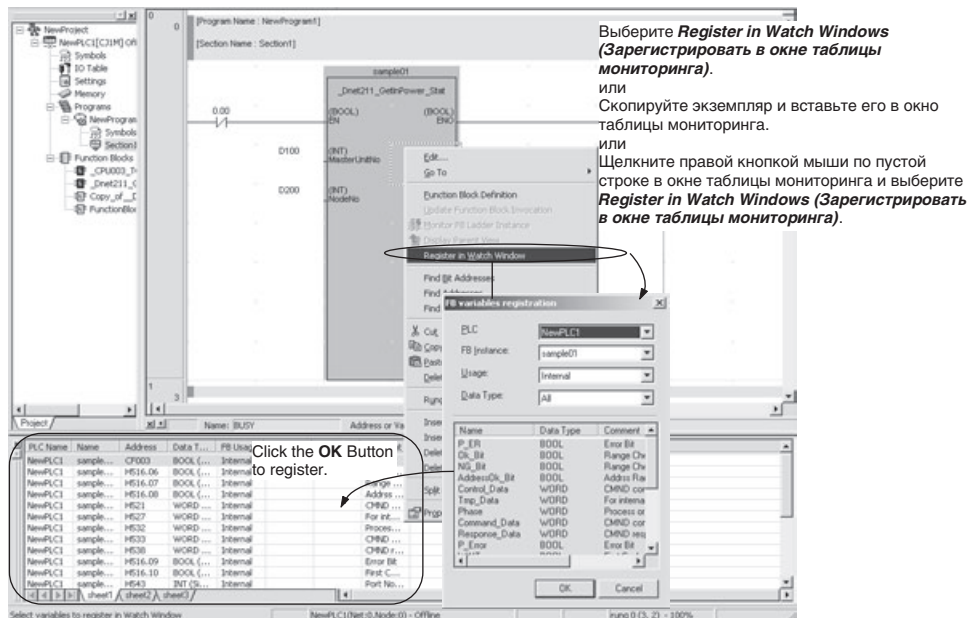
- 1,2,3... 1. Выберите **View – Window – Watch (Вид – Окно – Таблица мониторинга)**.
Отобразится окно таблицы мониторинга.
2. Вызовите диалоговое окно **FB variables registration (Регистрация переменных функц. блока)**, используя один из трех следующих способов.
- Щелкните правой кнопкой мыши по экземпляру и выберите **Register in Watch Window (Зарегистрировать в окне таблицы мониторинга)** в контекстном меню.
 - Скопируйте экземпляр и вставьте его в окно таблицы мониторинга.
 - Щелкните правой кнопкой мыши по пустой строке в окне таблицы мониторинга и выберите **Register in Watch Windows (Зарегистрировать в окне таблицы мониторинга)** в контекстном меню.



3. Выберите тип переменной в поле **Usage (Использование)** и тип данных в поле **Data Type (Тип данных)**. Также может быть выбран экземпляр в поле **FB Instance (Экземпляр функц. блока)**. По умолчанию в поле **Usage** выбрано **N: Internal (Внутренняя переменная)**. Другие возможные значения: **I: Input (Входная переменная)**, **O: Output (Выходная переменная)** и **E: External (Внешняя переменная)**.

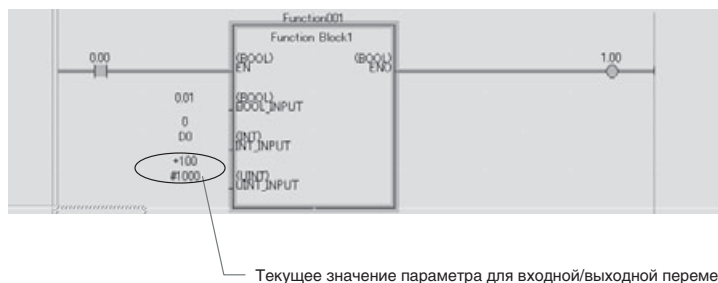
По умолчанию в поле **Data Type** выбрано **A: All (Все)**. Также может быть выбран тип данных **BOOL** или **INT**.

- Щелкните по кнопке **ОК**. Выбранная переменная будет зарегистрирована в окне таблицы мониторинга. Пример отображения значений в окне таблицы мониторинга показан на рисунке ниже.



Мониторинг входных и выходных переменных в экземплярах

Текущие значения входных и выходных переменных (параметров) отображаются под параметрами.



Имитация выполнения программ на языке LD/ST в экземплярах

В программном обеспечении CX-One версии 1.1 (CX-Programmer версии 6.1) и более поздних версий предусмотрена функция имитации режима выполнения. С помощью этой функции можно смоделировать выполнение программы экземпляра функционального блока (на языке LD или ST) непосредственно на ПК. Поддерживаются функции пошагового выполнения и выполнения до точки останова.

Для возврата к первоначальному экземпляру щелкните правой кнопкой мыши по окну мониторинга программы на языке ST и выберите пункт **To Upper Layer (На уровень выше)** в контекстном меню.

Включение функции имитации

Для включения функции имитации выполните следующие действия.

- 1,2,3...** 1. Откройте программу, содержащую экземпляр, который требуется отладить.
2. Выберите **View – Toolbars (View – Панели инструментов)** и выберите опцию **Отладка программы** на вкладке **Toolbars (Панели инструментов)**.
3. Выберите **Work Online Simulator (Соединение с Имитатором)** в меню **PLC (ПЛК)** в CX-Programmer и загрузите программу в компонент CX-Simulator, работающий на ПК.

Примечание. Шаги 2 и 3 можно выполнять в произвольном порядке.

Пошаговое выполнение (Step Run)

Программа выполняется пошагово (покомандно). Когда выполнение экземпляра остановлено, эта функция переходит к первому шагу (команде) программы данного экземпляра (на языке LD или ST).

Программа экземпляра может быть выполнена в пошаговом (Step Run) или непрерывном (Continuous Step Run) (см. примечание) режиме.

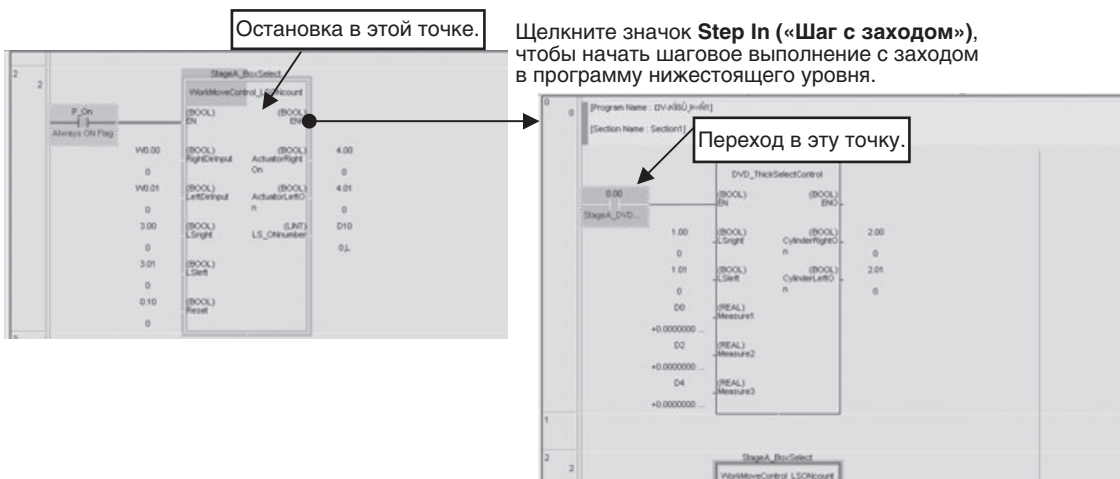
Примечание. Для режима непрерывного выполнения (Continuous Step Run) необходимо задать длительность выполнения. Для этого выберите в CX-Programmer команду **Tools – Options (Сервис – Настройка)**, откройте вкладку PLC (ПЛК) и задайте требуемое значение в поле *Continuous Step Interval (Интервал непрерывного выполнения)*.

Шаг с заходом (Step In)

Чтобы начать выполнение программы экземпляра (на языке LD/ST) в режиме пошагового выполнения (Step Run), используйте следующий порядок действий.

- 1,2,3...
1. Остановите выполнение программы экземпляра (см. примеч.).
 2. Щелкните по значку **Step In (Шаг с заходом)** или выберите команду **Tools – Simulation – Mode – Step In (Сервис – Имитация – Режим – Шаг с заходом)**.

Пример: выполнение шага с заходом во внутреннюю программу экземпляра на языке LD



Пример: выполнение шага с заходом во внутреннюю программу экземпляра на языке ST

Щелкните значок **Step In** («Шаг с заходом»), чтобы начать шаговое выполнение с заходом в программу нижестоящего уровня.

В программе на языке ST слева от позиции остановки отображается стрелка.

Остановка в этой точке.

Переход в эту точку.

Примечание. Во время выполнения программы в точке, расположенной за пределами экземпляра функционального блока, выполнение происходит так же, как при обычном пошаговом выполнении (Step Run).

Шаг с выходом (Step Out)

Для остановки пошагового выполнения программы экземпляра (на языке LD или ST) (режим Step Run) и возврата к программе вышестоящего уровня (вызвавшей данный экземпляр) выполните следующие действия.

- 1,2,3...
1. Во время операции пошагового выполнения расположите указатель мыши на любой точке остановки в пределах экземпляра.
 2. Щелкните по значку **Step Out** (**Шаг с выходом**) или выберите **Tools – Simulation – Mode – Step Out** (**Сервис – Имитация – Режим – Шаг с выходом**).

Пример: Возврат из программы на языке ST в вызывающую программу или вызывающий экземпляр

Остановка в этой точке.

Щелкните значок **Step Out** («Шаг с выходом») для возврата в вызывающую программу.

Переход в эту точку.

Примечание. Команда Step Out (Шаг с выходом) может быть выполнена только для программы внутри экземпляра (на языке LD/ST).

Цвета отображения курсора при использовании функции имитации

Цвет курсора (или стрелки в программе на языке ST) показывает, не остановлено ли в данный момент выполнение программы, а также указывает причину остановки выполнения.

Операция отладки	Цвет (по умолчанию)	Состояние выполнения программы	Подробное описание
Пошаговое выполнение (Step Run) или непрерывное выполнение (Continuous Step Run)	Розовый	Выполнение приостановлено имитатором	Ожидается выполнение следующего шага в режиме пошагового выполнения или нажата кнопка Pause (Пауза) .
	Обычный цвет	Программа не выполняется из-за действия блокировки или другой функции.	Шаг не выполняется вследствие действия такой команды, как IL, MILR/MILH, JMP0 или FOR/BREAK.
Точка остановки	Синий	Точка остановки имитатора	Выполнение приостановлено в точке остановки (break point).

Примечание

- (1) Если установлен флажок **Tools – Simulation – Always Display Current Execution Point (Сервис – Имитация – Всегда отображать текущую точку выполнения)**, во время пошагового или непрерывного выполнения программа экземпляра автоматически пролистывается, так что в конце выполнения на экране всегда отображается точка, в которой было приостановлено выполнение программы.
- (2) Цвет курсора (или стрелки в программе на языке ST), используемый по умолчанию для индикации приостановки выполнения программы в окне функции имитации выполнения, можно изменить. Для этого выберите **Tools – Options (Сервис – Настройка)** и откройте вкладку **Appearance (Оформление)**. Выберите **Pause Simulator (Приостановка в режиме имитации)**, **Simulator Instruction Break (Приостановка в точке остановки)** или **Simulator IO Break (Остановка по внешней причине)** и установите требуемый цвет для этого состояния.

Выполнение программы экземпляра до точки остановки

В программе экземпляра можно разместить точку остановки (break point). При достижении точки остановки выполнение программы автоматически приостанавливается. (В этом случае использовать операцию Step In (Шаг с заходом) невозможно.)

Примечание.

Точка остановки, установленная в программе экземпляра, действительна только для этого экземпляра. (В других экземплярах, созданных из этого же определения функционального блока, она не действует.)

3-2-19 Редактирование определений функциональных блоков в режиме онлайн

Редактирование программ определений функциональных блоков (на языке LD или ST) в том числе возможно и непосредственно во время выполнения прикладной программы в модуле ЦПУ, если последний работает в режиме «Мониторинг». Благодаря этому можно производить отладку и правку определений функциональных блоков в круглосуточно работающих системах, в которых остановка работы недопустима.

Для редактирования определений функциональных блоков в режиме онлайн требуется CX-Programmer версии 7.0 или выше (т. е. CX-One версии 2.0 или выше), а также модуль ЦПУ серии CS/CJ с версией модуля 4.0 или выше (см. примеч.) или модуль ЦПУ серии CJ2.

Данная функция недоступна при имитации режима выполнения в CX-Simulator.

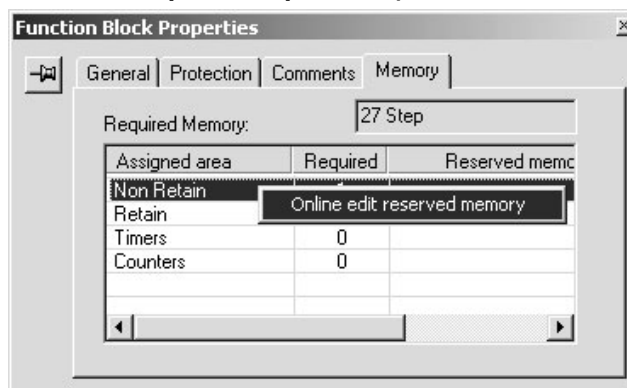
Примечание. При работе с модулями ЦПУ серии CS/CJ с версией модуля 3.0 возможности онлайн-редактирования ограничены.

- Возможно изменение входных и выходных параметров экземпляров, изменение команд вне экземпляров и удаление экземпляров.
- Невозможно добавление экземпляров, изменение имен экземпляров, внесение изменений в таблицы переменных или в программы определений функциональных блоков.

Резервирование памяти для добавления внутренней переменной при онлайн-редактировании

Для того чтобы внутренняя переменная могла быть добавлена в таблицу переменных определения функционального блока, необходимо заранее зарезервировать память в соответствии с размером добавляемой переменной. Для этих целей используется отдельная область памяти, не связанная с областью памяти экземпляров функционального блока (адреса которой распределяются между переменными автоматически). Ниже описан порядок действий для резервирования памяти перед началом редактирования функционального блока в режиме онлайн.

- 1,2,3...
1. В рабочей области проекта щелкните правой кнопкой мыши по редактируемому определению функционального блока и выберите **Properties (Свойства)** в контекстном меню.
 2. Откройте вкладку **Memory (Память)**, щелкните правой кнопкой мыши по области, для которой требуется зарезервировать память, и выберите **Online edit reserved memory (Зарезервировать память для онлайн-редактирования)** в контекстном меню.



3. Введите требуемый объем резервируемой памяти в каждом поле отобразившегося диалогового окна Memory Size Edit for FB Online Edit (Изменение размера памяти для онлайн-редактирования функц. блока).

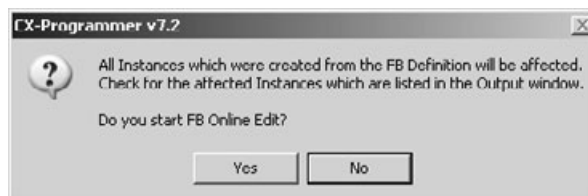


Редактирование и загрузка определения функционального блока

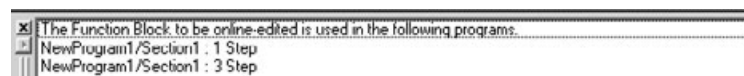
- 1,2,3... 1. В режиме установленной связи с ПЛК (т. е. в режиме онлайн) щелкните правой кнопкой мыши по определению функционального блока в рабочей области проекта (см. примечание) и выберите команду **FB Online Edit – Begin (Редактировать функц. блок онлайн – Начать)** в контекстном меню.

Примечание. Онлайн-редактирование также может быть запущено из окна определения функционального блока, окна мониторинга программы экземпляра (на языке LD/ST) или из команды вызова функционального блока (из обычной программы на языке LD или из программы функционального блока на языке LD).

Прежде чем будет начато онлайн-редактирование функционального блока, отобразится следующее диалоговое окно.



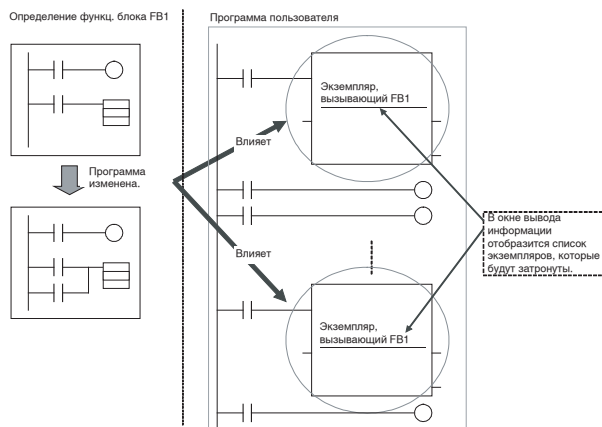
Одновременно с этим в окне вывода информации отобразится список экземпляров, которые будут затронуты.



- Примечание.** Влияние изменений в определении функционального блока на экземпляры
При внесении изменений в определение функционального блока также изменяется содержимое всех экземпляров, вызывающих это определение функционального блока.

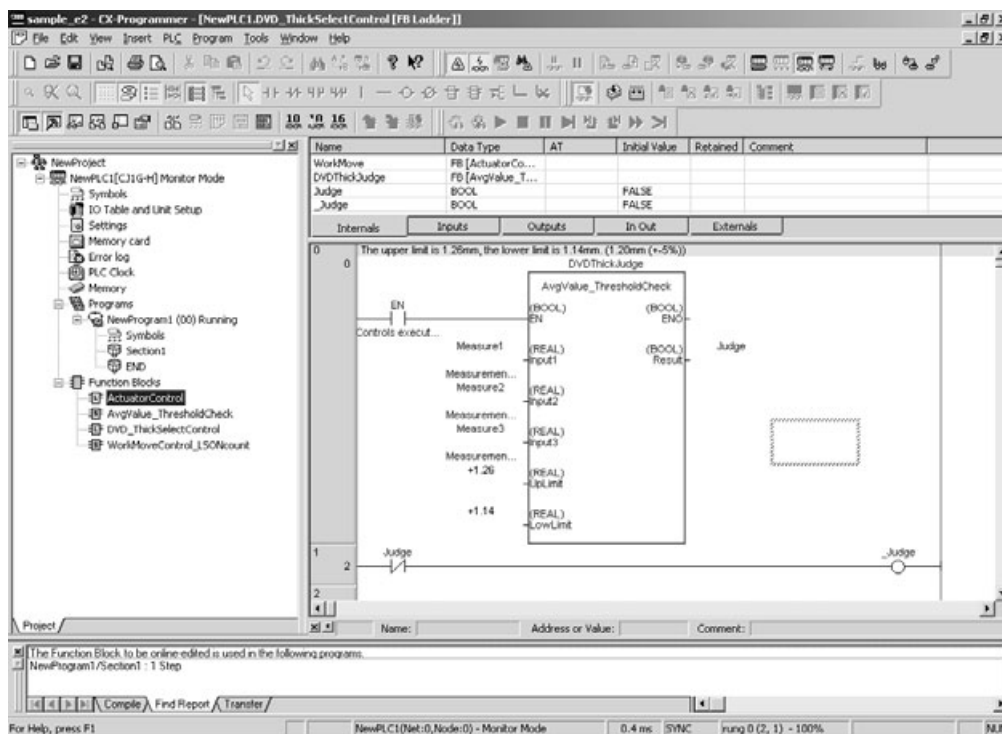
Сказанное иллюстрирует следующий рисунок.

Пример

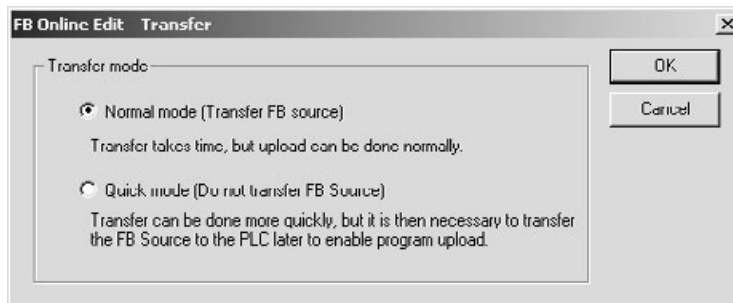


Прежде чем изменять определения функциональных блоков, проанализируйте влияние этих изменений на работу всей программы в целом.

- Щелкните кнопку **Yes (Да)**. Отобразится доступное для изменения содержимое определения функционального блока.



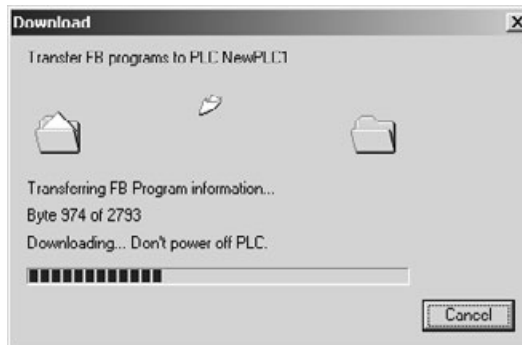
3. Завершив правку содержимого определения функционального блока, выберите **FB online Edit – Send Changes (Редактировать функц. блок онлайн – Передать изменения)**. Откроется показанное ниже диалоговое окно FB Online Edit – Transfer (Редактирование функц. блока онлайн – Загрузка).



4. Выберите один из следующих режимов загрузки программы и щелкните кнопку **Yes (Да)**.
 - Normal Mode (Обычный режим)
 - Quick Mode (Быстрый режим)

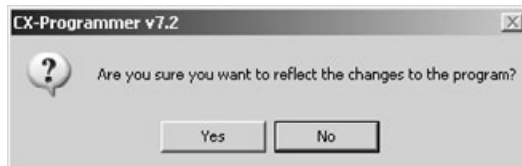
Подробное описание режимов загрузки программы см. в разделах *Режимы загрузки* на стр. 150 и *Выбор режима загрузки* на стр. 150.

Измененное определение функционального блока будет загружено в буферную память модуля ЦПУ. В процессе загрузки будет отображаться следующее диалоговое окно.



(В это время в модуле ЦПУ пока еще используется прежнее определение функционального блока.)

По завершении загрузки отобразится следующее диалоговое окно.



(В это время в модуле ЦПУ пока еще используется прежнее определение функционального блока.)

5. Щелкните кнопку **Yes (Да)**. В программу пользователя в модуле ЦПУ будет записано обновленное определение функционального блока из буферной памяти модуля ЦПУ. (В случае щелчка по кнопке **No (Нет)** обновленное определение функционального блока будет удалено из буферной памяти, программа пользователя изменена не будет.)

И в том и в другом случае программа вернется в состояние, в котором редактирование определений функциональных блоков невозможно. Для редактирования другого определения функционального блока следует вновь выбрать **FB online Edit – Begin (Редактировать функц. блок онлайн – Начать)** и начать процедуру онлайн-редактирования с самого начала.

Режимы загрузки

Обычный режим

В режиме обычной загрузки (Normal Mode) в модуль ЦПУ вместе с объектным кодом передается исходный код программы. Из-за большого объема передаваемых данных загрузка в обычном режиме длится дольше, чем в быстром режиме. До полного завершения операции загрузки выполнение других операций редактирования и загрузки невозможно.

Примечание. Если в настройках была выбрана опция *Display confirmation of FB online edit changes (Требовать подтверждения изменений при онлайн-редактировании функц. блока)*, после завершения передачи исходного кода отображается диалоговое окно с просьбой подтвердить запись переданного исходного кода в память пользователя в модуле ЦПУ.

Быстрый режим

В режиме быстрой загрузки (Quick Mode) в модуль ЦПУ передается только объектный код программы. Исходный код не передается, благодаря чему загрузка выполняется быстрее, чем в обычном режиме. После передачи объектного кода либо 1) выберите **Program – Transfer FB Source (Программа – Передать исходный код функц. блока)** для загрузки исходного кода, либо 2) выполните загрузку исходного кода, следуя указаниям, которые отображаются в диалоговом окне при выходе в режим офлайн.

После загрузки объектного кода внизу окна отображается желтым цветом надпись «FB Source», означающая, что исходный код еще не был загружен. После того как исходный код будет загружен, это сообщение исчезнет.

Выбор режима загрузки

Как правило, измененные определения функциональных блоков следует загружать в обычном режиме. Если загрузка длится очень долго, до начала загрузки нужно до максимума повысить скорость передачи (бит/с). Если загрузка по-прежнему занимает много времени, что сильно задерживает отладку программы при непрерывном онлайн-редактировании, можно в виде исключения использовать быстрый режим, однако нужно хорошо понимать ограничения этого режима, описанные в примечании ниже (*Ограничения в режиме быстрой загрузки*).

Ниже приведены ориентировочные значения времени загрузки для восьми определений функциональных блоков, исходный код которых в сумме занимает 8 Кбайт для всех определений и всех экземпляров.

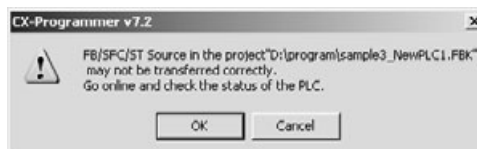
	Обычный режим	Быстрый режим
При 115,2 Кбит/с:	5 с	1 с
При 19,2 Кбит/с:	10 с	2 с

Примечание. Ограничения в режиме быстрой загрузки
Программа, содержащая функциональные блоки, не может быть корректно считана из модуля ЦПУ в CX-Programmer, если в модуль ЦПУ не был загружен исходный код всех определений функциональных

блоков. Воспользовавшись режимом быстрой загрузки для передачи измененных определений функциональных блоков, обязательно загрузите исходный код этих определений позже, выбрав команду *Program – Transfer FB Source (Программа – Передать исходный код функц. блока)*. Даже если исходный код не был загружен в модуль ЦПУ пользователем, он будет загружен туда автоматически при выходе из режима онлайн, если только в работе компьютера или CX-Programmer не произойдет какой-либо сбой до завершения загрузки исходного кода. В последнем случае считывание программы из модуля ЦПУ может оказаться невозможным (см. примеч.).

Примечание. Исходный код, возможно, удастся загрузить, даже если возникнет описанная выше проблема.

- a. При следующем запуске программы CX-Programmer отобразится диалоговое окно, показанное ниже.



- b. Щелкните по кнопке **ОК**.
- c. Установите связь с модулем ЦПУ (т. е. перейдите в режим онлайн), в который были переданы данные в режиме загрузки.
- d. При переходе в режим онлайн запускается резервная копия проекта СХР (созданная автоматически), и отображается следующее диалоговое окно. (Содержание сообщения: «Проверить возможность загрузки сохраненной копии исходного кода FB/SFC/ST ? Если будет нажата кнопка «Нет», корректное считывание программы из ПЛК в режиме онлайн будет невозможно.»)



- e. Щелкните кнопку **Yes (Да)** и далее следуйте указаниям диалоговых окон. Копию исходного кода, автоматически сохраненную на компьютере, можно сравнить с объектным кодом в модуле ЦПУ и, в случае их совпадения, загрузить ее в модуль ЦПУ.

Примечание.

Исходный код и объектный код

Перед загрузкой программы в модуль ЦПУ CX-Programmer, как правило, транслирует исходный код программы в объектный код (чтобы он мог быть исполнен в модуле ЦПУ) и передает в модуль ЦПУ объектный код программы вместе с исходным кодом. В модуле ЦПУ исходной код и объектный код программы записываются в память пользователя, а также во встроенную флеш-память. Передача и восстановление программы в CX-Programmer возможны лишь при условии наличия одновременно исходного и объектного кодов в модуле ЦПУ.

Отмена изменений в определениях функциональных блоков

Для того чтобы отменить все изменения, произведенные в определении функционального блока, выберите **FB online Edit – Cancel (Редактировать функц. блок онлайн – Отменить)**. Определение функционального блока не будет загружено в модуль ЦПУ и будет возвращено к своему первоначальному виду.

Влияние на работу модуля ЦПУ

Выполнение онлайн-редактирования в режиме «Мониторинг» влияет на работу модуля ЦПУ следующим образом: 1) Во время записи в модуль ЦПУ длительность цикла модуля ЦПУ возрастает в несколько раз. 2) Длительность цикла возрастает снова, когда резервная копия результатов онлайн-редактирования сохраняется во встроенную флеш-память. (Во время этих операций на лицевой панели модуля ЦПУ мигает индикатор ВКУР, а в CX-Programmer отображается ход выполнения операций.)

Макс. увеличение времени цикла при онлайн-редактировании

В следующей таблице приведены максимальные пределы возрастания длительности цикла при онлайн-редактировании.

Во время онлайн-редактирования	Во время резервного копирования
Макс. 12 мс	4% от длительности цикла

Примечание.

Контрольная длительность цикла

Проследите, чтобы при перезаписи программы во время онлайн-редактирования в режиме «Мониторинг» не оказалась превышена контрольная длительность цикла, заданная в настройках ПЛК. В противном случае произойдет ошибка превышения времени цикла и модуль ЦПУ прекратит работу. При возникновении этой ошибки следует переключить модуль ЦПУ в режим «Программирование», а затем вернуть его в режим «Мониторинг» или «Выполнение» для возобновления работы.

Предупреждение

Если используется синхронная работа модулей, увеличение времени синхронной обработки, вызванное редактированием в режиме онлайн, изменяет время работы непредсказуемым образом. Прежде чем приступить к редактированию в режиме онлайн, убедитесь в том, что увеличение времени синхронной обработки не повлияет отрицательно на работу главной и ведомой осей.

Ограничения при онлайн-редактировании функциональных блоков

В отношении редактирования определений функциональных блоков в режиме онлайн действуют следующие ограничения.

- Онлайн-редактирование невозможно для определений функциональных блоков, объем которых превышает 4К шагов (кроме модулей ЦПУ серии CJ2).
- В течение одной операции онлайн-редактирования программа функционального блока не может быть увеличена или уменьшена более чем на 0,5 К шагов (кроме модулей ЦПУ серии CJ2).
- Добавление или удаление входных, выходных и входных-выходных переменных невозможно.
- Добавление новых экземпляров функциональных блоков невозможно.
- Изменение имен экземпляров невозможно.

- Возможно добавление внутренних переменных, изменение комментариев к внутренним переменным, удаление внутренних переменных из таблицы переменных определения функционального блока. Для добавления внутренней переменной, однако, требуется заранее зарезервировать память. Подробное описание смотрите в разделе *Резервирование памяти для добавления внутренней переменной при онлайн-редактировании* на стр. 146.
- После завершения онлайн-редактирования прежние состояния флагов для всех выполняемых по фронту команд (DIFU(013), команды с префиксом @, DIFD(014) и команды с префиксом %) инициализируются (т. е. сбрасываются).
- Закончив онлайн-редактирование, не выключайте напряжение питания ПЛК до тех пор, пока модуль ЦПУ не завершит сохранение резервной копии данных во встроенную флеш-память (т. е. пока не перестанет мигать индикатор ВКУР). Если питание будет выключено до завершения резервного копирования, резервная копия данных создана не будет и программа возвратится к состоянию, в котором она находилась до начала выполнения онлайн-редактирования.

**Часть 2:
Структурированный
текст (ST)**

РАЗДЕЛ 4

Введение в структурированный текст

Данный раздел содержит общие сведения о функциях программного обеспечения CX-Programmer, предназначенных для создания программ на языке структурированного текста, и поясняет функции, которые отсутствуют в версиях CX-Programmer, не поддерживающих язык структурированного текста.

4-1	Язык структурированного текста	158
4-1-1	Общие сведения	158
4-2	Характеристики CX-Programmer	159
4-2-1	Совместимость моделей ПЛК с языком программирования ST (задачи на языке ST) .	159
4-2-2	Характеристики.	160

4-1 Язык структурированного текста

Данный раздел содержит общую информацию о возможностях использования языка структурированного текста для программирования задач в программном обеспечении CX-Programmer. Сведения об особенностях применения языка структурированного текста для создания других программ (функциональных блоков или SFC) содержатся в следующих разделах.

- Использование программ на языке ST в экземплярах функциональных блоков:
см. *Часть 1: Функциональные блоки* в настоящем руководстве.
- Использование программ на языке ST в последовательных функциональных схемах (SFC):
см. *CX-Programmer — Руководство по работе: SFC (W469)*.

4-1-1 Общие сведения

Язык структурированного текста (ST) — это язык программирования высокого уровня, предназначенный для промышленных средств автоматизации (преимущественно, для ПЛК) и описанный в стандарте МЭК 61131-3. Благодаря наличию стандартных выражений, операторов и функций язык структурированного текста превосходно подходит для программирования различных математических операций, которые довольно сложно реализовать на языке релейно-контактных схем. (Однако язык ST подходит не для всех операций, которые могут быть запрограммированы на языке LD.)

Язык структурированного текста, поддерживаемый в CX-Programmer версии 7.2 и более поздних версий, соответствует стандарту МЭК 61131-3, и создаваемые на нем программы могут назначаться для задач прикладной программы модуля ЦПУ.

Что касается модели ЦПУ, то это может быть модуль ЦПУ серии CS/CJ с версией прошивки 4.0 и выше либо модуль ЦПУ серии CJ2.

Ниже перечислены основные возможности языка структурированного текста.

- Синтаксис языка ST включает множество управляющих выражений и конструкций, включая выражения для организации цикла и выражения выбора (IF-THEN-ELSE); большое число различных операторов, в том числе арифметические операторы, операторы сравнения и логические операторы (AND/OR); множество математических функций, например функции для извлечения и объединения текстовых строк, функции для выполнения операций с картой памяти, функции передачи текстовых строк и тригонометрические функции.

- Процесс написания программы на языке ST в целом очень схож с написанием программ на таких высокоуровневых языках, как Паскаль или Си. Возможность добавления комментариев делает программу более понятной и удобной для чтения.

```

Программа на языке ST
IF score > setover THEN      (*Если score>setover*)
  underNG := FALSE;        (*Выключение underNG*)
  OK := FALSE;             (*Выключение OK*)
  overNG := TRUE;         (*Включение overNG*)

ELSIF score < setunder THEN (*Если score=<setover и score < setunder*)
  overNG := FALSE;        (*Выключение overNG*)
  OK := FALSE;           (*Выключение OK*)
  underNG := TRUE;       (*Включение underNG*)

ELSE                          (*Если setover>score>setunder*)
  underNG := FALSE;       (*Выключение underNG*)
  overNG := FALSE;       (*Выключение overNG*)
  OK := TRUE;             (*Включение OK*)

END_IF;                      (*Конец выражения IF*)
    
```

- Программы на языке ST могут считываться и загружаться точно так же, как и обычные программы (не поддерживается лишь индивидуальное считывание/загрузка задач программы на языке ST).
- В программе, написанной на языке ST, могут вызываться функциональные блоки (написанные на языке LD или ST).
- Поддерживаются одномерные массивы переменных, что упрощает обработку данных в прикладных программах.

4-2 Характеристики CX-Programmer

В данном разделе приведены требования к оборудованию и характеристики программного обеспечения CX-Programmer, связанные с программированием на языке структурированного текста (создание задач на языке ST). Общие сведения о характеристиках программы CX-Programmer и требованиях к оборудованию можно найти в руководстве *CX-Programmer — Руководство по работе (W446)*.

- Подробную информацию о характеристиках программы CX-Programmer и требованиях к оборудованию для других программ (функциональных блоков или SFC) см. в разделе *Часть 1: Функциональные блоки* в настоящем руководстве или в руководстве *CX-Programmer — Руководство по работе: SFC (W469)*.

4-2-1 Совместимость моделей ПЛК с языком программирования ST (задачи на языке ST)

Модели программируемых контроллеров, поддерживающие создание задач на языке структурированного текста, перечислены в следующей таблице.

Модель ПЛК	Модель модуля ЦПУ
CJ2H	CJ2H-CPU68/67/66/65/64/68-EIP/67-EIP/66-EIP/65-EIP/64-EIP
CJ2M	CJ2M-CPU11/12/13/14/15/31/32/33/34/35
CS1G-H версии 4.0	CS1G-CPU45H/44H/43H/42H

Модель ПЛК	Модель модуля ЦПУ
CS1H-H версии 4.0	CS1H-CPU67H/66H/65H/64H/63H
CJ1G-H версии 4.0	CJ1G-CPU45H/44H/43H/42H
CJ1H-H версии 4.0	CJ1H-CPU67H/66H/65H/67H-R/66H-R/65H-R/64H-R
CJ1M версии 4.0	CJ1M-CPU23/22/21/13/12/11

4-2-2 Характеристики

Параметр	Описание
Языки программирования, которые могут использоваться для задач.	SFC, LD или ST (возможно произвольное комбинирование программ на этих языках).
Минимальная структурная единица программы на языке ST	Задача До 288 задач (32 циклические задачи и 256 дополнительных циклических задач)
Задачи, программы которых могут создаваться на языке ST	Циклические задачи и дополнительные циклические задачи
Онлайн-редактирование.	Редактирование в окне представления программы на языке ST Примечание. Может быть выбран обычный режим (вместе с объектным кодом загружается исходный код) или быстрый режим (исходный код программы не загружается).
Переменные-массивы	Использование переменных-массивов возможно в программах на языке SFC, LD и ST.

РАЗДЕЛ 5

Характеристики языка структурированного текста

Данный раздел содержит технические данные и характеристики, которыми следует руководствоваться при создании программ на языке структурированного текста, а также примеры программ и сведения об ограничениях.

5-1	Характеристики языка структурированного текста	162
5-1-1	Обзор языка структурированного текста	162
5-2	Допустимые типы данных в программах на языке ST	163
5-2-1	Основные типы данных	163
5-2-2	Производные типы данных	164
5-3	Ввод программ на языке структурированного текста	164
5-3-1	Основные правила синтаксиса	164
5-3-2	Отображение окна ввода программы на языке ST в CX-Programmer	167
5-4	Элементы синтаксиса языка структурированного текста	169
5-4-1	Выражения.	169
5-4-2	Переменные.	171
5-4-3	Ввод фиксированных значений (констант)	171
5-4-4	Операторы	172
5-4-5	Стандартные функции	173
5-4-6	Расширенные функции OMRON.	179
5-5	Описание выражений языка структурированного текста	183
5-5-1	Присваивание	183
5-5-2	Управляющие выражения (конструкции).	183
5-6	Пример программы на языке структурированного текста	204
5-6-1	Использование программы на языке ST в функциональном блоке	204
5-7	Ограничения	204
5-7-1	Ограничения	204
5-7-2	Часто задаваемые вопросы	205

5-1 Характеристики языка структурированного текста

5-1-1 Обзор языка структурированного текста

Язык структурированного текста (язык ST) — это высокоуровневый текстовый язык программирования, близкий по синтаксису к языку Паскаль и имеющий в своем составе конструкции выбора и цикла.

Элементы синтаксиса языка структурированного текста

■ Элементы синтаксиса языка структурированного текста

Программа на языке структурированного текста составляется из выражений (statement). Выражения бывают двух типов: выражения присваивания и управляющие выражения (которые также называют конструкциями).

- Выражение присваивания: использует оператор присваивания, служит для присвоения некоторого значения (например, результата вычисления числового выражения) некоторой переменной.
- Управляющее выражение: к управляющим выражениям относятся конструкции выбора, ветвления, цикла и т. п.

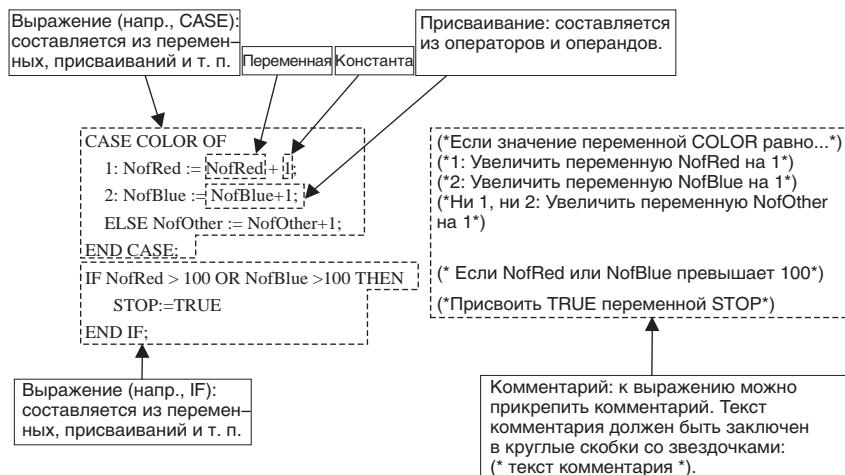
Подробную информацию о выражениях каждого типа см. в разделе 5-4 *Элементы синтаксиса языка структурированного текста*.

■ Содержание выражения

В общем случае выражение состоит из следующих элементов.

- Переменные (см. 5-4-2 *Переменные*)
- Константы (см. 5-4-3 *Ввод фиксированных значений (констант)*)
- Операторы (см. 5-4-4 *Операторы*)
- Функции (см. 5-4-5 *Стандартные функции* и 5-4-6 *Расширенные функции OMRON*)

■ Пример управляющего выражения



Примечание.

В программе на языке ST в качестве операндов вводятся не физические адреса памяти ввода/вывода, а имена переменных. Адреса, используемые для переменных, задаются пользователем отдельно. Подробное описание параметров и способов настройки переменных можно найти в руководстве *CX-Programmer — Руководство по работе (W446)*.

5-2 Допустимые типы данных в программах на языке ST

В следующей таблице перечислены типы данных, которые могут использоваться в программах на языке ST. Информацию о типах данных, которые могут использоваться в программах на языке ST в функциональных блоках, см. в части: *Часть 1: Функциональные блоки* в настоящем руководстве.

5-2-1 Основные типы данных

Тип данных	Содержание	Размер	Диапазон значений
BOOL	Логическое значение	1	0 (ЛОЖЬ), 1 (ИСТИНА)
INT	Целое	16	-32 768...+32 767
DINT	Двойное целое	32	-2 147 483 648...+2 147 483 647
LINT	Длинное (8 байтов) целое	64	-9 223 372 036 854 775 808...+9 223 372 036 854 775 807
UINT	Целое без знака	16	&0...65 535
UINT BCD	Двоично-десятичное целое без знака	---	(см. примеч. 1).
UDINT	Двойное целое без знака	32	&0...4 294 967 295
UDINT BCD	Двоично-десятичное двойное целое без знака	---	(см. примеч. 1).
ULINT	Длинное (8 байтов) целое без знака	64	&0...18 446 744 073 709 551 615
ULINT BCD	Двоично-десятичное длинное (8 байтов) целое без знака	---	(см. примеч. 1).
REAL	Вещественное число	32	$-3,402823 \times 10^{38}$... $-1,175494 \times 10^{-38}$, 0, $+1,175494 \times 10^{-38}$... $+3,402823 \times 10^{38}$
LREAL	Длинное вещественное число	64	$-1,79769313486232 \times 10^{308}$... $-2,22507385850720 \times 10^{-308}$, 0, $2,22507385850720 \times 10^{-308}$... $1,79769313486232 \times 10^{308}$
WORD	16-битовое значение	16	#0000...FFFF или &0...65 535
DWORD	32-битовое значение	32	#00000000...FFFFFFFF или &0...4 294 967 295
LWORD	64-битовое значение	64	#0000000000000000...FFFFFFFFFFFFFFFF или &0...18 446 744 073 709 551 615
STRING (см. примеч. 3)	Текстовая строка	Переменная	---
FUNCTION BLOCK	Экземпляр функционального блока	---	---
CHANNEL	Слово	---	(См. примеч. 1)
NUMBER	Константа или число	---	(См. примеч. 2)
TIMER	Таймер	Флаг заверш.: 1 Текущ. знач.: 16	Номер таймера: 0...4095 Флаг завершения таймера: 0, 1 Текущ. значение таймера: 0...9999 (BCD), 0...65535 (двоичный)
COUNTER	Счетчик	Флаг заверш.: 1 Текущ. знач.: 6	Номер счетчика: 0...4095 Флаг завершения счетчика: 0, 1 Текущ. значение счетчика: 0...9999 (BCD), 0...65535 (двоичный)

Примечание (1) В программах на языке ST эти типы данных воспринимаются как типы данных, указанные ниже.

- UNIT BCD воспринимается как WORD.

- UDINT BCD воспринимается как DWORD.
 - ULINT BCD воспринимается как LWORD.
 - CHANNEL воспринимается как WORD.
- (2) Использование этого типа данных в программе на языке ST невозможно. В случае его использования произойдет ошибка программы.
- (3) Способ ввода описан в *Разделе 5-3 Ввод программ на языке структурированного текста*.

5-2-2 Производные типы данных

Тип данных	Содержание
Массив	1-мерный массив; макс. 32 000 элементов
Структура	Пользовательский тип данных

5-3 Ввод программ на языке структурированного текста

5-3-1 Основные правила синтаксиса

Разделители в выражениях

- Каждое выражение (как присваивающее, так и управляющее) должно завершаться точкой с запятой (;). Перевод строки не является признаком завершения выражения, другими словами для завершения выражения недостаточно просто нажать клавишу «Ввод».
- Не следует использовать знак разделителя (;) внутри выражения. Это приведет к ошибке синтаксиса.

Комментарии

- Комментарии заключаются в круглые скобки со звездочками: (**комментарий**). Текст комментария может содержать любые символы, за исключением круглых скобок и звездочек. Вложение одного комментария в другой не допускается.

Форма записи

Пример

(**комментарий**)

(*Это комментарий*)

Примечание. Размещение одного комментария внутри другого невозможно:

(*(*вложение комментариев не поддерживается*)*)

Знаки пробела, перевода строки и табуляции

- Пробелы, знаки перевода строки (возврата каретки) и табуляции могут использоваться в любом количестве, в любом сочетании и в любом месте внутри выражений. Использование этих знаков для разделения ключевых слов и числовых выражений делает программу более наглядной и удобной для чтения.
- Не допускается использовать пробелы, знаки перевода строки и табуляции между указанными ниже лексемами (лексема — наименьшая группа символов, воспринимаемая транслятором как значащая), если это не предусмотрено синтаксисом языка ST. Лексемы: ключевые слова, имена переменных, специальные символы, константы (числовые значения).

Ключевые слова (в верхнем или нижнем регистре):	AND, CASE, DO, ELSE, FOR, IT, NOT, OF, OR, REPEAT, THEN, TO, UNTIL, WHILE, XOR, TRUE, FALSE, ELSIF, BY, EXIT, RETURN
Имена переменных:	любой набор текстовых символов, не распознаваемый как ключевое слово, воспринимается как имя переменной.
Специальные символы:	<=, >=, <>, :=, .., &, (*, *)
Константы (числовые значения):	<ul style="list-style-type: none"> Десятичное число: числовое значение без префикса. 16-ричное число: числовое значение с префиксом 16#. Двоичное число: числовое значение с префиксом 2#. 8-ричное число: числовое значение с префиксом 8#.

Если в пределах любой из указанных выше лексем будет использован знак пробела, перевода строки или табуляции, каждая из частей лексемы с обеих сторон от вставленного разделителя будет восприниматься как отдельная лексема. Поэтому знаки пробела, перевода строки и табуляции внутри единичных (неделимых) лексем использоваться не должны.

- Ключевые слова и имена переменных обязательно должны разделяться пробелом, знаком перевода строки, табуляции или другим знаком разделения лексем. Использование знаков разделения лексем в других комбинациях лексем является факультативным.

В приведенном ниже примере значок □ указывает места в тексте программы, где применение пробела, знака перевода строки, табуляции или другого знака разделения лексем обязательно.

```
IF □ A > 0 THEN □ X = 10 ;
ELSE □
    X := 0 ;
END_IF ;
```

Нечувствительность к регистру

Запрещенные символы в именах переменных

- Ключевые слова и имена переменных нечувствительны к регистру (могут набираться как в нижнем, так и в верхнем регистре).
- В именах переменных нельзя использовать перечисленные ниже символы, заключенные в квадратные скобки.
- [!], ["], [#], [\$], [%], [&], ['], [(, [)], [-], [=], [^], [~], [N], [I], [@], [], [{], [:], [+], [;], [*], [], [}], [.], [<], [.], [>], [/], [?]
- В качестве первого символа имени переменной нельзя использовать цифры 0...9.
- В именах переменных нельзя использовать два знака подчеркивания подряд.

- В именах переменных нельзя использовать пробелы.

При нарушении любого из указанных выше правил будет отображено сообщение об ошибке.

Приоритет операторов

- При составлении выражения следует учитывать приоритет операторов и, при необходимости, заключать в скобки те или иные операции в пределах выражения.

Пример: операнд AND обладает приоритетом над операндом OR, поэтому в логическом выражении X OR Y AND Z первой будет выполнена операция Y AND Z.

Тип данных STRING

- Поддерживаются текстовые строки длиной до 255 буквенно-цифровых символов.
Строчные и заглавные буквы в текстовых строках не различаются.

- Текстовые строки, применяемые в программе на языке ST, хранятся в памяти ПЛК следующим образом:

Значение текстовой строки '123456'

n	31 32
n+1	33 34
n+2	35 36
n+3	00 00

В конце текстовой строки содержится код «null» (00).

- Текстовые строки должны заключаться в одинарные кавычки.

Представление	Описание
'A'	Обозначает текстовую строку «A» (ASCII 41).
' '	Обозначает текстовую строку, содержащую один «пробел» (ASCII 20).
''	Обозначает пустую текстовую строку.

- Две шестнадцатеричные цифры со знаком доллара (\$) спереди воспринимаются как шестнадцатеричное значение.

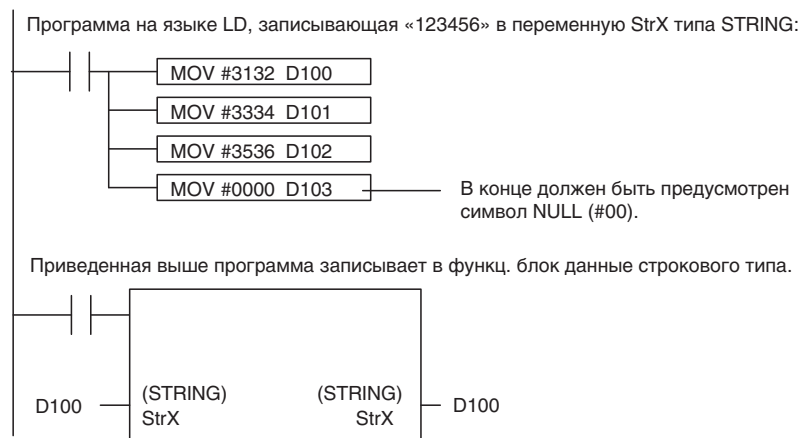
Представление	Описание
\$02	Шестнадцатеричное число 02 (код начала строки)
\$03	Шестнадцатеричное число 03 (код конца строки)

- Некоторые буквенные символы со знаком доллара (\$) спереди имеют особое значение, что отражено в таблице ниже.

Представление	Описание
\$\$	Знак доллара (ASCII 24)
'\$'	Одинарная кавычка (ASCII 27)
\$L или \$l	Перевод строки (ASCII 0A)
\$N или \$n	Возврат каретки + перевод строки (ASCII 0D 0A)
\$P или \$p	Новая страница (ASCII 0C)
\$R или \$r	Возврат каретки (ASCII 0D)
\$T или \$t	Символ табуляции (ASCII 09)

- Если в программе на языке LD вызывается функциональный блок с программой на языке ST и в переменную типа STRING функционального блока передается текстовая строка, в конец текстовой строки необходимо добавить символ NULL (#00).

Пример: передача строковых данных в переменную StrX типа STRING функционального блока:



Особенности использования типов данных TIMER и COUNTER

Ниже показано, как должны описываться переменные типа TIMER и COUNTER в программе на языке структурированного текста.

1) Описание переменных типа TIMER.

Флаг завершения таймера:

TIMER_type_variable_name.CF

Текущ. значение таймера:

TIMER_type_variable_name.PV

(Пример) Флаг завершения таймера:Timer1.CF

Текущ. значение таймера:Timer1.PV

2) Описание переменных типа COUNTER

Флаг завершения счетчика:

COUNTER_type_variable_name.CF

Текущ. значение счетчика:

COUNTER_type_variable_name.PV

(Пример) Флаг завершения счетчика:Counter1.CF

Текущ. значение счетчика:Counter1.PV

Флаги завершения могут быть только прочитаны, запись в них невозможна.

Текущие значения доступны для чтения и записи.

5-3-2 Отображение окна ввода программы на языке ST в CX-Programmer

Цвет отображения текста

При вводе программы непосредственно в окне ввода или после вставки программы в окно ввода из текстового редактора CX-Programmer автоматически отображает ключевые слова, комментарии и ошибки, используя указанные ниже цвета.

- Ключевые слова (зарезервированные слова): синий
- Комментарии: зеленый
- Ошибки: красный
- Прочие: черный

Изменение параметров шрифта

Размер и цвет используемого шрифта можно изменить, выбрав команду **Options (Настройки)** в меню **Tools (Сервис)**, открыв вкладку **Appearance (Вид)** и щелкнув кнопку **ST Font (Шрифт ST)**. В отобразившемся окне можно поменять шрифт, размер шрифта (по умолчанию установлено 8 точек) и его цвет.

5-4 Элементы синтаксиса языка структурированного текста

5-4-1 Выражения

Выражение		Назначение	Пример
Конец выражения		Завершение выражения	;
Комментарий		Любой текст между (* и *) воспринимается как комментарий.	(* <i>текст комментария</i> *)
Выражение присваивания	Присваивание	Переменной, расположенной слева от оператора присваивания, присваивается результат вычисления выражения, значение переменной или фиксированное значение из правой части.	A:=B;
Управляющие выражения (конструкции)	IF, THEN, ELSIF, ELSE, END_IF	Выполнение выражения, если равно «истине» его условие выполнения.	IF (<i>условие_1</i>) THEN (<i>выражение_1</i>); ELSIF (<i>условие_2</i>) THEN (<i>выражение_2</i>); ELSE (<i>выражение_3</i>); END_IF;
	CASE, ELSE, END_CASE	Выполнение того или иного выражения в зависимости от значения переменной.	CASE (<i>переменная</i>) OF 1: (<i>выражение_1</i>); 2: (<i>выражение_2</i>); 3: (<i>выражение_3</i>); ELSE (<i>выражение_4</i>); END_CASE;
	FOR, TO, BY, DO, END_FOR	Циклическое выполнение выражения в соответствии с начальным значением, конечным значением и шагом приращения.	FOR (<i>идентификатор</i>) := (<i>начальное_значение</i>) TO (<i>конечное_значение</i>) BY (<i>шаг_приращения</i>) DO (<i>выражение</i>); END_FOR;
	WHILE, DO, END_WHILE	Циклическое выполнение выражения, пока условие остается равным «истине».	WHILE (<i>условие</i>) DO (<i>выражение</i>); END_WHILE;
	REPEAT, UNTIL, END_REPEAT	Циклическое выполнение выражения, пока условие не становится равным «истине».	REPEAT (<i>выражение</i>); UNTIL (<i>условие</i>) END_REPEAT;
	EXIT	Прекращение циклического выполнения.	EXIT;
	RETURN	Программы на языке ST: Завершение выполняемой задачи на языке ST и выполнение следующей задачи. Программы на языке ST в SFC: Завершение выполняемой программы действия на языке SFC и выполнение следующей программы действия. Программы на языке ST в функц. блоках: Возврат из вызванной программы в точку вызова в вызвавшей программе.	RETURN;

Выражение		Назначение	Пример
Управляющие выражения (конструкции)	Вызов экземпляра функционального блока	Вызов экземпляра функционального блока.	В случае использования в функциональном блоке: Имя переменной с типом данных FUNCTION BLOCK (имя входной переменной вызываемого определения функционального блока := имя переменной вызывающего определения функционального блока или константа, ..., имя выходной переменной вызываемого определения функционального блока или константа => имя выходной переменной вызывающего определения функционального блока, ...);

5-4-2 Переменные

Информацию о параметрах и способах настройки переменных см. в руководстве *CX-Programmer — Руководство по работе (W469)*.

5-4-3 Ввод фиксированных значений (констант)

В программе на языке ST допускается ввод значений в десятичном, шестнадцатеричном, восьмеричном или двоичном формате. Ниже показаны примеры ввода значений в разных форматах.

Формат	Способ ввода	Пример (для десятичн. значения 12)
Десятичн.:	Только число (без префикса)	12
16-ричный:	Число с префиксом 16#	16#C
8-ричный:	Число с префиксом 8#	8#14
Двоичный:	Число с префиксом 2#	2#1100
Текст. строка:	Заключается в одинарные кавычки	'Всем привет!'

Примечание. Для представления отрицательных двоичных, восьмеричных и шестнадцатеричных чисел используется дополнение до двух. Переменная типа INT может иметь значения в диапазоне от -32 768 до 32 767 — в десятичном формате и от 0000 до FFFF — в шестнадцатеричном формате, поэтому для представления отрицательных целых чисел применяется дополнение до двух. Например, если в переменную типа INT будет записано десятичное значение -10, в шестнадцатеричном виде оно будет представлено как 16#FFF6.

5-4-4 Операторы

Операция	Символ	Поддерживаемые типы данных	Приоритет 1: минимальный 11: максимальный
Круглые и квадратные скобки	(выражение), массив[индекс]		1
Выполнение функции	идентификатор	Зависит от функции (см. Приложение С Описание функций)	2
Возведение в степень	**	REAL, LREAL	3
Дополнение	NOT	BOOL, WORD, DWORD, LWORD	4
Умножение	*	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	5
Деление	/	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	5
Сложение	+	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL, STRING	6
Вычитание	-	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	6
Сравнение	<, >, <=, >=	BOOL, INT, DINT, LINT, UINT, UDINT, ULINT, WORD, DWORD, LWORD, REAL, LREAL, STRING	7
Равенство	=	BOOL, INT, DINT, LINT, UINT, UDINT, ULINT, WORD, DWORD, LWORD, REAL, LREAL, STRING	8
Неравенство	<>	BOOL, INT, DINT, LINT, UINT, UDINT, ULINT, WORD, DWORD, LWORD, REAL, LREAL, STRING	8
Логическое И	&	BOOL, WORD, DWORD, LWORD	9
Логическое И	AND	BOOL, WORD, DWORD, LWORD	9
Логическое исключающее ИЛИ	XOR	BOOL, WORD, DWORD, LWORD	10
Логическое ИЛИ	OR	BOOL, WORD, DWORD, LWORD	11

Примечание. Выполнение операций зависит от типа значений, над которыми выполняются операции.

Поэтому выбор правильного типа данных для переменных играет важную роль. Например, если выполняется сложение двух значений типа INT, то переменная, в которую записывается результат сложения, также должна быть типа INT. Особого внимания требуют случаи возникновения переноса или заема при выполнении операций над целочисленными переменными. Например, операция $(A/B)*2$ над переменными целого типа $A=3$ и $B=2$ в качестве результата вернет 2: $A/B = 1$ (так как дробная часть в значении 1,5 будет отброшена), $(A/B)*2 = 2$.

5-4-5 Стандартные функции

Тип функции	Синтаксис
Числовые функции	Абсолютные значения, тригонометрические функции и т. п.
Арифметические функции	Экспонента (EXPT)
Функции преобразования типа данных	<i>Исх_тип_данных_TO_Новый_тип_данных (Имя_переменной)</i>
Функции преобразования числовых/строковых данных	<i>Исх_тип_данных_TO_STRING (Имя_переменной)</i> <i>STRING_TO_Новый_тип_данных (Имя_переменной)</i>
Функции сдвига данных	Побитовый сдвиг (SHL и SHR), побитовый циклический сдвиг (ROL и ROR) и т. п.
Функции управления данными	Ограничение сверху/снизу (LIMIT) и т. п.
Функции выбора данных	Выбор данных (SEL), макс. значение (MAX), мин. значение (MIN), мультиплексор (MUX) и т. п.

Числовые функции

Функция	Тип данных аргумента	Тип данных возвращаемого значения	Описание	Пример
ABS (<i>аргумент</i>)	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	Абсолютное значение [<i>аргумент</i>]	a = ABS (b) (*вычисление абсолютного значения переменной b и запись результата в переменную a*)
SQRT (<i>аргумент</i>)	REAL, LREAL	REAL, LREAL	Квадратный корень: $\sqrt{\text{аргумент}}$	a = SQRT (b) (*вычисление квадратного корня переменной b и запись результата в переменную a*)
LN (<i>аргумент</i>)	REAL, LREAL	REAL, LREAL	Натуральный логарифм: LOG_e аргумент	a = LN (b) (*вычисление натурального логарифма переменной b и запись результата в переменную a*)
LOG (<i>аргумент</i>)	REAL, LREAL	REAL, LREAL	Десятичный логарифм: LOG_{10} аргумент	a = LOG (b) (*вычисление десятичного логарифма переменной b и запись результата в переменную a*)
EXP (<i>аргумент</i>)	REAL, LREAL	REAL, LREAL	Экспонента: $e^{\text{аргумент}}$	a = EXP (b) (*вычисление экспоненты переменной b и запись результата в переменную a*)
SIN (<i>аргумент</i>)	REAL, LREAL	REAL, LREAL	Синус: SIN аргумент	a = SIN (b) (*вычисление синуса переменной b и запись результата в переменную a*)
COS (<i>аргумент</i>)	REAL, LREAL	REAL, LREAL	Косинус: COS аргумент	a = COS (b) (*вычисление косинуса переменной b и запись результата в переменную a*)
TAN (<i>аргумент</i>)	REAL, LREAL	REAL, LREAL	Тангенс: TAN аргумент	a = TAN (b) (*вычисление тангенса переменной b и запись результата в переменную a*)
ASIN (<i>аргумент</i>)	REAL, LREAL	REAL, LREAL	Арксинус: SIN^{-1} аргумент	a = ASIN (b) (*вычисление арксинуса переменной b и запись результата в переменную a*)

Функция	Тип данных аргумента		Тип данных возвращаемого значения	Описание	Пример
ACOS (аргумент)	REAL, LREAL		REAL, LREAL	Арккосинус: COS^{-1} аргумент	a = ACOS (b) (*вычисление арккосинуса переменной b и запись результата в переменную a*)
ATAN (аргумент)	REAL, LREAL		REAL, LREAL	Арктангенс: TAN^{-1} аргумент	a = ATAN (b) (*вычисление арктангенса переменной b и запись результата в переменную a*)
EXPT (основание, показатель)	Основание	REAL, LREAL	REAL, LREAL	Возведение в степень: основание ^{показатель}	a = EXPT (b, c) (*возведение переменной b в степень c и запись результата в переменную a*)
	Показатель	INT, DINT, LINT, UINT, UDINT, ULINT			
MOD (делимое, делитель)	Делимое	INT, UINT, UDINT, ULINT, DINT, LINT	INT, UINT, UDINT, ULINT, DINT, LINT	Остаток	a := MOD(b, c) (*вычисление остатка от деления переменной b на переменную c и запись результата в переменную a*)
	Делитель	INT, UINT, UDINT, ULINT, DINT, LINT			

Примечание. Значения, возвращаемые числовыми функциями, имеют такой же тип данных, что и у аргументов этих функций. Поэтому для переменных, подставляемых в качестве возвращаемых значений функций, должен указываться тот же тип данных, что и у аргументов.

Функции для работы с текстовыми строками

В следующей таблице перечислены функции, поддерживаемые модулями ЦПУ серии CS/CJ с версией модуля 4.0 и выше, а также модулями ЦПУ серии CJ2.

Функция	Тип данных аргумента		Тип данных возвращаемого значения	Описание	Пример
LEN(Строка)	Строка	STRING	INT	Определение длины текстовой строки.	a = LEN (b) (*определение количества символов в строке b и запись результата в переменную a*)
LEFT(<Входная_строка>, <Количество_символов>)	Входная_строка	STRING	STRING	Считывание символов текстовой строки, начиная слева.	a = LEFT (b,c) (*считывание c символов, расположенных в начале (слева) текстовой строки b, и запись результата в переменную a*)
	Количество_символов	INT, UINT			

Функция	Тип данных аргумента		Тип данных возвращаемого значения	Описание	Пример
RIGHT(<Входная_строка>, <Количество_символов>)	Входная_строка	STRING	STRING	Считывание символов текстовой строки, начиная справа.	a = RIGHT (b,c) (*считывание с символов, расположенных в конце (справа) текстовой строки b, и запись результата в переменную a*)
	Количество_символов	INT, UINT			
MID(<Входная_строка>, <Количество_символов>, <Позиция>)	Входная_строка	STRING	STRING	Считывание символов в пределах текстовой строки.	a = MID (b,c,d) (*считывание с символов в пределах текстовой строки b, начиная с позиции d, и запись результата в переменную a*)
	Количество_символов	INT, UINT			
	Позиция	INT, UINT			
CONCAT(<Входная_строка_1>, <Входная_строка_2>, ...)* До 32 входных строк.*	Входная_строка	STRING	STRING	Конкатенация текстовых строк.	a = CONCAT (b,c...) (*объединение текстовых строк b, c... и запись полученной строки в переменную a*)
INSERT(<Входная_строка>, <Вставляемая_строка>, <Позиция>)	Входная_строка	STRING	STRING	Вставка одной текстовой строки в другую.	a = INSERT (b,c,d) (*вставка текстовой строки c в позицию d текстовой строки b и запись полученной строки в переменную a*)
	Вставляемая_строка	STRING			
	Позиция	INT, UINT			
DELETE(<Входная_строка>, <Количество_символов>, <Позиция>)	Входная_строка	STRING	STRING	Удаление символов из текстовой строки.	a = DELETE (b,c,d) (*удаление с символов из текстовой строки b, начиная с позиции d, и запись полученной строки в переменную a*)
	Количество_символов	INT, UINT			
	Позиция	INT, UINT			
REPLACE(<Входная_строка>, <Замещающая_строка>, <Количество_символов>, <Позиция>)	Входная_строка	STRING	STRING	Замена символов в текстовой строке.	a = REPLACE (b,c,d,e) (*замена d символов в текстовой строке b текстовой строкой c, начиная с позиции e, и запись полученной строки в переменную a*)
	Замещающая_строка	STRING			
	Количество_символов	INT, UINT			
	Позиция	INT, UINT			
FIND(<Входная_строка>, <Искомая_строка>)	Входная_строка	STRING	INT	Поиск символов в текстовой строке.	a = FIND (b,c) (*поиск текстовой строки c в текстовой строке b и запись позиции первого вхождения в переменную a; в случае необнаружения строки c записывается 0*)
	Искомая_строка	STRING			

Функции преобразования типа данных

Данные одного типа могут быть преобразованы в эквивалентные данные другого типа. Для этих целей предусмотрена описанная ниже функция.

Синтаксис

Исх_тип_данных_TO_Новый_тип_данных (имя_переменной)

Пример: REAL_TO_INT (C)

В данном примере тип данных REAL переменной C будет преобразован в тип данных INT.

Комбинации типов данных

В следующей таблице приведены комбинации типов данных, допускающие конвертацию.

(Да = преобразование возможно, Нет = преобразование невозможно)

ИЗ	В ->													
	BOOL	INT	DINT	LINT	UINT	UDINT	ULINT	WORD	DWORD	LWORD	REAL	LREAL	BCD_WORD	BCD_DWORD
BOOL	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет
INT	Нет	Нет	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да
DINT	Нет	Да	Нет	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да
LINT	Нет	Да	Да	Нет	Да	Да	Да	Да	Да	Да	Да	Да	Нет	Нет
UINT	Нет	Да	Да	Да	Нет	Да	Да	Да	Да	Да	Да	Да	Да	Да
UDINT	Нет	Да	Да	Да	Да	Нет	Да	Да	Да	Да	Да	Да	Да	Да
ULINT	Нет	Да	Да	Да	Да	Да	Нет	Да	Да	Да	Да	Да	Нет	Нет
WORD	Нет	Да	Да	Да	Да	Да	Да	Нет	Да	Да	Нет	Нет	Нет	Нет
DWORD	Нет	Да	Да	Да	Да	Да	Да	Да	Нет	Да	Нет	Нет	Нет	Нет
LWORD	Нет	Да	Да	Да	Да	Да	Да	Да	Да	Нет	Нет	Нет	Нет	Нет
REAL	Нет	Да	Да	Да	Да	Да	Да	Нет	Нет	Нет	Нет	Да	Нет	Нет
LREAL	Нет	Да	Да	Да	Да	Да	Да	Нет	Нет	Нет	Да	Нет	Нет	Нет
WORD_BCD	Нет	Да	Да	Нет	Да	Да	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет
DWORD_BCD	Нет	Да	Да	Нет	Да	Да	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет

Функции преобразования числовых/строковых данных

Числовые данные могут быть преобразованы в строковые данные, а строковые данные могут быть преобразованы в числовые данные. Для этих целей предусмотрены описанные ниже функции.

Синтаксис

Исх_тип_данных_TO_STRING (имя_переменной)

Пример: INT_TO_STRING (C)

В данном примере целочисленная переменная C будет преобразована в строковую переменную (STRING).

STRING_TO_Новый_тип_данных (имя_переменной)

Пример: STRING_TO_INT (C)

В данном примере строковая переменная C будет преобразована в переменную целого типа.

Комбинации типов данных

В следующей таблице приведены комбинации типов данных, допускающие конвертацию.

(Да = преобразование возможно, Нет = преобразование невозможно)

ИЗ	В ->													
	BOOL	INT	DINT	LINT	UINT	UDINT	ULINT	WORD	DWORD	LWORD	REAL	LREAL	STRING	
BOOL	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	
INT	Нет	Нет	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да	
DINT	Нет	Да	Нет	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да	
LINT	Нет	Да	Да	Нет	Да	Да	Да	Да	Да	Да	Да	Да	Нет	
UINT	Нет	Да	Да	Да	Нет	Да	Да	Да	Да	Да	Да	Да	Да	
UDINT	Нет	Да	Да	Да	Да	Нет	Да	Да	Да	Да	Да	Да	Да	
ULINT	Нет	Да	Да	Да	Да	Да	Нет	Да	Да	Да	Да	Да	Нет	
WORD	Нет	Да	Да	Да	Да	Да	Да	Нет	Да	Да	Нет	Нет	Да	
DWORD	Нет	Да	Да	Да	Да	Да	Да	Да	Нет	Да	Нет	Нет	Да	
LWORD	Нет	Да	Да	Да	Да	Да	Да	Да	Да	Нет	Нет	Нет	Нет	
REAL	Нет	Да	Да	Да	Да	Да	Да	Да	Да	Да	Нет	Нет	Нет	
LREAL	Нет	Да	Да	Да	Да	Да	Да	Да	Да	Да	Нет	Нет	Нет	
STRING	Нет	Да	Да	Нет	Да	Да	Нет	Да	Да	Нет	Нет	Нет	Нет	

**Функции сдвига
данных**

Функция	Тип данных 1-го аргумента	Тип данных 2-го аргумента	Тип данных возвращаемого значения	Описание	Пример
SHL(<Сдвигаемые_данные>, <Количество_битов>)	BOOL, WORD, DWORD, LWORD	INT, UINT, UDINT, ULINT, DINT, LINT	BOOL, WORD, DWORD, LWORD	Сдвиг строки битов на n битов влево. В разряды в правой части строки записываются нули.	$a := \text{SHL}(b,c)$ (* Сдвиг строки битов b на c битов влево и запись полученной строки в a^*)
SHR(<Сдвигаемые_данные>, <Количество_битов>)	BOOL, WORD, DWORD, LWORD	INT, UINT, UDINT, ULINT, DINT, LINT	BOOL, WORD, DWORD, LWORD	Сдвиг строки битов на n битов вправо. В разряды в левой части строки записываются нули.	$a := \text{SHR}(b,c)$ (* Сдвиг строки битов b на c битов вправо и запись полученной строки в a^*)
ROL(<Циклически_сдвигаемые_данные>, <Количество_битов>)	BOOL, WORD, DWORD, LWORD	INT, UINT, UDINT, ULINT, DINT, LINT	BOOL, WORD, DWORD, LWORD	Циклический сдвиг строки битов на n битов влево.	$a := \text{ROL}(b,c)$ (* Циклический сдвиг строки битов b на c битов влево и запись полученной строки в a^*)
ROR(<Циклически_сдвигаемые_данные>, <Количество_битов>)	BOOL, WORD, DWORD, LWORD	INT, UINT, UDINT, ULINT, DINT, LINT	BOOL, WORD, DWORD, LWORD	Циклический сдвиг строки битов на n битов вправо.	$a := \text{ROR}(b, c)$ (* Циклический сдвиг строки битов b на c битов вправо и запись полученной строки в a^*)

**Функции
управления
данными**

Функция	Тип данных 1-го аргумента	Тип данных 2-го аргумента	Тип данных 3-го аргумента	Тип данных возвращаемого значения	Описание	Пример
LIMIT (<Нижнее_предельное_значение>, <Входное_значение>, <Верхнее_предельное_значение>)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Определение принадлежности входного значения диапазону, заданному верхним и нижним предельными значениями, и возврат соответствующего значения.	$a := \text{LIMIT}(b,c,d)$ (*Если $c < b$, в a записывается b . Если $b \leq c < d$, в a записывается c . Если $d < c$, в a записывается d .*)

Функции выбора данных

Функция	Тип данных 1-го аргумента	Тип данных 2-го аргумента	Тип данных 3-го аргумента	Тип данных возвращаемого значения	Описание	Пример
SEL(<Условие_выбора>, <Заданное_значение_1>, <Заданное_значение_2>)	BOOL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Выбор одного из двух значений на основании состояния условия выбора.	a := SEL(b,c,d) (*Если b = ИСТИНА, в a записывается c. Если b = ЛОЖЬ, в a записывается d.*)
MUX(<Условие_выбора>, <Заданное_значение_1>, <Заданное_значение_2>, ...)	INT, UINT, UDINT, ULINT, DINT, LINT	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Выбор одного из нескольких (максимум 30) значений в соответствии со значением условия выбора.	a := MUX(b,c,d,...) (*В a записывается значение с порядковым номером (b+1).*)
MAX(<Заданное_значение_1>, <Заданное_значение_2>, <Заданное_значение_3>, ...) *2	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Выбор максимального значения среди нескольких (максимум 31) заданных значений.	a := MAX(b,c,d, ...) (* В a записывается наибольшее из значений c, d,*)
MIN(<Заданное_значение_1>, <Заданное_значение_2>, <Заданное_значение_3>, ...) *2	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Выбор минимального значения среди нескольких (максимум 31) заданных значений.	a := MIN(b,c,d, ...) (* В a записывается наименьшее из значений c, d,*)

- Примечание.**
- (1) Для функции MUX может быть указан максимум 31 аргумент (т. е. максимум 30 значений для выбора).
 - (2) Для функций MAX и MIN может быть указано от 1 до 31 аргумента (т. е. максимум 31 значение для поиска максимума или минимума).

5-4-6 Расширенные функции OMRON

Тип функции	Описание
Функции для работы с картой памяти	Функции для записи данных на карты памяти.
Функции связи	Функции для передачи и приема текстовых строк.
Функции преобразования угловых величин	Функции для преобразования угловых величин из градусов в радианы и наоборот.
Функции таймеров/счетчиков	Функции для реализации таймеров и счетчиков разного типа.

Функции для работы с картой памяти В следующей таблице перечислены функции, поддерживаемые модулями ЦПУ серии CS/CJ с версией модуля 4.0 и выше, а также модулями ЦПУ серии CJ2.

Функция	Тип данных аргумента		Тип данных возвращаемого значения	Описание	Пример
WRITE_TEXT(<Записываемая_строка>, <Имя_каталога_и_имя_файла>,<Разделитель>, <Параметр>)	Записываемая_строка	STRING	---	Запись текстовой строки на карту памяти.	WRITE_TEXT(a,b,c,d) (*Запись текстовой строки a в файл; имя файла и имя каталога указываются в переменной b; если переменная d = 0, текстовая строка добавляется в файл вместе с разделителем, который указан в переменной c; если переменная d = 1, создается новый файл.*)
	Имя_каталога_и_имя_файла	STRING			
	Разделитель	STRING			
	Параметр	INT, UINT, WORD			

Функции связи В следующей таблице перечислены функции, поддерживаемые модулями ЦПУ серии CS/CJ с версией модуля 4.0 и выше, а также модулями ЦПУ серии CJ2.

Функция	Тип данных аргумента		Тип данных возвращаемого значения	Описание	Пример
TXD_CPU(<Передаваемая_строка>)	Передаваемая_строка	STRING	---	Передача текстовой строки через порт RS-232C модуля ЦПУ.	TXD_CPU(a) (*Передача текстовой строки a через порт RS-232C модуля ЦПУ.*)
TXD_SCB(<Передаваемая_строка>, <Последовательный_порт>)	Передаваемая_строка	STRING	---	Передача текстовой строки через последовательный порт платы последовательного интерфейса.	TXD_SCB(a,b) (*Передача текстовой строки a через последовательный порт платы последовательного интерфейса, указанный переменной b.*)
	Последовательный_порт	INT, UINT, WORD			

Функция	Тип данных аргумента		Тип данных возвращаемого значения	Описание	Пример
TXD_SCU(<Передаваемая_строка>, <Номер_модуля_SCU>, <Последовательный_порт>,<Внутренний_логический_порт>)	Передаваемая_строка	STRING	---	Передача текстовой строки через последовательный порт модуля последовательного интерфейса.	TXD_SCU(a,b,c,d) (*Передача текстовой строки a через последовательный порт, указанный переменной c, модуля последовательного интерфейса, указанного переменной b, с использованием внутреннего логического порта, указанного переменной d. Переменная d указывает номер внутреннего логического порта.*)
	Номер_модуля_SCU	INT, UINT, WORD	---		
	Последовательный_порт	INT, UINT, WORD	---		
	Внутренний_логический_порт	INT, UINT, WORD	---		
RXD_CPU(<Получатель>, <Количество_символов>)	Получатель	STRING	---	Прием текстовой строки через порт RS-232C модуля ЦПУ.	RXD_CPU(a,b) (*Прием указанного переменной b количества символов через порт RS-232C модуля ЦПУ и запись полученных данных в переменную a.*)
	Количество_символов	INT, UINT, WORD			
RXD_SCB(<Получатель>, <Количество_символов>,<Последовательный_порт>)	Получатель	STRING	---	Прием текстовой строки через последовательный порт платы последовательного интерфейса.	RXD_SCB(a,b,c) (*Прием указанного переменной b количества символов через последовательный порт платы последовательного интерфейса, указанный переменной c, и запись принятых данных в переменную a.*)
	Количество_символов	INT, UINT, WORD			
	Последовательный_порт	INT, UINT, WORD			
RXD_SCU(<Получатель>, <Количество_символов>, <Номер_модуля_SCU>, <Последовательный_порт>, <Внутренний_логический_порт>)	Получатель	STRING	---	Прием текстовой строки через последовательный порт модуля последовательного интерфейса.	RXD_SCU(a,b,c,d,e) (*Прием указанного переменной b количества символов через последовательный порт, указанный переменной d, модуля последовательного интерфейса, указанного переменной c, с использованием внутреннего логического порта, указанного переменной e, и запись принятых данных в переменную a. Переменная e указывает номер внутреннего логического порта.*)
	Количество_символов	INT, UINT, WORD			
	Номер_модуля_SCU	INT, UINT, WORD			
	Последовательный_порт	INT, UINT, WORD			
	Внутренний_логический_порт	INT, UINT, WORD			

Команды для преобразования угловых значений

В следующей таблице перечислены функции, поддерживаемые модулями ЦПУ серии CS/CJ с версией модуля 4.0 и выше, а также модулями ЦПУ серии CJ2.

Функция	Тип данных аргумента	Тип данных возвращаемого значения	Описание	Пример
DEG_TO_RAD (аргумент)	REAL, LREAL	REAL, LREAL	Перевод значения угла, выраженного в градусах, в радианы.	a:=DEG_TO_RAD(b) (*Перевод содержащегося в переменной b значения угла из градусов в радианы и запись результата в переменную a.*)
RAD_TO_DEG (аргумент)	REAL, LREAL	REAL, LREAL	Перевод значения угла, выраженного в радианах, в градусы.	a:=RAD_TO_DEG(b) (*Перевод содержащегося в переменной b значения угла из радианов в градусы и запись результата в переменную a.*)

Функции для работы с таймерами/счетчиками

В следующей таблице перечислены функции, поддерживаемые модулями ЦПУ серии CJ2.

Функция	Тип данных аргумента	Тип данных возвращаемого значения	Описание	Пример	
TIMX (<Условие_выполнения>, <Адрес_таймера>, <Уставка_таймера>)	Условие_выполнения	BOOL	Нет	Название: 100 мс ТАЙМЕР Действие: Реализует вычитающий таймер с шагом отсчета 100 мс.	TIMX(a,b,c) (*Если соблюдается условие выполнения a, запускается таймер TIMX с уставкой c и адресом таймера b.*)
	Адрес_таймера	TIMER			
	Уставка_таймера	UINT			
TIMHX (<Условие_выполнения>, <Адрес_таймера>, <Уставка_таймера>)	Условие_выполнения	BOOL	Нет	Название: 10 мс ТАЙМЕР Действие: Реализует вычитающий таймер с шагом отсчета 10 мс.	TIMHX(a,b,c) (*Если соблюдается условие выполнения a, запускается таймер TIMHX с уставкой c и адресом таймера b.*)
	Адрес_таймера	TIMER			
	Уставка_таймера	UINT			
TMNHX (<Условие_выполнения>, <Адрес_таймера>, <Уставка_таймера>)	Условие_выполнения	BOOL	Нет	Название: 1 мс ТАЙМЕР Действие: Реализует вычитающий таймер с шагом отсчета 1 мс.	TMNHX(a,b,c) (*Если соблюдается условие выполнения a, запускается таймер TMNHX с уставкой c и адресом таймера b.*)
	Адрес_таймера	TIMER			
	Уставка_таймера	UINT			
TIMUX (<Условие_выполнения>, <Адрес_таймера>, <Уставка_таймера>)	Условие_выполнения	BOOL	Нет	Название: 0,1 мс ТАЙМЕР Действие: Реализует вычитающий таймер с шагом отсчета 0,1 мс.	TIMUX(a,b,c) (*Если соблюдается условие выполнения a, запускается таймер TIMUX с уставкой c и адресом таймера b.*)
	Адрес_таймера	TIMER			
	Уставка_таймера	UINT			
TMUHX (<Условие_выполнения>, <Адрес_таймера>, <Уставка_таймера>)	Условие_выполнения	BOOL	Нет	Название: 0,01 мс ТАЙМЕР Действие: Реализует вычитающий таймер с шагом отсчета 0,01 мс.	TMUHX(a,b,c) (*Если соблюдается условие выполнения a, запускается таймер TMUHX с уставкой c и адресом таймера b.*)
	Адрес_таймера	TIMER			
	Уставка_таймера	UINT			

Функция	Тип данных аргумента		Тип данных возвращаемого значения	Описание	Пример
TTIMX (<Условие_выполнения>, <Вход_сброса>, <Адрес_таймера>, <Уставка_таймера>)	Условие_выполнения	BOOL	Нет	Название: НАКАПЛИВАЮЩИЙ ТАЙМЕР Действие: Реализует суммирующий таймер с шагом отсчета 0,1 с.	TTIMX(a,b,c,d) (*Если соблюдается условие выполнения a, запускается таймер TTIMX с уставкой d и адресом таймера c. Когда вход сброса b включен, текущее значение и флаг завершения таймера сброшены.*)
	Вход_сброса	BOOL			
	Адрес_таймера	TIMER			
	Уставка_таймера	UINT			
CNTX(<Вход_счета>, <Вход_сброса>, <Адрес_счетчика>, <Уставка_счетчика>)	Вход_счета	BOOL	Нет	Название: СЧЕТЧИК Действие: Реализует счетчик обратного счета.	CNTX(a,b,c,d) (*При каждом включении (переходе из «0» в «1») входа счета a срабатывает счетчик CNTX с уставкой d и адресом счетчика c. Когда вход сброса b включен, текущее значение и флаг завершения счетчика сброшены.*)
	Вход_сброса	BOOL			
	Адрес_счетчика	COUNTER			
	Уставка_счетчика	UINT			
CNTRX (<Прямой_счет>, <Обратный_счет>, <Вход_сброса>, <Адрес_счетчика>, <Уставка_счетчика>)	Прямой_счет	BOOL	Нет	Название: РЕВЕРСИВНЫЙ СЧЕТЧИК Действие: Реализует счетчик прямого / обратного счета.	CNTRX(a,b,c,d,e) (*Запускается счетчик CNTRX с уставкой e и адресом счетчика d. Текущее значение счетчика увеличивается при включении входа прямого счета a и уменьшается при включении входа обратного счета b. Когда вход сброса c включен, текущее значение и флаг завершения счетчика сброшены.*)
	Обратный_счет	BOOL			
	Вход_сброса	BOOL			
	Адрес_счетчика	COUNTER			
	Уставка_счетчика	UINT			
TRSET (<Условие_выполнения>, <Адрес_таймера>)	Условие_выполнения	BOOL	Нет	Название: СБРОС ТАЙМЕРА Действие: Сброс указанного таймера.	TRSET(a,b) (*Если соблюдается условие выполнения a, таймер с адресом b сбрасывается.*)
	Адрес_таймера	TIMER			

5-5 Описание выражений языка структурированного текста

5-5-1 Присваивание

■ Назначение

В левую часть выражения (переменную) подставляется правая часть выражения (логическое/числовое выражение, переменная или константа).

■ Зарезервированные слова

:=

Комбинация из двоеточия (:) и знака «равно» (=).

■ Синтаксис выражения

Переменная := *Выражение*, *переменная* или *константа*;

■ Использование

Выражение присваивания служит для ввода требуемого значения в переменную. Это базовое выражение, которое используется как до выполнения управляющих конструкций, так и внутри них. С его помощью можно задавать начальные значения, сохранять результаты вычислений, увеличивать или уменьшать значения переменных.

■ Описание

Подстановка (запись) значения *выражения*, *переменной* или *константы* в *переменную*.

Примеры

Пример 1: в переменную A записывается результат вычисления выражения X+1.

A:=X+1;

Пример 2: в переменную A записывается значение переменной B.

A:=B;

Пример 3: в переменную A записывается значение константы 10.

A:=10;

■ Предостережения

Присваиваемое значение выражения, переменной или константы должно иметь такой же тип данных, что и переменная, которой присваивается значение. В противном случае произойдет ошибка синтаксиса.

5-5-2 Управляющие выражения (конструкции)

IF (с одним условием)

■ Назначение

Данная конструкция служит для выполнения выражения при выполнении указанного условия. Если условие не соблюдается, выполняется другое выражение.

■ Зарезервированные слова

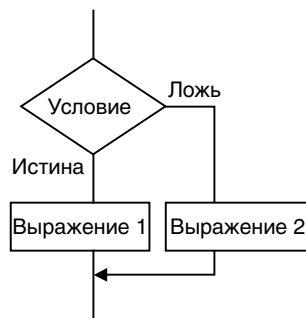
IF, THEN, (ELSE), END_IF

Примечание. Слово ELSE может отсутствовать.

■ Синтаксис выражения

```
IF <условие> THEN
  <выражение_1>;
ELSE
  <выражение_2>;
END_IF;
```

■ Блок схема выполнения



■ Использование

Конструкция IF выполняет одну из двух возможных операций в зависимости от состояния одиночного условия (условного выражения).

■ Описание

Если *условие* = истина, выполняется *выражение_1*

Если *условие* = ложь, выполняется *выражение_2*

■ Предостережения

- Слово IF должно использоваться со словом END_IF.
- *условие* должно содержать логическое выражение, возвращающее один из двух результатов: «истина» или «ложь».

Пример: IF(A>10)

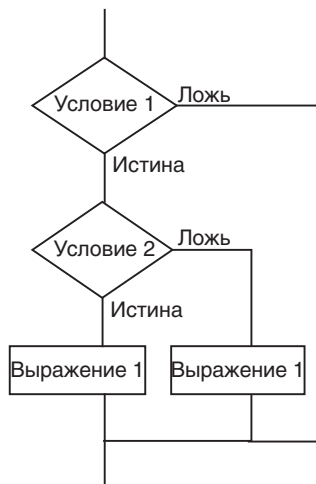
Вместо выражения в *условии* также может быть указана просто логическая переменная. Состояние «1» (ВКЛ) воспринимается как «истина», состояние «0» (ВЫКЛ) воспринимается как «ложь».

- В *выражении_1* и *выражении_2* могут использоваться выражения присваивания, а также конструкции IF, CASE, FOR, WHILE или REPEAT.

Пример:

```
IF <условие_1> THEN
  IF <условие_2> THEN
    <выражение_1>;
  ELSE
    <выражение_2>;
  END_IF;
END_IF;
```

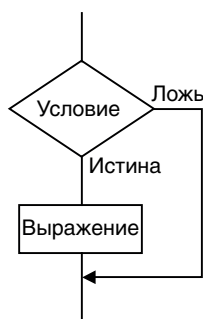
Ниже показана блок-схема выполнения для данного примера:



Слово ELSE относится к слову THEN, которое находится непосредственно перед ним (см. пример программы выше).

- Каждое из выражений (*выражение_1* и *выражение_2*) может состоять из нескольких отдельных выражений. Выражения в пределах каждой группы выражений обязательно должны отделяться друг от друга точкой с запятой (;).
- Выражение ELSE может отсутствовать. Если *условие* не соблюдается (= ложь), а выражение ELSE отсутствует, никакая операция не выполняется.

■ Блок схема выполнения



■ Примеры

Пример 1. Если выражение $A > 0$ истинно, в переменную X записывается числовое значение 10. Если выражение $A > 0$ ложно, в переменную X записывается числовое значение 0.

```

IF A>0 THEN
  X:=10;
ELSE
  X:=0;
END_IF;
  
```

Пример 2. Если оба выражения, $A > 0$ и $B > 1$, верны (= истина), в переменную X записывается числовое значение 10, а в переменную Y записывается числовое значение 20. Если оба выражения, $A > 0$ и $B > 1$, не верны (= ложь), в обе переменные, X и Y, записывается числовое значение 0.

```
IF A>0 AND B>1 THEN
  X:=10; Y:=20;
ELSE
  X:=0; Y:=0;
END_IF;
```

Пример 3. Если переменная логического типа A=1 (ВКЛ), в переменную X записывается числовое значение 10. Если переменная A=0 (ВЫКЛ), в переменную X записывается числовое значение 0.

```
IF A THEN X:=10;
ELSE X:=0;
END_IF;
```

IF (с несколькими условиями)

■ **Назначение**

Данная конструкция служит для выполнения выражения при выполнении указанного условия. В зависимости от того, какое условие соблюдается, выполняется то или иное выражение. Если ни одно из условий не соблюдается, выполняется выражение, предусмотренное для этого случая.

■ **Зарезервированные слова**

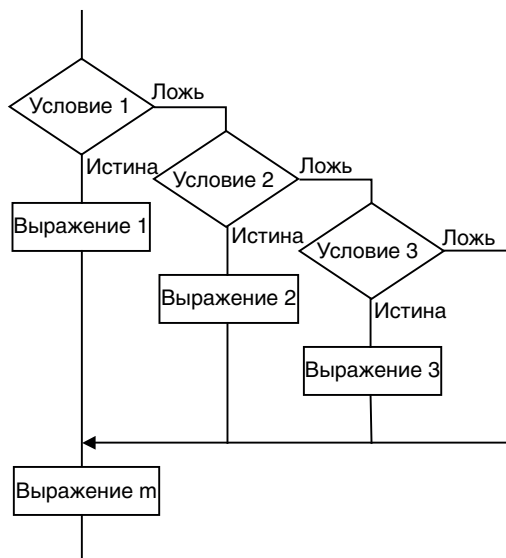
IF, THEN, ELSIF, (ELSE), END_IF

Примечание. Слово ELSE может отсутствовать.

Синтаксис выражения

```
IF <условие_1> THEN <выражение_1>;
  ELSIF <условие_2> THEN <выражение_2>;
  ELSIF <условие_3> THEN <выражение_3>;
  . . .
  ELSIF <условие_n> THEN <выражение_n>;
ELSE <выражение_m>;
END_IF;
```

Блок схема выполнения



■ Использование

Конструкция IF выполняет одну из возможных операций в зависимости от того, какое из множества заданных условий (*условных выражений*) выполняется в данный момент.

■ Описание

Если *Условие 1* = истина, то выполняется *выражение 1*

Если *Условие 1* = ложь, то

Если *Условие 2* = истина, то выполняется *выражение 2*

Если *Условие 2* = ложь, то

Если *Условие 3* = истина, то выполняется *выражение 3*

и так далее...

Если *Условие n* = истина, то выполняется *выражение n*

Если ни одно из условий не равно «истине», то выполняется *выражение m*.

■ Предостережения

- Слово IF должно использоваться со словом END_IF.
- *Условие_□* может содержать логическое выражение, возвращающее в качестве результата «истину» или «ложь» (напр., IF(A>10)). В качестве *условия* вместо выражения также может быть указана переменная логического типа (BOOL). Состояние «1» (ВКЛ) воспринимается как «истина», а состояние «0» (ВЫКЛ) воспринимается как «ложь».
- *Выражение_□* может содержать выражения присваивания, IF, CASE, FOR, WHILE или REPEAT.
- *Выражение_□* может состоять из нескольких отдельных выражений. Выражения в пределах каждой группы выражений обязательно должны отделяться друг от друга точкой с запятой (;).
- Выражение ELSE может отсутствовать. Если ни одно из *условий* не соблюдается (все равны «ложь»), а выражение ELSE отсутствует, никакая операция не выполняется.

■ Примеры

Пример 1. Если выражение A>0 истинно, в переменную X записывается числовое значение 10.

Если выражение A>0 ложно, но переменная B=1, в переменную X записывается числовое значение 1.

Если выражение A>0 ложно, но переменная B=2, в переменную X записывается числовое значение 2.

Если ни одно из этих условий не выполняется, в переменную X записывается числовое значение 0.

```
IF A>0 THEN X:=10;
      ELSIF B=1 THEN X:=1;
      ELSIF B=2 THEN X:=2;
ELSE X:=0;
END_IF;
```

CASE

■ **Назначение**

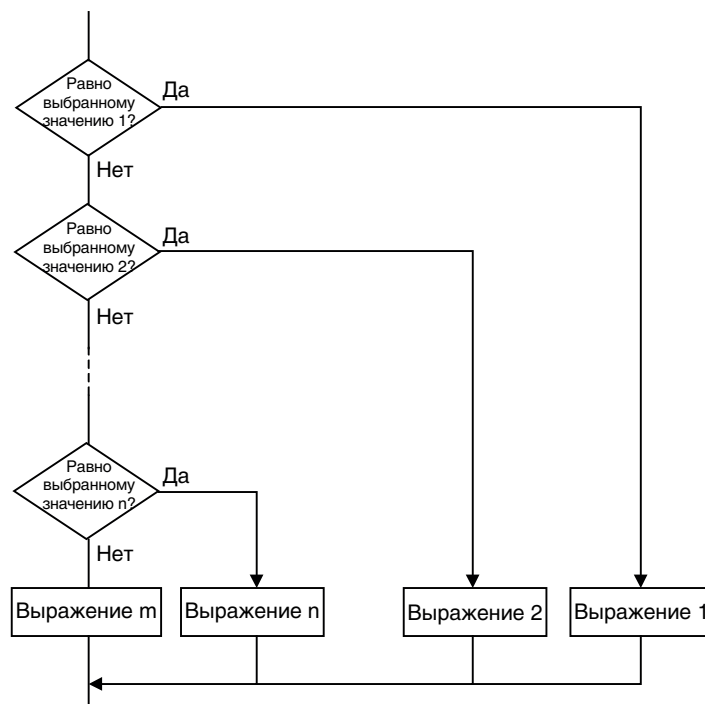
Данная конструкция выполняет выражение, соответствующее целому числу, совпадающему с результатом заданного условного выражения (возвращающего целое значение). Если результат условного выражения не равен ни одному из указанных целых чисел, никакое выражение не выполняется, либо выполняется выражение, указанное для этого случая.

■ **Зарезервированное слово**

CASE

■ **Синтаксис выражения**

```
CASE <условное_выражение_целого_типа> OF
  <целое_значение_1> : <выражение_1> ;
  <целое_значение_2> : <выражение_2> ;
  . . .
  <целое_значение_n> : <выражение_n> ;
ELSE <выражение_m> ;
END_CASE ;
```

■ **Блок схема выполнения**■ **Использование**

Конструкция CASE выполняет ту или иную операцию в зависимости от значения целочисленного условного выражения (также называемого переключателем).

■ **Описание**

Если *условное_выражение_целого_типа* совпадает с *целым_значением_n*, выполняется *выражение_n*.
Если *условное_выражение_целого_типа* не совпадает с *целым_значением_n*, выполняется *выражение_m*.

■ Предостережения

- Слово CASE должно использоваться со словом END_CASE.
- *условное_выражение_целого_типа* должно возвращать целочисленный результат (INT, DINT, LINT, UINT, UDINT или ULINT).
- *Выражение_□* может содержать выражения присваивания, IF, CASE, FOR, WHILE или REPEAT.
- *Выражение_□* может состоять из нескольких отдельных выражений. Выражения в пределах каждой группы выражений обязательно должны отделяться друг от друга точкой с запятой (;).
- В качестве *условного_выражения_целого_типа* может быть указана целочисленная переменная (INT, DINT, LINT, UINT, UDINT или ULINT) или выражение, возвращающее целочисленный результат.
- Вместо одного *целого_значения_n* можно указать несколько возможных целых значений, перечислив их через запятую. Можно также указать диапазон значений, используя в качестве разделителя две точки (..).

■ Примеры

Пример 1. Если переменная A = 1, в переменную X записывается числовое значение 1. Если переменная A = 2, в переменную X записывается числовое значение 2. Если переменная A = 3, в переменную X записывается числовое значение 3. Если переменная A не равна ни одному из этих значений, в переменную Y записывается значение 0.

```
CASE A OF
    1:X:=1;
    2:X:=2;
    3:X:=3;
ELSE Y:=0;
END_CASE;
```

Пример 2. Если переменная A = 1, в переменную X записывается числовое значение 1. Если переменная A = 2 или 5, в переменную X записывается числовое значение 2. Если значение переменной A лежит в пределах от 6 до 10, в переменную X записывается числовое значение 3. Если переменная A = 11, 12 или находится в пределах от 15 до 20, в переменную X записывается числовое значение 4. Если переменная A не равна ни одному из указанных выше значений и не попадает ни в один из указанных выше диапазонов, в переменную Y записывается значение 0.

```
CASE A OF
    1:X:=1;
    2,5:X:=2;
    6..10:X:=3;
    11,12,15..20:X:=4;
ELSE Y:=0;
END_CASE;
```

FOR

■ Назначение

Данная конструкция служит для циклического выполнения указанного выражения до достижения указанной переменной (далее называемой «переменной цикла») указанного значения.

■ Зарезервированные слова

FOR, TO, (BY), DO, END_FOR

Примечание. Слово BY может отсутствовать.

■ Синтаксис выражения

```
FOR <переменная_цикла> := <выражение_начального_значения> TO
<выражение_конечного_значения> BY
<выражение_величины_приращения>
DO
<выражение>;
END_FOR;
```

■ Блок схема выполнения



■ Использование

Цикл FOR используется для циклического выполнения операций, когда заранее известно количество циклов выполнения. В частности, конструкцию FOR удобно использовать для обработки множества элементов массива, указывая переменную цикла в качестве индекса массива.

■ Описание

Если *переменная_цикла* равна *начальному_значению*, выполняется *выражение*. После выполнения к *переменной_цикла* добавляется полученное значение *выражения_приращения*, и если *переменная_цикла* \leq *выражения_конечного_значения* (см. примеч. 1), то снова выполняется *выражение*. После выполнения к *переменной_цикла* вновь добавляется полученное значение *выражения_приращения*, и если *переменная_цикла* $<$ *выражения_конечного_значения* (см. примеч. 1), то снова выполняется *выражение*. Этот процесс циклически повторяется.

Наконец, когда *переменная_цикла* $>$ *выражения_конечного_значения* (см. примеч. 2), цикл завершается.

Примечание (1) При отрицательном значении *выражения_приращения* выражение выполняется, если *переменная_цикла* \geq *выражения_конечного_значения*.

(2) При отрицательном значении *выражения_приращения* цикл завершается, если *переменная_цикла* < *выражения_конечного_значения*.

■ Предостережения

- В качестве *выражения_приращения* может быть указано отрицательное значение.
- Слово FOR должно использоваться со словом END_FOR.
- *начальное_значение*, *выражение_конечного_значения* и *выражение_приращения* должны быть целого типа (INT, DINT, LINT, UINT, UDINT или ULINT).
- После выполнения последнего цикла к переменной цикла (которая к этому времени равна конечному значению) добавляется 1, и на этом цикл завершается.

Пример: в приведенном ниже примере программы на языке ST переменная *a* принимает значение «истина» (TRUE).

```
FOR i:=0 TO 100 DO
  array[i]:=0;
END_FOR;
```

```
IF i=101 THEN
  a:=TRUE;
ELSE
  a:=FALSE;
END_IF;
```

- Не следует изменять значение переменной цикла непосредственно в теле цикла FOR. Это может привести к непредсказуемому результату.

Пример:

```
FOR i:=0 TO 100 BY 1 DO
  array[i]:=0;
  i:=i+5;
END_FOR;
```

- *Выражение* может содержать выражения присваивания, IF, CASE, FOR, WHILE или REPEAT.
- *Выражение* может состоять из нескольких отдельных выражений. Эти выражения обязательно должны отделяться друг от друга точкой с запятой (;).
- BY *выражение_приращения* может отсутствовать. В этом случае величина приращения принимается равной 1.
- В качестве *выражения_начального_значения*, *выражения_конечного_значения* и *выражения_приращения* могут указываться целочисленные переменные (INT, DINT, LINT, UINT, UDINT или ULINT) или выражения, возвращающие целочисленный результат.

Пример 1. Переменная цикла *n* увеличивается от 0 до 50 с шагом 5, в переменную-массив SP[*n*] циклически записывается значение 100.

```
FOR n:=0 TO 50 BY 5 DO
  SP[n]:=100;
END_FOR;
```

Пример 2. Вычисляется сумма всех элементов массива DATA[*n*] (от DATA[1] до DATA[50]), результат записывается в переменную SUM.

```
FOR n:=0      TO 50  BY 1 DO
  SUM:=SUM+DATA[n];
END_FOR;
```

Пример 3. Определяется наибольшее и наименьшее значения элементов массива DATA[n] (от DATA[1] до DATA[50]). Максимальное значение записывается в переменную MAX, а минимальное значение записывается в переменную MIN. Значения элементов массива DATA[n] находятся в диапазоне от 0 до 1000.

```
MAX:=0;
MIN:=1000;
FOR n:=1      TO 50  BY 1 DO
  IF DATA[n]>MAX THEN
    MAX:=DATA[n];
  END IF;
  IF DATA[n]<MIN THEN
    MIN:=DATA[n];
  END IF;
END_FOR;
```

WHILE

■ Назначение

Данная конструкция циклически выполняет указанные выражения в течение всего времени, пока соблюдается указанное условие.

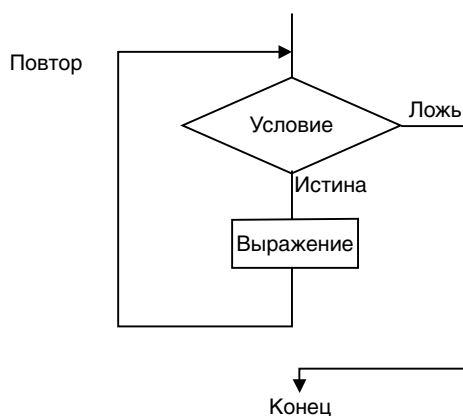
■ Резервированные слова

WHILE, DO, END_WHILE

■ Синтаксис выражения

```
WHILE <условие> DO
  <выражение>;
END_WHILE;
```

■ Блок схема выполнения



■ Использование

Цикл WHILE используется, когда количество циклов выполнения требуемых операций заранее неизвестно (и зависит от соблюдения некоторого условия). Операции выполняются, только пока условное выражение остается равным «истине» (цикл с предварительной проверкой условия выполнения).

■ Описание

Перед выполнением *выражения* проверяется *условие* .
Если *условие* верно (= истина), выполняется *выражение*. После выполнения *выражения* *условие* проверяется снова. Этот процесс циклически повторяется. Если *условие* перестает соблюдаться (= ложь), *выражение* не выполняется, проверка *условия* больше не производится.

■ Предостережения

- Слово WHILE должно использоваться со словом END_WHILE.
- Если *условное выражение* не равно «истине» (= ложь) до выполнения *выражения*, *выражение* не выполняется.
- *Выражение* может содержать выражения присваивания, IF, CASE, FOR, WHILE или REPEAT.
- *Выражение* может состоять из нескольких отдельных выражений. Выражения в пределах каждой группы выражений обязательно должны отделяться друг от друга точкой с запятой (;).
- Вместо *выражения* с оператором сравнения в *условии* также может быть указана просто переменная логического типа (BOOL).

■ Примеры

Пример 1. Переменная A циклически увеличивается на 7, пока ее значение не становится больше, чем 1000.

```
A:=0;  
WHILE A<=1000 DO  
  A:=A+7;  
END_WHILE;
```

Пример 2. Если переменная X<3000, значение X умножается на 2, результат записывается в элемент переменной-массива DATA[1]. Затем значение X вновь умножается на 2, результат записывается в элемент переменной-массива DATA[2]. Этот процесс циклически повторяется.

```
n:=1;  
WHILE X<3000 DO  
  X:=X*2;  
  DATA[n]:=X;  
  n:=n+1;  
END_WHILE;
```

REPEAT**■ Назначение**

Данная конструкция циклически выполняет некоторое выражение, пока не оказывается соблюдено указанное условие.

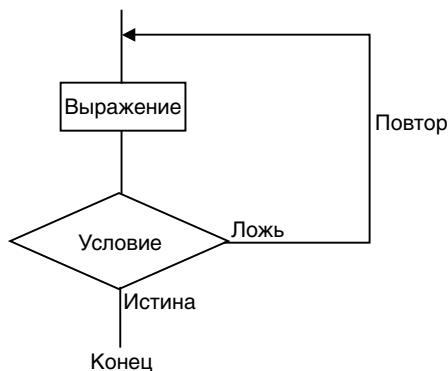
■ Резервированные слова

REPEAT, UNTIL, END_REPEAT

■ Синтаксис выражения

```
REPEAT  
  <выражение>;  
UNTIL <условие>  
END_REPEAT
```

■ Блок схема выполнения



■ Использование

Цикл REPEAT используется, если некоторую выполненную операцию необходимо повторять циклически до тех пор, пока не окажется соблюдено определенное условие. Количество циклов повтора заранее неизвестно (зависит от того, когда окажется соблюдено заданное условие). Такая конструкция позволяет анализировать результат выполнения операции и на основании этого принимать решение о необходимости повтора операции (цикл с проверкой условия после выполнения).

Описание

В первый раз *выражение* выполняется без проверки условия выполнения. Состояние *условного выражения* проверяется после выполнения *выражения*. Если *условие* не соблюдается (= ложь), *выражение* выполняется еще раз. Если *условие* соблюдается (= истина), *выражение* не выполняется, цикл на этом прекращается.

■ Предостережения

- Слово REPEAT должно использоваться со словом END_REPEAT.
- Даже если *условие* равно «истине» еще до выполнения *выражения*, *выражение* все равно будет выполнено (один раз).
- *Выражение* может содержать выражения присваивания, IF, CASE, FOR, WHILE или REPEAT.
- *Выражение* может состоять из нескольких отдельных выражений. Выражения в пределах каждой группы выражений обязательно должны отделяться друг от друга точкой с запятой (;).
- Вместо выражения с оператором сравнения в *условии* также может быть указана просто переменная логического типа (BOOL).

■ Примеры

Пример 1. Получение суммы чисел от 1 до 10 и запись суммы в переменную TOTAL.

```

A:=1;
TOTAL:=0;
REPEAT
  TOTAL:=TOTAL+A;
  A:=A+1;
UNTIL A>10
END_REPEAT;
  
```

EXIT

■ **Назначение**

Данное выражение используется только внутри конструкций цикла (FOR, WHILE, REPEAT) и служит для досрочного принудительного прекращения цикла. Данное выражение также может использоваться как часть конструкции IF внутри конструкции цикла, если досрочный выход должен происходить при наступлении определенного условия.

■ **Зарезервированные слова**

EXIT

■ **Синтаксис выражения (пример: применение в конструкции IF)**

```
FOR (WHILE, REPEAT) выражение
```

```
IF <условие> THEN EXIT;  
END_IF;
```

```
END_FOR (WHILE, REPEAT);
```

■ **Использование**

Выражение EXIT используется для принудительного прекращения циклического выполнения до наступления заданного условия выхода из цикла.

■ **Описание (пример: применение в конструкции IF)**

Если *условное_выражение* равно «истине», цикл (FOR, WHILE, REPEAT) прерывается досрочно, никакие выражения после слова EXIT не выполняются.

Примечание

- (1) Вместо выражения с оператором сравнения в *условии* также может быть указана просто переменная логического типа (BOOL).
- (2) Даже если *условие* равно «истине» еще до выполнения *выражения*, *выражение* все равно будет выполнено.

■ **Пример**

Значение переменной n изменяется от 1 до 50 с шагом 1, в каждом цикле к значению элемента переменной-массива DATA[n] добавляется значение n. Если значение DATA[n] превысит 100, цикл будет прерван.

```
FOR n:=1; TO 50 BY 1 DO  
  DATA[n]:=DATA[n]+n;  
  IF DATA[n]>100 THEN EXIT;  
END_IF;  
END_FOR;
```

RETURN

■ **Назначение**

Действие выражения RETURN зависит от того, в программе какого типа оно используется.

- Прикладная программа на языке ST:
Принудительное завершение выполняемой задачи на языке ST и переход к выполнению следующей задачи.

- Программа на языке ST в SFC:
Принудительное завершение выполняемой программы действия и переход к выполнению следующей программы действия или программы перехода.
- Программа на языке ST в функциональном блоке:
Принудительное завершение выполнения вызванного экземпляра функционального блока на языке ST, содержащего выражение RETURN, возврат в программу вызвавшего экземпляра функционального блока и выполнение команды, расположенной следом за точкой вызова.

■ **Зарезервированные слова**

RETURN

■ **Синтаксис выражения**

RETURN

■ **Использование**

Выражение RETURN используется для принудительного завершения задачи на языке ST, программы действия на языке SFC и функционального блока на языке ST.

**Выражение вызова
функционального блока**

■ **Назначение**

Это выражение служит для вызова экземпляра функционального блока.

■ **Зарезервированные слова**

Нет

■ **Синтаксис выражения**

После имени экземпляра (см. примеч.) в скобках вводятся аргументы (переменные, значения которых передаются во входные переменные вызываемого функционального блока) и возвращаемые значения (переменные, в которые возвращаются значения выходных переменных функционального блока). Существуют два варианта записи вызова экземпляра функционального блока (вариант записи 1 и вариант записи 2). Оба этих варианта подробно описаны ниже.

Примечание. Используется тип данных любой внутренней переменной функционального блока (если ST используется в экземпляре функц. блока) или глобальной переменной (если ST используется в задаче на языке ST или в программе действия на языке SFC).

■ **Вариант записи 1**

В этом варианте записи в скобках указываются как аргументы (имена переменных вызываемого определения функционального блока), так и возвращаемые значения.

$A(B:=C, \dots, D:=E)$

A: Имя экземпляра

B: Имя входной переменной вызываемого функционального блока

C: Одно из следующих значений (зависит от используемой программы на языке ST):

- входная переменная вызываемого функционального блока или константа (если ST используется в экземпляре функц. блока);

- имя глобальной или локальной переменной (если ST используется в задаче на языке ST или в программе действия на языке SFC).

D: Имя выходной переменной вызываемого функционального блока или константа

E: Одно из следующих значений (зависит от используемой программы на языке ST):

- выходная переменная вызываемого функционального блока или константа (если ST используется в экземпляре функц. блока);
- имя глобальной или локальной переменной (если ST используется в задаче на языке ST или в программе действия на языке SFC).

Примечание. После каждого выражения присваивания вида $V:=C$ ставится запятая. Запятая также ставится после каждого выражения присваивания вида $D=>E$, кроме самого последнего из них, за которым сразу следует закрывающая скобка.

■ Вариант записи 2

В данном варианте записи в скобках указываются только возвращаемые значения, а аргументы (имена переменных вызываемого определения функционального блока) не указываются.

$A(C, \dots, E)$

A: Имя экземпляра

V: Опускается (имя входной переменной вызываемого функционального блока)

C: Одно из следующих значений (зависит от используемой программы на языке ST):

- входная переменная вызываемого функционального блока или константа (если ST используется в экземпляре функц. блока);
- имя глобальной или локальной переменной (если ST используется в задаче на языке ST или в программе действия на языке SFC).

D: Опускается (имя выходной переменной вызываемого функционального блока или константа)

E: Одно из следующих значений (зависит от используемой программы на языке ST):

- выходная переменная вызываемого функционального блока или константа (если ST используется в экземпляре функц. блока);
- имя глобальной или локальной переменной (если ST используется в задаче на языке ST или в программе действия на языке SFC).

Примечание. Если аргументы V и D не указываются (как показано выше), значения переменных группы C автоматически передаются в вызываемый функциональный блок в том порядке, в котором эти значения зарегистрированы в таблице переменных. В свою очередь, значения выходных переменных, возвращаемые вызываемым функциональным блоком, автоматически передаются в группу переменных E в том порядке, в котором эти значения зарегистрированы в таблице переменных.

■ **Использование**

Выражение вызова функционального блока используется для вызова определения функционального блока (написанного на языке ST или LD) из программы, написанной на языке ST.

■ **Описание**

1. В таблице переменных (внутренних или глобальных) регистрируется экземпляр функционального блока с указанными ниже параметрами.

Параметр внутренней переменной	Содержание	Пример
Имя	Любое имя экземпляра	Calcu_execute
Тип данных	FUNCTION BLOCK	FUNCTION BLOCK
Опред. функц. блока	Выбранное определение функционального блока	Calculation

2. После имени экземпляра, зарегистрированного на шаге 1 (Calcu_execute в данном примере), в скобках перечисляются переменные, между которыми будет производиться обмен значениями, либо сами значения. Запись выражения завершается точкой с запятой (см. пример ниже).

Calcu_execute (A:=B,C=>D) ;

Значение B передается в A, значение C возвращается в D.

A: Имя входной переменной вызываемого функционального блока

B: Одно из следующих значений (зависит от используемой программы на языке ST):

- входная переменная вызываемого функционального блока или константа (если ST используется в экземпляре функц. блока);
- имя глобальной или локальной переменной (если ST используется в задаче на языке ST или в программе действия на языке SFC).

C: Имя выходной переменной вызываемого функционального блока или константа

D: Одно из следующих значений (зависит от используемой программы на языке ST):

- выходная переменная вызываемого функционального блока или константа (если ST используется в экземпляре функц. блока);
- имя глобальной или локальной переменной (если ST используется в задаче на языке ST или в программе действия на языке SFC).

■ **Дополнительные примеры**

Приведенные ниже примеры более подробно поясняют использование выражения вызова функционального блока в программе на языке структурированного текста.

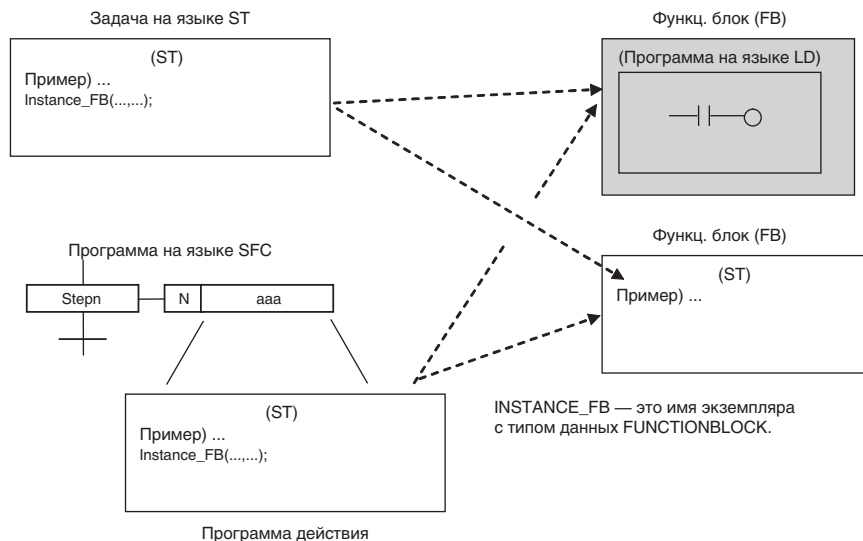
Пример 1.

Данная группа примеров иллюстрирует вызов функционального блока из программ разного типа (задача на языке ST, SFC и функциональный блок).

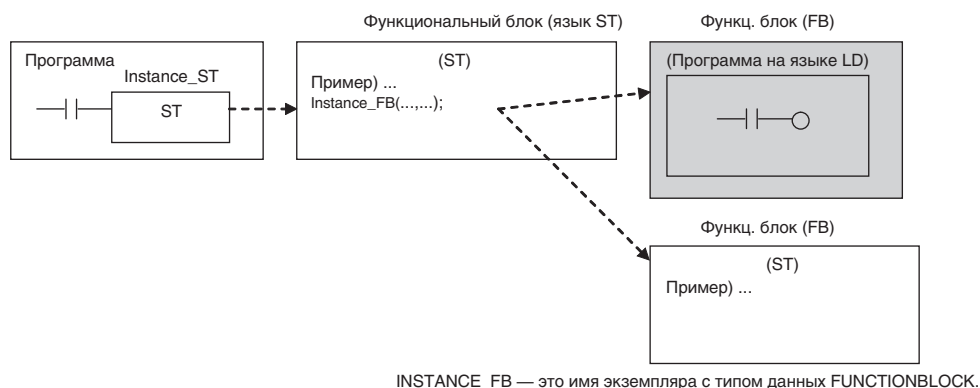
- Общие условия:
 Вызывается функциональный блок.
 Вызываемый функциональный блок написан на языке LD или ST.

Подробное описание вызова

a. Вызов функционального блока из задачи на языке ST или программы на языке SFC



b. Вызов функционального блока из другого функционального блока



Описание переменных

- Описание переменных вызывающей программы на языке ST или SFC

В следующей таблице перечислены переменные программы на языке ST или SFC, а также соответствующие им переменные в вызываемом функциональном блоке.

Имя переменной в программе задачи ST/SFC	Соответствующие переменные в вызываемом функциональном блоке
IN1	FB_IN1 (входная переменная, принимает значение)
IN2	FB_IN2 (входная переменная, принимает значение)
IN3	FB_IN3 (входная переменная, принимает значение)
OUT1	FB_OUT1 (выходная переменная, возвращает значение)

Имя переменной в программе задачи ST/SFC	Соответствующие переменные в вызываемом функциональном блоке
OUT2	FB_OUT2 (выходная переменная, возвращает значение)
OUT3	FB_OUT3 (выходная переменная, возвращает значение)
A Примечание. Тип данных: BOOL	EN (внутренняя переменная, принимает значение)
B Примечание. Тип данных: BOOL	ENO (внутренняя переменная, возвращает значение)
Instance_FB Примечание. Тип данных: FUNCTIONBLOCK	Определение вызывающего функционального блока: функциональный блок

- Описание переменных вызывающего функционального блока

В следующей таблице перечислены переменные вызывающего функционального блока и соответствующие им переменные вызываемого функционального блока.

Тип переменной	Имя переменной в вызываемом функциональном блоке	Соответствующие переменные в вызываемом функциональном блоке
Входные переменные	IN1	FB2_IN1 (принимает значение)
	IN2	FB2_IN2 (принимает значение)
	IN3	FB2_IN3 (принимает значение)
Выходные переменные	OUT1	FB2_OUT1 (возвращает значение)
	OUT2	FB2_OUT2 (возвращает значение)
	OUT3	FB2_OUT3 (возвращает значение)
Внутренние переменные	A Примечание. Тип данных: BOOL	EN (принимает значение)
	B Примечание. Тип данных: BOOL	ENO (возвращает значение)
Внутренние переменные (экземпляр)	Instance_FB Примечание. Тип данных: FUNCTIONBLOCK	Определение вызывающего функционального блока: функциональный блок 2

- Описание переменных вызываемого функционального блока

В следующей таблице перечислены переменные вызываемого функционального блока и соответствующие им переменные в источнике вызова (программа на языке ST, программа на языке SFC или функциональный блок).

Тип переменной	Имя переменной в вызываемом функциональном блоке	Соответствующие переменные в источнике вызова
Входные переменные	FB_IN1	IN1 (передает значение)
	FB_IN2	IN2 (передает значение)
	FB_IN3	IN3 (передает значение)
Выходные переменные	FB_OUT1	OUT1 (принимает значение)
	FB_OUT2	OUT2 (принимает значение)
	FB_OUT3	OUT3 (принимает значение)

Примеры

■ Пример варианта записи 1

```
Instance_FB(EN:=A,FB_IN1:=IN1,FB_IN2:=IN2,FB_IN3:= IN3,
FB_OUT1=>OUT1,FB_OUT2=> OUT2,FB_OUT3=> OUT3,ENO=>B)
```

- Аргументы и возвращаемые значения могут указываться в произвольном порядке.
- Входные переменные должны находиться в начале списка или сразу после переменной EN (при ее наличии).
- Выходные переменные могут не указываться, если они не используются.
- В одном выражении вызова функционального блока нельзя использовать одновременно способ указания переменных 2 и способ указания переменных A.

■ Примеры других вариантов записи

- Без указания переменной EN:

```
Instance_FB(FB_IN1:=IN1,FB_IN2:=IN2,FB_IN3:= IN3,
FB_OUT1=>OUT1,FB_OUT2=> OUT2,FB_OUT3=> OUT3,ENO=>B)
```
- Без указания переменных EN и ENO:

```
Instance_FB(FB_IN1:=IN1,FB_IN2:=IN2,FB_IN3:= IN3,
FB_OUT1=>OUT1,FB_OUT2=> OUT2,FB_OUT3=> OUT3)
```
- Без указания переменной ENO:

```
Instance_FB(EN:=A,FB_IN1:=IN1,FB_IN2:=IN2,FB_IN3:=IN3,
FB_OUT1=>OUT1,FB_OUT2=>OUT2,FB_OUT3=>OUT3)
```
- Без указания переменной FB_OUT2, значение которой не используется:

```
Instance_FB(EN:=A,FB_IN1:=IN1,FB_IN2:=IN2,FB_IN3:=IN3,
FB_OUT1=>OUT1,FB_OUT3=>OUT3,ENO=>B)
Instance_FB(FB_IN1:=IN1,FB_IN2:=IN2,FB_IN3:=IN3,
FB_OUT1=>OUT1,FB_OUT3=>OUT3)
```
- Ввод переменных в другом порядке:

```
Instance_FB(EN:=A,FB_IN1:=IN1,FB_OUT1=> OUT1,FB_IN2:=IN2,
FB_OUT2=>OUT2,FB_IN3:= IN3,FB_OUT3=> OUT3,ENO=>B)
```

■ **Пример варианта записи 2**

В данном примере вводятся только переменные параметров (в том числе константы), относящиеся к новому экземпляру.

Instance_FB(IN1, IN2, IN3, OUT1, OUT2, OUT3)

Instance_FB(IN1, IN2, IN3, OUT1)

- Аргументы и возвращаемые значения должны указываться в строго определенном порядке, а именно: входная переменная 1, входная переменная 2, ..., выходная переменная 1, выходная переменная 2, ...
- Входные переменные должны располагаться в начале списка или сразу после переменной EN (если она указывается).
- Если значение некоторой выходной переменной фактически не используется и переменная располагается не внутри списка выходных переменных, эту выходную переменную можно не указывать .

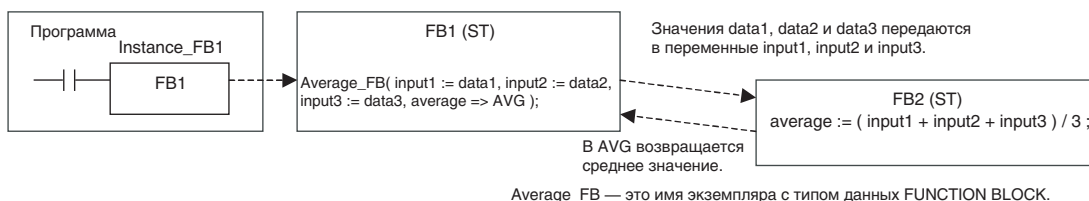
Пример: Instance_FB(IN1, IN2, IN3, OUT1, OUT3)

В данном примере расположенная в конце списка переменная OUT3 возвращает значение из переменной FB_OUT2.

- Переменные EN и ENO нельзя вводить в качестве аргументов или возвращаемых значений.
- В одном выражении вызова функционального блока нельзя использовать одновременно способ указания переменных 1 и способ указания переменных В.

■ **Пример 2**

В следующем примере из функционального блока 1 вызывается функциональный блок 2, вычисляющий среднее значение.



Функциональный блок 1

- Таблица переменных

Тип переменной	Имя переменной	Тип данных	Передача/получение в/из FB2
Входная переменная	EN	BOOL	---
Входная переменная	data1	INT	input1 (получает значение)
Входная переменная	data2	INT	input2 (получает значение)
Входная переменная	data3	INT	input3 (получает значение)
Входная переменная	bCheck	BOOL	---
Выходная переменная	ENO	BOOL	---

Тип переменной	Имя переменной	Тип данных	Передача/получение в/из FB2
Выходная переменная	AVG	INT	average (возвращает значение)
Внутренняя переменная	Average_FB	FUNCTION BLOCK Определение вызываемого функционального блока: Функциональный блок 2	---

- Работа программы на языке ST

Если bCheck = истина, вызывается функциональный блок 2 для вычисления среднего значения. Значения переменных data1, data2 и data3 передаются, соответственно, во входные переменные input1, input2 и input3 функционального блока 2. В переменную AVG возвращается результат вычисления (*average*).

Примечание. В приведенном ниже фрагменте программы для вызова функционального блока Average_FB используется способ указания переменных A (указываются переменные обоих функциональных блоков).

```
IF bCheck = TRUE THEN
```

```
  Average(input1:=data1, input2:=data2, input3:=data3, average
=>AVG);
```

```
ELSE
```

```
  RETURN;
```

```
END_IF;
```

Функциональный блок 2

- Таблица переменных

Тип переменной	Имя переменной	Тип данных	Передача/получение в/из FB1
Входная переменная	EN	BOOL	---
Входная переменная	input1	INT	data1 (принимает значение)
Входная переменная	input2	INT	data2 (принимает значение)
Входная переменная	input3	INT	data3 (принимает значение)
Выходная переменная	ENO	BOOL	---
Выходная переменная	average (возвращает значение)	INT	AVG

- Работа программы на языке ST

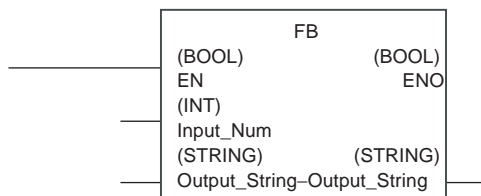
Программа вычисляет среднее арифметическое значение переменных input1, input2 и input3 и записывает результат в переменную *average*.

```
average := (input1+input2+input3) / 3;
```

5-6 Пример программы на языке структурированного текста

5-6-1 Использование программы на языке ST в функциональном блоке

Преобразование целого числа в двоично-десятичное и вывод в виде текстовой строки



Входная переменная

```
INT Input_Num;
```

Вх.-вых. переменная

```
STRING Output_String;
```

Внутренняя переменная

```
WORD Num_BCD;
```

```
(*Проверка входного параметра Input_BCD (BCD-значение*)
```

```
IF (Input_BCDNum >= 0 & Input_BCD <= 16 # Num <= 9999) THEN
```

```
    ENO := true;
```

```
ELSE
```

```
    ENO := false;
```

```
    RETURN;
```

```
END_IF;
```

```
Num_BCD := INT_TO_BCD_WORD (Input_Num) ;
```

```
(*Пример: если Num = 100 (16#0064), то BCD-значение = 0100*)
```

```
Output_String := WORD_TO_STRING (Num_BCD) ;
```

```
(*Преобразование BCD-числа 0100 в текстовую строку*)
```

5-7 Ограничения

5-7-1 Ограничения

■ Вложение конструкций

- Количество уровней вложения конструкций IF, CASE, FOR, WHILE или REPEAT ничем не ограничено.

■ Ограничение на используемые типы данных

- Переменные типа WORD, DWORD, INT, DINT, UINT, UDINT или ULINT могут принимать только целочисленные значения. Например, если A является переменной типа INT, выражение A:=1; допустимо. При попытке присвоения нецелого значения произойдет ошибка синтаксиса. Например, если A является переменной типа INT, выражение A:=2.5; вызовет ошибку синтаксиса.

- Переменные типа REAL и UREAL могут принимать только вещественные значения (десятичные значения с плавающей точкой). Например, если A является переменной типа REAL, выражение A:=1.5; возможно. При попытке присвоения значения, не являющегося вещественным, произойдет ошибка синтаксиса. Например, если A является переменной типа REAL, запись A:=2; вызовет ошибку синтаксиса. В данном случае следует использовать запись A:=2.0;.
- Переменные типа BOOL могут принимать только битовые (логические) значения (TRUE, FALSE). Например, если A является переменной типа BOOL, выражение A:=FALSE; допустимо. При попытке присвоения значения, не являющегося логическим (BOOL), произойдет ошибка синтаксиса. Например, если A является переменной типа BOOL, выражение A:=5; вызовет ошибку синтаксиса.
- Типы данных, используемые в выражениях, должны согласоваться друг с другом. Например, если A, B и C являются переменными типа INT, выражение A:=B+C; допустимо. Однако, если A и B являются переменными типа INT, а C является переменной типа REAL или LINT, выражение A:=B+C; вызовет ошибку синтаксиса.
- В программе на языке структурированного текста не допускается использовать следующие переменные:
P_CY, P_EQ, P_ER, P_N, P_GE, P_GT, P_LE, P_LT, P_NE, P_OF и P_UF

■ Ограничения при мониторинге

- Следующие ограничения действуют в отношении мониторинга функции таймера:
Если переменная типа TIMER используется в команде «0,1 мс ТАЙМЕР» или «0,01 мс ТАЙМЕР», текущее значения переменной типа TIMER в окне мониторинга программы на языке ST не отображается. Вместо него в этом случае отображается прочерк («-»).
- Если текущее значение переменной типа TIMER используется также в другом месте окна редактирования программы на языке ST или присвоено другой переменной, все эти текущие значения будут недостоверными.
- Следующие ограничения действуют в отношении использования 2-байтовых символов в именах переменных:
Если в имени переменной используются любые 2-байтовые символы, между именем переменной и оператором необходимо вставлять 1-байтовый знак пробела. Без этого пробела, возможно, не удастся корректно осуществлять мониторинг текущего значения переменной.

5-7-2 Часто задаваемые вопросы

В: В какой форме должно записываться шестнадцатеричное значение?

О: Значение должно записываться с префиксом 16#. Пример: 16#123F. Аналогичным образом, для записи восьмеричных и двоичных чисел к ним необходимо добавлять префиксы 8# и 2#. Если число будет записано без любого из этих префиксов, оно будет воспринято, как десятичное число.

В: Сколько раз выполняется цикл FOR?

О: В приведенном ниже примере содержимое конструкции FOR выполняется 101 раз. Цикл прекращается, когда значение переменной *i* становится равным 101.

```
FOR i:=0 TO 100 BY 1 DO  
  a:=a+1;  
END_FOR;
```

В: Что происходит, если указанный индекс массива превосходит размер массива?

О: Например, если имеется массив переменных INT[10], содержащий 10 элементов, в приведенном ниже фрагменте программы ошибка обнаружена не будет. Однако выполнение такой программы приведет к непредсказуемому результату.

```
i:=15;  
INT[i]:=10;
```

В: Регистрируются ли автоматически переменные в таблице переменных, когда они вводятся в программу на языке ST в окне редактора языка ST?

О: Нет. Перед использованием переменных их необходимо зарегистрировать в таблице переменных.

В: Можно ли в программе на языке ST непосредственно использовать команды языка LD?

О: Нет.

РАЗДЕЛ 6

Создание программ на языке структурированного текста

Данный раздел подробно описывает процесс создания программ на языке структурированного текста.

6-1	Порядок действий	208
6-1-1	Создание проекта	208
6-1-2	Создание новой программы на языке ST	208
6-1-3	Назначение программы на языке ST задаче	209
6-1-4	Создание программы на языке ST	210
6-1-5	Компилирование программ на языке ST (проверка программы)	215
6-1-6	Загрузка программ в модуль ЦПУ и их считывание из модуля ЦПУ	215
6-1-7	Сравнение программ на языке ST	216
6-1-8	Мониторинг и отладка программ на языке ST	216
6-1-9	Редактирование программ на языке ST в режиме онлайн	217

6-1 Порядок действий

Данный раздел подробно описывает процесс создания программ на языке структурированного текста. Подробную информацию о создании функциональных блоков с применением языка структурированного текста см. в РАЗДЕЛ 3 Создание функциональных блоков, Часть 1: Функциональные блоки в настоящем руководстве.

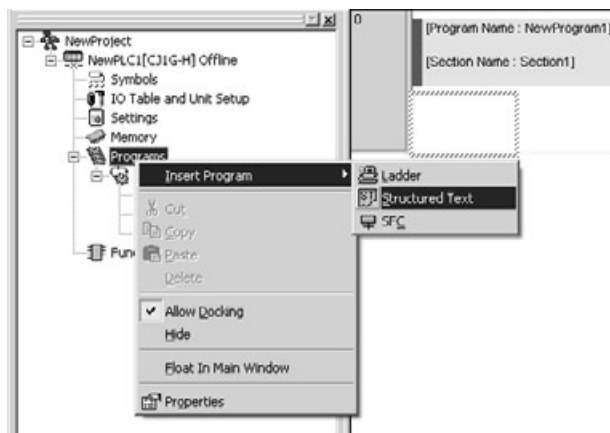
6-1-1 Создание проекта

- 1,2,3...
1. Запустите CX-Programmer и выберите команду **New (Создать)** в меню **File (Файл)**.
 2. В диалоговом окне Change PLC (Изменение ПЛК) в списке *Device Type (Тип устройства)* выберите модель ПЛК, поддерживающую язык структурированного текста. Список моделей ПЛК, поддерживающих программирование на языке ST, приведен в разделе 4-2-1 *Совместимость моделей ПЛК с языком программирования ST (задачи на языке ST)*.
 3. Щелкните кнопку **Settings (Настройка)** и выберите модель ЦПУ в поле *CPU Type*. Подробную информацию о настройке других параметров можно найти в руководстве *CX-Programmer — Руководство по работе (W446)*.

6-1-2 Создание новой программы на языке ST

Для создания программы на языке ST в проекте CX-Programmer соблюдайте следующий порядок действий.

- 1,2,3...
1. Щелкните правой кнопкой мыши по узлу **Programs (Программы)** на дереве проекта.
 2. Выберите команду **Insert Program – ST (Вставить программу – ST)** в отобразившемся контекстном меню.



На дерево проекта будет добавлена программа на языке ST, а справа от окна дерева проекта отобразится окно редактирования программы на языке ST (ST Editor).

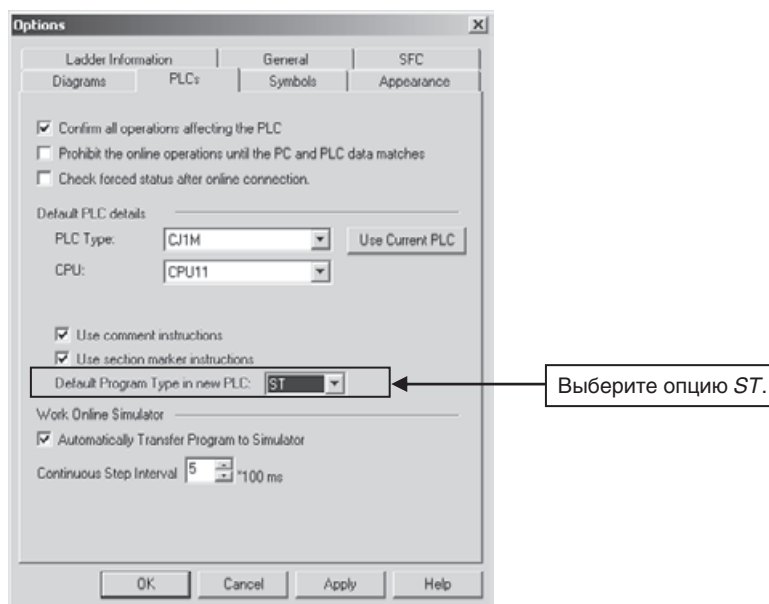
Примечание

- (1) Таким же образом могут быть созданы программы на языке LD и SFC. Для этого следует щелкнуть правой кнопкой мыши по узлу **Programs (Программы)** на дереве проекта и выбрать, соответственно, команду **Insert Program – Ladder (Вставить программу – LD)** или **Insert Program – SFC (Вставить программу – SFC)** в контекстном меню.

Подробную информацию о программировании на языке LD см. в руководстве *CX-Programmer — Руководство по работе (W446)*.

Подробную информацию о программировании на языке SFC см. в руководстве *CX-Programmer — Руководство по работе: SFC (W469)*.

- (2) После того как новый проект создан, для ПЛК можно выбрать используемый по умолчанию язык программирования. Чтобы выбрать язык ST, вызовите окно **Tools – Options (Сервис – Настройка)**, откройте вкладку **PLCs (ПЛК)** и выберите ST в поле, которое показано на рисунке ниже.



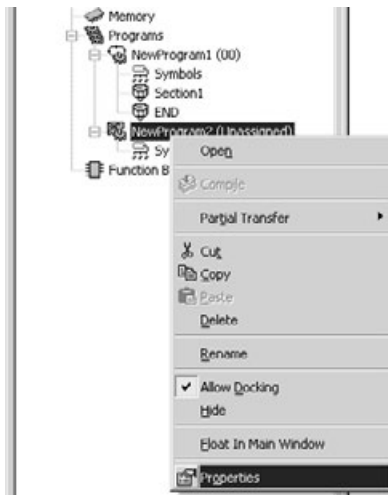
6-1-3 Назначение программы на языке ST задаче

Программа на языке ST, вставленная в проект, должна быть назначена некоторой задаче прикладной программы (задача является минимальной структурной единицей прикладной программы). Если программа еще не назначена никакой задаче, поверх значка этой программы на дереве проекта отображается крестик.

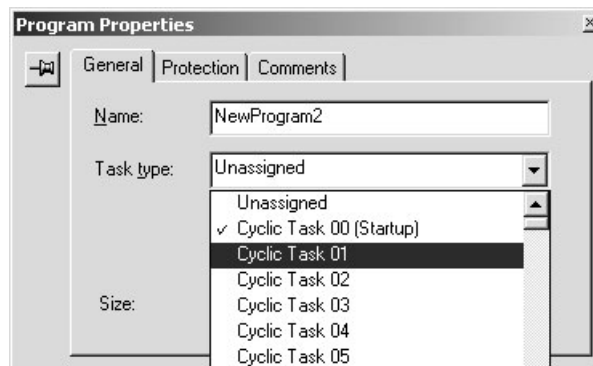
Примечание. Описанный ниже порядок действий позволяет назначить программу задаче после того, как программа создана, но программы обязательно должны назначаться задачам до загрузки программы пользователя в ПЛК.

Для того чтобы назначить созданную программу задаче, выполните следующие действия.

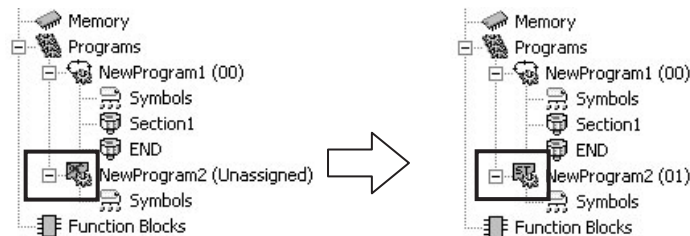
- 1,2,3... 1. Щелкните правой кнопкой мыши по узлу программы на языке ST на дереве проекта и выберите **Properties (Свойства)** в контекстном меню.



2. Откройте вкладку **General (Общие)** в отобразившемся диалоговом окне Program Properties (Свойства программы) и выберите требуемую задачу в списке **Task Type (Тип задачи)**. Введите имя программы в текстовом поле **Name (Имя)** на этой же вкладке.



3. Закройте диалоговое окно Program Properties (Свойства программы), щелкнув кнопку **Close (Закрывать)**.



4. После того как программа на языке ST назначена задаче, крестик поверх ее значка на дереве проекта больше не отображается. В скобках после имени программы будет отображаться номер задачи, которой она назначена.

6-1-4 Создание программы на языке ST

Процесс создания программы на языке ST может проходить по одному из следующих сценариев.

- Сначала регистрируются переменные, а после вводится программа.

- Переменные регистрируются в процессе ввода программы.

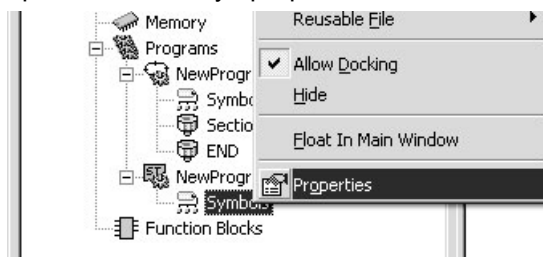
Ввод программы на языке ST после регистрации переменных

Различают переменные двух типов: глобальные переменные и локальные переменные. В этом разделе поясняется создание локальных переменных. Информацию о создании глобальных переменных можно найти в руководстве *CX-Programmer — Руководство по работе (W446)*.

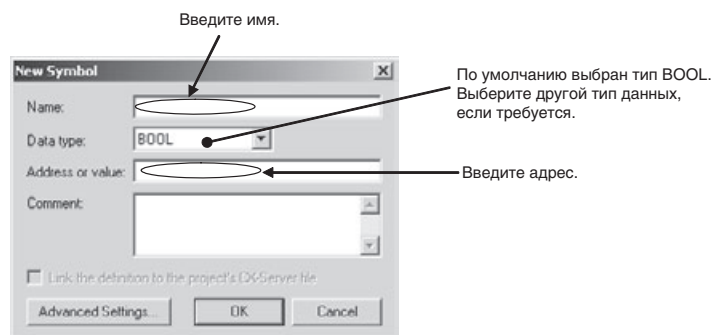
1. Регистрация переменных (с локальными адресами)

1,2,3...

1. На дереве проекта дважды щелкните по узлу **Symbols (Символы)**, принадлежащему программе на языке ST.



2. Отобразится таблица символов. Вызовите контекстное меню щелчком правой кнопки мыши и выберите команду **Insert Symbol (Вставить символ)**. (Можно также выбрать команду **Symbol (Символ)** в меню **Insert (Вставка)**.)
3. Отобразится диалоговое окно New Symbol (Создание символа). Задайте указанные ниже параметры и щелкните кнопку **OK**.
 - Name (Имя): введите имя переменной.
 - Data type (Тип данных): выберите тип данных.
 - Address or Value (Адрес или значение): введите адрес.



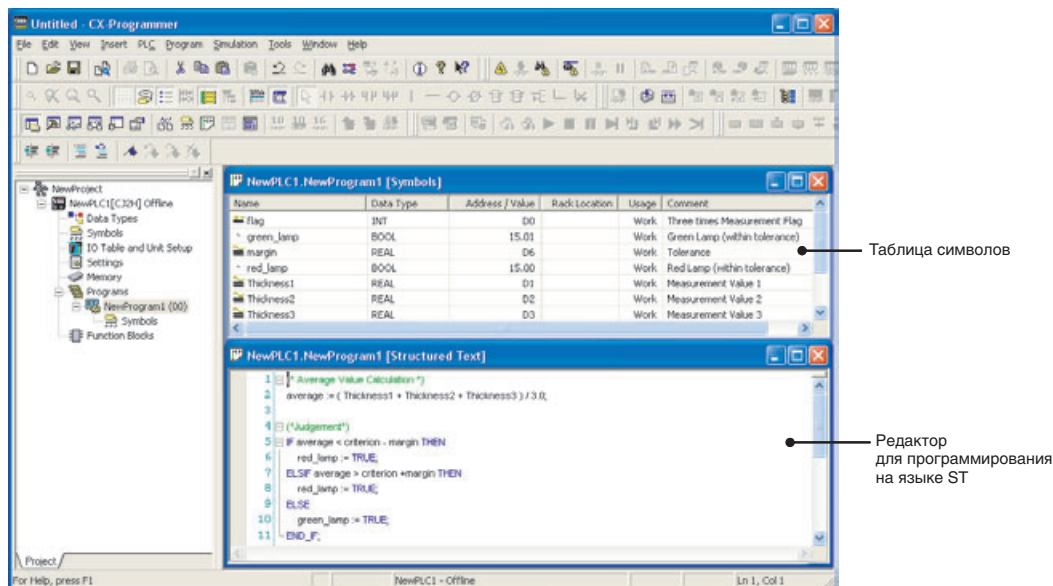
4. Переменная, сконфигурированная в таблице символов, будет зарегистрирована.

Примечание.

Чтобы не указывать для регистрируемых переменных конкретные адреса, можно воспользоваться предусмотренной в *CX-Programmer* функцией автоматического назначения адресов переменным. Подробное описание этой функции приведено в руководстве *CX-Programmer — Руководство по работе (W446)*.

2. Создание программы на языке ST

- 1,2,3... 1. Текст программы на языке ST можно вводить непосредственно в окне редактирования программы на языке ST (ST Editor) в CX-Programmer. Можно также сначала набрать программу в любом текстовом редакторе, а затем вставить ее в окно редактирования в CX-Programmer, выбрав команду **Paste (Вставить)** в меню **Edit (Правка)**.



Для ускорения ввода имен переменных в программу рекомендуется отобразить окно таблицы символов непосредственно над окном редактирования программ.

Примечание

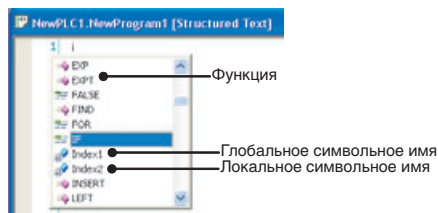
- (1) При вводе программы допускается использовать знаки «пробела» и «табуляции». На выполнение программы они не влияют.
- (2) Как при непосредственном наборе программы на языке ST в окне ввода программы, так и после вставки программы из текстового редактора все зарезервированные ключевые слова автоматически отображаются шрифтом синего цвета, комментарии отображаются зеленым цветом, текстовые строки — коричневым цветом, ошибки — красным цветом, а все остальное — черным цветом.
- (3) Для того чтобы изменить размер или цвет шрифта, выберите команду **Options (Настройка)** в меню Tools (Сервис), после чего щелкните кнопку **ST Font (Шрифт ST)** на вкладке Appearance (Вид). В данном окне можно поменять шрифт, размер шрифта (по умолчанию установлено 8 точек) и его цвет.
- (4) Информацию о характеристиках языка структурированного текста вы можете найти в **РАЗДЕЛ 5 Характеристики языка структурированного текста, Часть 2: Структурированный текст (ST)** в настоящем руководстве.

Регистрация переменных при использовании структурированного текста

Если программа создается на языке структурированного текста и в нее вводится не зарегистрированное ранее имя переменной, диалоговое окно регистрации переменной при этом не вызывается. Следует обязательно зарегистрировать все новые переменные в таблице переменных. Это можно сделать либо сразу при вводе переменных в программу, либо по завершении создания программы.

Ввод функций и переменных

- 1,2,3... 1. При вводе первой буквы функции или зарегистрированной переменной в окне редактора программы на языке ST отображается список ключевых слов.



Принадлежность ключевого слова функции или переменной можно определить по значку слева от ключевого слова.

Значок	Тип ключевого слова
	Функция
	Элементы выражений языка ST
	Глобальное символьное имя
	Локальное символьное имя Символьное имя функционального блока

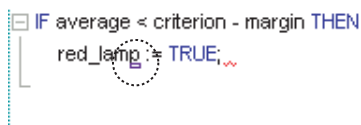
2. Выделите требуемую функцию или переменную и нажмите клавишу **Ввод**, **пробел** или **Tab**. Выбранная функция или переменная будет вставлена в программу в окне редактирования.

Примечание (a) Чтобы закрыть список без вставки переменной или функции в программу, нажмите клавишу **Esc**.
 (b) Функции и переменные можно вводить в программу в окне редактирования непосредственно, не выбирая их в списке ключевых слов.

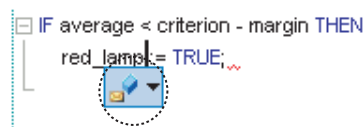
Регистрация переменных в процессе ввода программы на языке ST

В отличие от программирования на языке LD, при написании программы на языке ST ввод незарегистрированной переменной не сопровождается автоматическим отображением диалогового окна с предложением зарегистрировать переменную в таблице символов.

- 1,2,3... 1. По завершении ввода строки (т. е. после нажатия клавиши **Enter**) все незарегистрированные переменные в этой строке выделяются двойным подчеркиванием.



2. При наведении указателя мыши на двойную линию подчеркивания отображается кнопка.



Оказание помощи при вводе в редакторе программ на языке ST

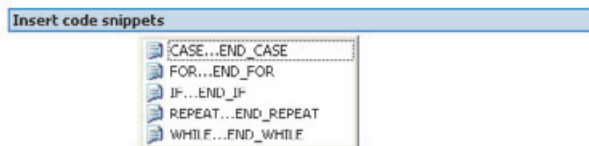
1,2,3...

3. Щелчок по этой кнопке вызывает диалоговое окно, в котором можно зарегистрировать новую переменную. Настройте каждый параметр в этом диалоговом окне и щелкните кнопку **ОК**.

1) Ввод управляющего выражения.

Для быстрого ввода управляющей конструкции можно использовать любой из двух способов, описанных ниже.

1. Выберите **Insert Code Snippets (Вставить ключевые слова)** в контекстном меню и выберите требуемые ключевые слова в списке, вид которого показан ниже.



2. В предоставленном списке ключевых слов выберите первый элемент управляющей конструкции, введите его и нажмите клавишу **Tab**.

Ключевое слово	Управляющая конструкция
IF	<pre>IF expression THEN ELSE END_IF;</pre>
FOR	<pre>FOR variable := expression1 TO expression2 BY expression3 DO END_FOR;</pre>
CASE	<pre>CASE expression OF ELSE END_CASE;</pre>
REPEAT	<pre>REPEAT UNTIL expression END_REPEAT;</pre>
WHILE	<pre>WHILE expression DO END_WHILE;</pre>

Примечание.

Для каждого *выражения (expression)* и *переменной (variable)* должно быть введено условие выполнения управляющей конструкции.

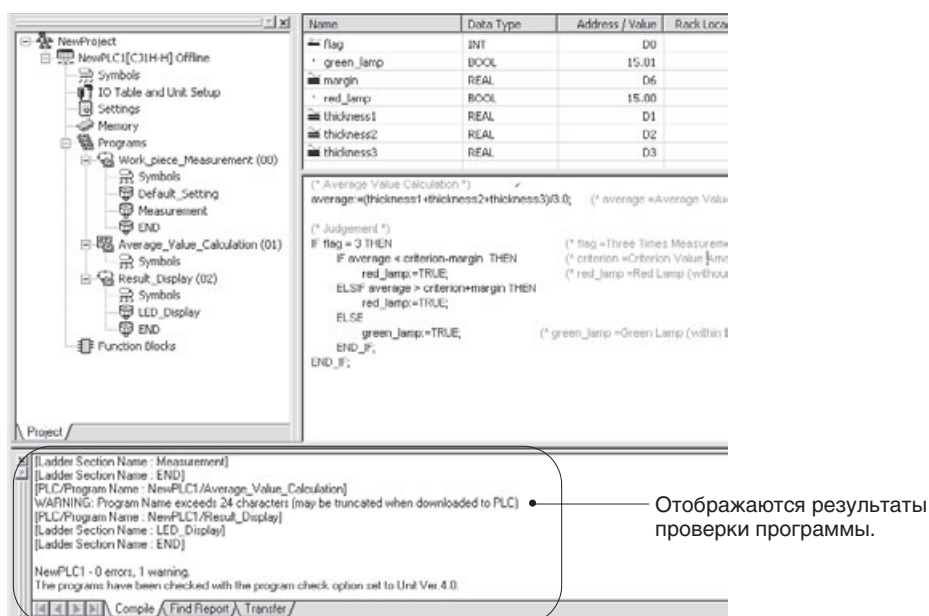
2) Для выбранного ключевого слова отображается подсказка с информацией о содержании этого ключевого слова. Оперативно узнавая о предназначении тех или иных ключевых слов и просматривая комментарии к переменным, пользователь может быстро выбирать элементы, наиболее соответствующие его цели.

- 3) С помощью специальной команды меню или кнопки на панели инструментов можно увеличивать или уменьшать отступ каждой строки.
- 4) С помощью специальной команды меню или кнопки на панели инструментов можно выбрать, как должна восприниматься выделенная строка: как комментарий или как часть программы.

6-1-5 Компилирование программ на языке ST (проверка программы)

Программу на языке структурированного текста можно проверить, выполнив для нее операцию компиляции. Выполните следующие действия.

- 1,2,3... 1. Выделите требуемую программу на языке ST, щелкните по ней правой кнопкой мыши и выберите **Compile (Компилировать)** в контекстном меню. (Другой способ: нажать клавиши **Ctrl + F7**.)
2. Для выбранной программы на языке ST будет выполнена операция компиляции, результаты проверки программы автоматически отобразятся на вкладке Compile (Компиляция) в окне вывода информации.



Отображаются результаты проверки программы.

6-1-6 Загрузка программ в модуль ЦПУ и их считывание из модуля ЦПУ

Завершив создание прикладной программы, содержащей задачи на языке ST, пользователь может перейти в режим онлайн, установив связь между CX-Programmer и подключенным ПЛК, и загрузить программу в этот ПЛК. Возможно также и противоположное действие, то есть считывание программы из подключенного ПЛК в CX-Programmer.

Возможность индивидуальной загрузки или считывания задач прикладной программы отсутствует.

6-1-7 Сравнение программ на языке ST

Содержимое редактируемой программы на языке ST можно сравнить с содержимым блока программы на языке ST в ПЛК или в другом проекте, чтобы проверить их идентичность. Подробнее о функции сравнения программ можно прочитать в руководстве *CX-Programmer — Руководство по работе (W446)*.

6-1-8 Мониторинг и отладка программ на языке ST

Мониторинг переменных программы на языке ST

В CX-Programmer поддерживается мониторинг программ на языке ST.

В левой части окна (называемой окном мониторинга программы на языке ST) отображается программа на языке ST.

В правой части окна (называемой окном мониторинга переменных программы на языке ST или, просто, окном мониторинга переменных ST) отображаются значения переменных, используемых в программе на языке ST.

Здесь можно наблюдать за значениями переменных, изменять текущие значения, принудительно устанавливать/сбрасывать биты, копировать и вставлять переменные в окно таблицы мониторинга. (Эти операции подробно описаны ниже.)

Мониторинг переменных

Значения переменных отображаются в окне мониторинга переменных ST шрифтом синего цвета.

Примечание.

Для переменных типа TIMER в командах «0,1 мс ТАЙМЕР» или «0,01 мс ТАЙМЕР» в качестве текущего значения отображается прочерк («-»). Подробную информацию см. в разделе 5-7 *Ограничения*.

Изменение текущих значений

Для изменения текущего значения выберите требуемую переменную в окне мониторинга переменных ST (при выделении отображается в инверсном виде), щелкните правой кнопки мыши и выберите пункт **Set – Value (Задать – Значение)** в контекстном меню.



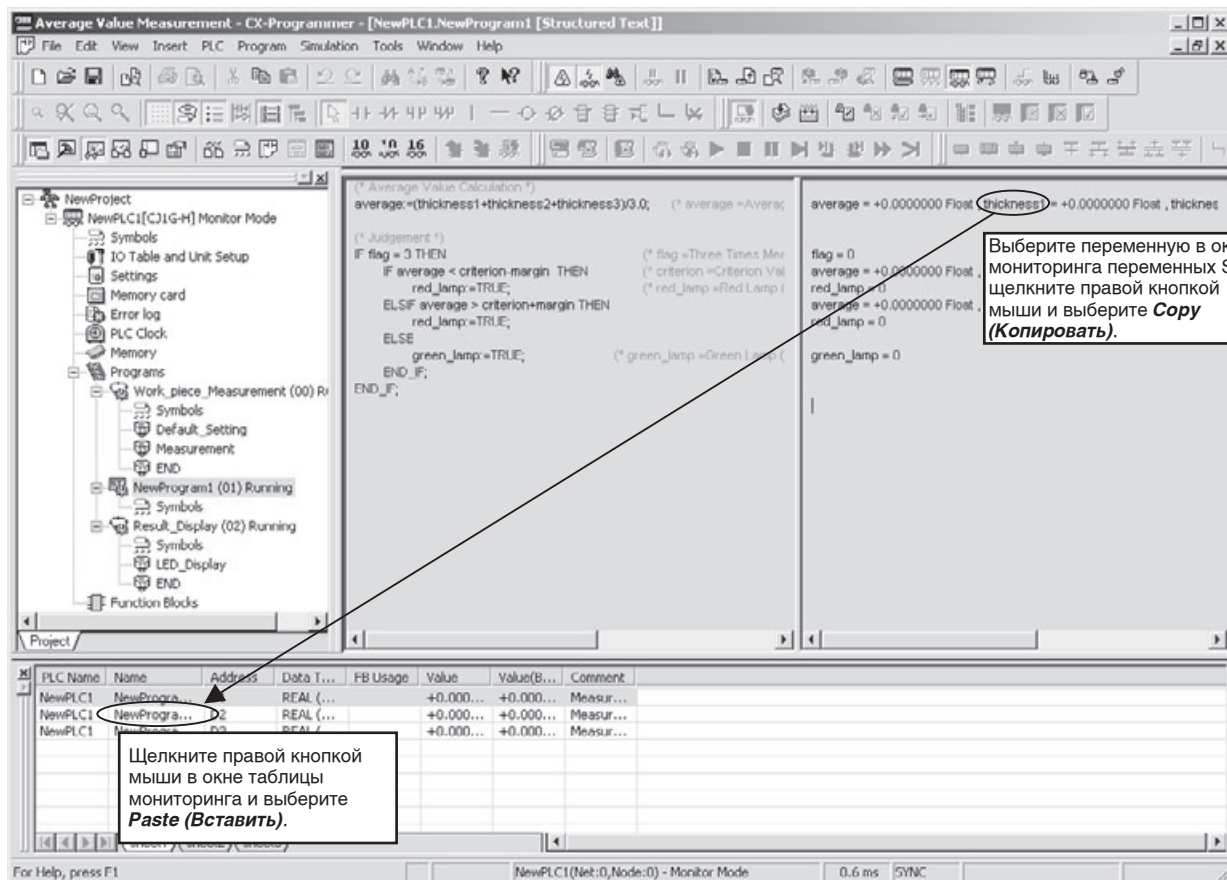
Отобразится диалоговое окно Set New Value (Ввод нового значения). Введите новое значение в поле Value (Значение).

Принудительная установка/сброс битов

Чтобы выполнить принудительную установку/сброс бита или отменить принудительно установленные состояния, выберите требуемую переменную в окне мониторинга переменных ST (при выделении отображается в инверсном виде), щелкните правой кнопкой мыши и выберите пункт **Force – On (Принудительно – Включить)**, **Force – Off (Принудительно – Выключить)**, **Force – Cancel (Принудительно – Отменить)** или **Force – Cancel All Forces (Принудительно – Отменить все)** в контекстном меню.

Копирование и вставка в окне таблицы мониторинга

- 1,2,3...
1. Для того чтобы скопировать переменную и вставить ее в окно таблицы мониторинга, выберите требуемую переменную в окне мониторинга переменных ST (при выделении отображается в инверсном виде), щелкните правой кнопки мыши и выберите пункт **Сору (Копировать)** в контекстном меню.
 2. Правой кнопкой мыши щелкните по окну таблицы мониторинга и выберите пункт **Paste (Вставить)** в контекстном меню.



Функция имитации выполнения программы на языке ST

В CX-Programmer предусмотрена возможность отладки и мониторинга программы на языке ST в эмуляторе режима выполнения.

6-1-9 Редактирование программ на языке ST в режиме онлайн

Редактирование программ, написанных на языке структурированного текста, возможно даже во время работы ПЛК (модуля ЦПУ). Благодаря этому отладку и правку программ на языке ST можно производить в круглосуточно работающих системах, в которых остановка работы недопустима.

Онлайн-редактирование программ на языке ST возможно в любом режиме работы ПЛК, кроме режима «Выполнение».

В режиме имитации выполнения программы на ПК эту функцию, однако, использовать невозможно.

Запуск онлайн-редактирования

- 1,2,3...
1. Запустите мониторинг.
 2. Выберите требуемую программу на языке ST в окне дерева проекта и отобразите ее содержимое в окне справа.
 3. Выберите **Program – Online Edit – Begin (Программа – Онлайн-редактирование – Начать)**. После этого можно приступить к редактированию программы на языке ST.
 4. Внесите в программу на языке ST необходимые изменения.

Загрузка изменений

- 1,2,3...
1. Завершив редактирование программы, выберите **Program – Online Edit – Send Changes (Программа – Онлайн-редактирование – Передать изменения)**. Отобразится диалоговое окно Send Changes (Передача изменений).
 2. Выберите требуемый режим загрузки и щелкните кнопку **ОК**. Измененная программа на языке ST будет загружена в ПЛК. Информацию о режимах загрузки см. в разделе *Режимы загрузки* на стр. 218 и *Выбор режима загрузки* на стр. 219.
 3. По завершении загрузки программа на языке ST возвращается в свое прежнее состояние, в котором ее редактирование невозможно. Если вновь возникнет необходимость в редактировании, описанную выше процедуру онлайн-редактирования следует выполнить с самого начала (см. «Запуск онлайн-редактирования» выше).

Отказ от изменений

Для отказа от изменений, произведенных в программе на языке ST, выберите **Program – Online Edit – Cancel (Программа – Онлайн-редактирование – Отмена)**. Измененная программа на языке ST не будет загружена в ПЛК и вернется к своему исходному состоянию, в котором она находилась до запуска онлайн-редактирования.

Режимы загрузки

Стандартный режим

В стандартном режиме в модуль ЦПУ загружается не только объектный код программы на языке ST, но и ее исходный код. Вследствие значительного объема передаваемых данных загрузка в стандартном режиме может занимать намного больше времени, чем в быстром режиме. До полного завершения операции загрузки выполнение других операций редактирования и загрузки невозможно.


Быстрый режим

В режиме быстрой загрузки в модуль ЦПУ загружается только объектный код программы на языке ST. Исходный код (т. е. сам структурированный текст) не загружается, благодаря чему загрузка программы в быстром режиме происходит намного быстрее, чем в обычном режиме. Выполнив загрузку объектного кода в режиме быстрой загрузки, в дальнейшем следует либо 1) выбрать **Program – Transfer SFC/ST Source to PLC (Программа – Загрузить в ПЛК исходный код SFC/ST)** для загрузки исходного кода, либо 2) произвести загрузку исходного кода, следуя указаниям, которые отображаются в диалоговом окне при выходе из режима онлайн.

После загрузки объектного кода (без исходного кода) внизу окна до выхода из режима онлайн отображается индикатор желтого цвета, информирующий о том, что исходный код еще не был загружен. После загрузки исходного кода этот индикатор пропадает.

Выбор режима загрузки

Как правило, измененную программу на языке ST следует загружать в стандартном режиме, если только онлайн-редактирование не производится слишком часто. Если загрузка длится очень долго, до начала загрузки нужно до максимума повысить скорость передачи (бит/с). Если загрузка по-прежнему занимает много времени, что сильно задерживает отладку программы при непрерывном онлайн-редактировании, можно в виде исключения использовать быстрый режим, однако нужно хорошо понимать ограничения этого режима, описанные в примечании ниже (*Ограничения в режиме быстрой загрузки*).

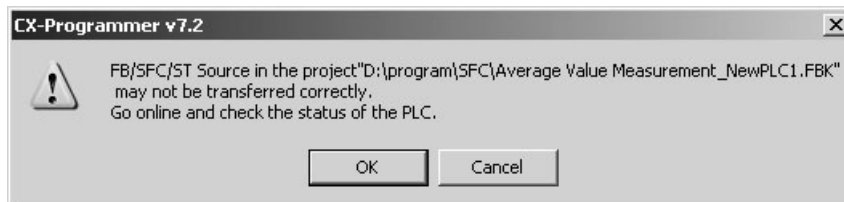
 **Предупреждение** **Ограничения в режиме быстрой загрузки (не передается исходный код ST)**
Если исходный код программы на языке ST в ПЛК не передается (в том числе если работа компьютера или CX-Programmer по той или иной причине прекращается раньше, чем завершается загрузка исходного кода), впоследствии такую программу будет невозможно корректно считать из ПЛК в CX-Programmer.

Примечание. Даже если возникнет указанная выше проблема, исходный код, возможно, удастся загрузить с помощью описанной ниже процедуры.

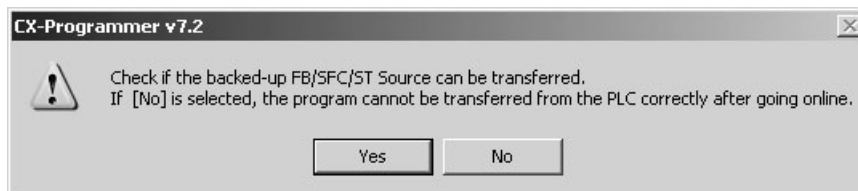
Считывание исходного кода из резервной копии проекта CX-Programmer

1,2,3...

1. Запустите программу CX-Programmer.
2. Если исходный код принадлежащей проекту программы на языке ST загружался в ПЛК в режиме быстрой загрузки и загрузка завершилась неудачей, отобразится показанное ниже диалоговое окно (уведомляющее о том, что исходный код программы на языке FB/SFC/ST, возможно, не был корректно загружен в ПЛК, и предлагающее перейти в режим онлайн с целью проверки состояния ПЛК).



3. Щелкните кнопку **ОК**. В CX-Programmer будет запущен проект, резервная копия которого была создана во время предыдущей загрузки программы в быстром режиме.
4. Установите связь с ПЛК (т. е. перейдите в режим онлайн), в который были переданы данные в режиме быстрой загрузки. Будет отображено следующее диалоговое окно. (Содержание сообщения: «Проверить возможность загрузки сохраненной копии исходного кода FB/SFC/ST? Если будет нажата кнопка «Нет», корректное считывание программы из ПЛК в режиме онлайн будет невозможно.»)



5. Щелкните кнопку **Yes (Да)**.
Если ПЛК не находится в режиме «Выполнение», будет выполнено сравнение программы проекта с программой ПЛК, и, в случае совпадения программ, исходный код программы на языке ST будет

загружен в модуль ЦПУ.

Если ПЛК находится в режиме «Выполнение», следует переключить его в другой режим работы и произвести загрузку исходного кода программы на языке ST, используя меню CX-Programmer.

Выполнение загрузки исходного кода ST вручную

- 1,2,3...
1. Запустите CX-Programmer и откройте файл проекта, содержащий исходный код программы на языке ST, который требуется загрузить в ПЛК.
 2. Установите связь с ПЛК (т. е. перейдите в режим онлайн), в который были переданы данные в режиме быстрой загрузки. В строке состояния CX-Programmer будет отображаться мигающее желтое сообщение *Src, Fail*.
 3. Выберите **Program – Online Edit – Transfer SFC/ST Source to PLC (Программа – Онлайн-редактирование – Загрузить в ПЛК исходный код SFC/ST)**. Откроется диалоговое окно загрузки исходного кода в ПЛК.
 4. Щелкните кнопку **ОК**.
Будет произведено сравнение автоматически созданной резервной копии исходного кода программы на языке ST с объектным кодом в подключенном ПЛК. Если окажется, что коды полностью соответствуют друг другу, исходный код программы на языке ST будет загружен в ПЛК.

Примечание. Перед загрузкой программы CX-Programmer, как правило, транслирует программу (исходный код языка ST) в объектный код (который может выполняться в модуле ЦПУ), после чего загружает в модуль ЦПУ и исходный код, и объектный код программы. Модуль ЦПУ хранит исходный и объектный коды программы на языке ST в памяти пользователя и во встроенной флеш-памяти. Передача и восстановление программы в CX-Programmer возможны лишь при условии наличия одновременно исходного и объектного кодов в модуле ЦПУ.

Ограничения при онлайн-редактировании программ на языке ST

В отношении редактирования программ на языке структурированного текста в режиме онлайн действуют следующие ограничения.

- Модули ЦПУ серии CJ2: количество шагов программы, которое может быть добавлено или удалено из определения функционального блока в течение одной операции онлайн-редактирования, ничем не ограничено.
- Онлайн-редактирование невозможно для программ на языке ST, объем которых превышает 4К шагов (кроме модулей ЦПУ серии CJ2).
- В течение одной операции онлайн-редактирования программа на языке ST не может быть увеличена или уменьшена более чем на 0,5 К шагов (кроме модулей ЦПУ серии CJ2).
- Закончив онлайн-редактирование, не выключайте напряжение питания ПЛК до тех пор, пока модуль ЦПУ не завершит сохранение резервной копии данных во встроенную флеш-память (т. е. пока не погаснет индикатор BKUP). Если питание будет выключено до завершения резервного копирования, резервная копия данных создана не будет и программа возвратится к состоянию, в котором она находилась до начала выполнения онлайн-редактирования.

Приложение А

Системные внешние переменные, поддерживаемые в функциональных блоках

Классификация	Название	Внешняя переменная в CX-Programmer	Тип данных	Адрес
Флаги условий	Флаг «Больше или равно» (GE)	P_GE	BOOL	CF00
	Флаг «Не равно» (NE)	P_NE	BOOL	CF001
	Флаг «Меньше или равно» (LE)	P_LE	BOOL	CF002
	Флаг ошибки выполнения команды (ER)	P_ER	BOOL	CF003
	Флаг переноса (CY)	P_CY	BOOL	CF004
	Флаг «Больше» (GT)	P_GT	BOOL	CF005
	Флаг «Равно» (EQ)	P_EQ	BOOL	CF006
	Флаг «Меньше» (LT)	P_LT	BOOL	CF007
	Флаг «Меньше нуля» (N)	P_N	BOOL	CF008
	Флаг переполнения (OF)	P_OF	BOOL	CF009
	Флаг потери значимости (UF)	P_UF	BOOL	CF010
	Флаг ошибки доступа	P_AER	BOOL	CF011
	Флаг «Всегда ВЫКЛ»	P_Off	BOOL	CF114
Флаг «Всегда ВКЛ»	P_On	BOOL	CF113	
Тактовые импульсы	Тактовые импульсы 0,02 с	P_0_02s	BOOL	CF103
	Тактовые импульсы 0,1 с	P_0_1s	BOOL	CF100
	Тактовые импульсы 0,2 с	P_0_2s	BOOL	CF101
	Тактовые импульсы 1 мин	P_1mim	BOOL	CF104
	Тактовые импульсы 1,0 с	P_1s	BOOL	CF102
Флаги и биты вспомогательной области	Флаг первого цикла	P_First_Cycle	BOOL	A200.11
	Флаг шага	P_Step	BOOL	A200.12
	Флаг первого выполнения задачи	P_First_Cycle_Task	BOOL	A200.15
	Максимальное время цикла	P_Max_Cycle_Time	UDINT	A262
	Время выполнения текущего цикла	P_Cycle_Time_Value	UDINT	A264
	Флаг ошибки времени цикла	P_Cycle_Time_Error	BOOL	A401.08
	Флаг пониженного напряжения батареи	P_Low_Battery	BOOL	A402.04
	Флаг ошибки проверки входов/выходов	P_IO_Verify_Error	BOOL	A402.09
	Бит выключения выходов	P_Output_Off_Bit	BOOL	A500.15
Слова библиотеки функц. блоков OMRON (см. примеч.)	Область CIO	P_CIO	WORD	A450
	Область HR	P_HR	WORD	A452
	Область WR	P_WR	WORD	A451
	Область DM	P_DM	WORD	A460
	Область EM0 – C	P_EM0...P_EMC	WORD	A461...A473

Примечание. Эти слова являются внешними переменными для библиотеки функциональных блоков OMRON. При создании функциональных блоков эти слова использовать не следует.

Приложение В

Ошибки программы на языке структурированного текста

Сообщения об ошибках

Сообщение об ошибке	Причина ошибки	Пример
%s' Input variables cannot be assigned a value (Присвоение значений входным переменным %s невозможно)	Входной переменной было присвоено значение.	
%s' operator not supported by %s data type (Оператором %s не поддерживается тип данных %s)	Тип данных использованного числового значения или переменной не поддерживается оператором.	A:=B+1; (*A и B являются переменными типа WORD *)
%s' variable had a read only memory AT Address and cannot be assigned a value (Невозможно присвоить значение переменной %s, так как для нее вручную назначен адрес памяти, доступный только для чтения)	Переменной, расположенной по адресу, доступному лишь для чтения (доступный только для чтения адрес вспомогательной области или флаг условия), было присвоено значение.	
Array index out of range (Индекс массива вне диапазона)	Был указан индекс массива, превышающий размер массива.	Array[100]:=10; (*Array — переменная-массив с размером 100*)
Conversion cannot convert from %s to %s (Невозможно преобразовать %s в %s)	В выражении присваивания числового значения: тип данных результата операции не соответствует типу данных целевой переменной, была подставлена переменная с отличающимся типом данных.	Y:=ABS(X); (*X — переменная типа INT, Y — переменная типа UINT*)
Division by Zero (Деление на ноль)	Числовое выражение содержит деление на 0.	
End of comment not found (Не найдено завершение комментария)	Отсутствует закрывающая скобка со звездочкой «*»), соответствующая открывающей скобке со звездочкой «(*», обозначающей комментарий.	(*текст комментария
Invalid Literal Format '%s' (Недопустимый формат литерала '%s')	Числовое значение представлено в недопустимом формате.	X:=123_; (*Отсутствует число после знака подчеркивания*) X:=1__23; (*Введено два знака подчеркивания подряд*) X:=2#301; Y:=8#90; (*Для двоичного и восьмеричного формата введены недопустимые числовые значения*) Примечание. Для более удобного зрительного восприятия числовых значений допускается размещать между ними знак подчеркивания. Префикс 2#, 8# и 16# впереди числового значения обозначает, что это числовое значение имеет, соответственно, двоичный, восьмеричный и шестнадцатеричный формат.

Сообщение об ошибке	Причина ошибки	Пример
Invalid Literal Value (Недопустимое значение литерала)	Использовано недопустимое числовое значение.	X:=1e2; (*Использован показатель степени для числового значения, не являющегося вещественным (REAL)*) Примечание. Буква «e» обозначает показатель степени с основанием 10.
Invalid array index (Недопустимый индекс массива)	В качестве индекса массива указана переменная нецелого типа или числовое выражение, дающее результат нецелого типа.	Array[Index]:=10; (*Index — переменная типа WORD)
Invalid constant (Недопустимая константа)	В качестве целочисленного переключателя в выражении CASE указана переменная или число нецелого типа либо числовое выражение, дающее результат нецелого типа.	CASE A OF (*A — переменная типа REAL*) 1: X:=1; 2: X:=2; END_CASE;
Invalid expression (Недопустимое числовое выражение)	Использовано недопустимое числовое выражение (например, в качестве целочисленного операнда или условия в выражении IF, WHILE, REPEAT, FOR, CASE).	WHILE DO (*В выражении WHILE отсутствует выражение-условие*) X:=X+1; END_WHILE;
Invalid parameter in FOR loop declaration (Недопустимый параметр в объявлении цикла FOR)	В выражении FOR использована переменная, не относящаяся к типу данных INT, DINT, LINT, UINT, UDINT или ULINT.	FOR I:=1 TO 100 DO (*I — переменная типа WORD*) X:=X+1; END_FOR;
Invalid statement (Неверный синтаксис выражения)	Выражение имеет недопустимый синтаксис. Например, в выражении IF, WHILE, REPEAT, FOR, CASE, REPEAT отсутствует соответствующее ключевое слово IF, WHILE, REPEAT, FOR, CASE или REPEAT.	X:=X+1; (*В выражении отсутствует ключевое слово REPEAT*) UNTIL X>10 END_REPEAT;
Invalid variable for Function output (Недопустимая переменная для выхода функции)	Для выхода функции указана недопустимая переменная (в качестве выходной переменной ENO указана переменная числового или небулевого (BOOL) типа).	Y:=SIN(X1, ENO=>1);
Missing (Отсутствует «(»)	В выражении вызова функции или команды преобразования формата данных отсутствует символ «(» (открывающая скобка).	Y:=INT_TO_DINT X);
Missing) (Отсутствует «)»)	В синтаксисе оператора либо в выражении вызова функции или команды преобразования формата данных отсутствует знак «)» (закрывающая скобка), соответствующий знаку «(» (открывающая скобка).	Y:=(X1+X2/2
Missing : (Отсутствует «:»)	После целочисленного варианта значения в выражении CASE отсутствует знак «:» (двоеточие).	CASE A OF 1 X:=1; END_CASE;
Missing := (Отсутствует «:=»)	В выражении присваивания отсутствует знак «:=».	
Missing ; (Отсутствует «;»)	Выражение не завершается знаком «;» (точка с запятой).	

Сообщение об ошибке	Причина ошибки	Пример
Missing DO (Отсутствует DO)	В выражении FOR или WHILE отсутствует ключевое слово «DO».	
Missing END_CASE (Отсутствует END_CASE)	В конце выражения CASE отсутствует выражение «END_CASE».	
Missing END_FOR (Отсутствует END_FOR)	В конце выражения FOR отсутствует ключевое слово «END_FOR».	
Missing END_IF (Отсутствует END_IF)	В конце выражения IF отсутствует ключевое слово «END_IF».	
Missing END_REPEAT (Отсутствует END_REPEAT)	В конце выражения REPEAT отсутствует ключевое слово «END_REPEAT».	
Missing END_WHILE (Отсутствует END_WHILE)	В конце выражения WHILE отсутствует ключевое слово «END_WHILE».	
Missing Input Parameter. All input variables must be set. (Отсутствует входной параметр. Должны быть заданы все входные переменные.)	Для функции не указан или не полностью определен аргумент.	Y:=EXPT(X);
Missing OF (Отсутствует OF)	В выражении CASE отсутствует ключевое слово «OF».	
Missing THEN (Отсутствует THEN)	В выражении IF отсутствует ключевое слово «THEN».	
Missing TO (Отсутствует TO)	В выражении FOR отсутствует ключевое слово «TO».	
Missing UNTIL (Отсутствует UNTIL)	В выражении REPEAT отсутствует ключевое слово «UNTIL».	
Missing [(Отсутствует «[«)	Не был указан индекс массива для переменной-массива.	X:=Array; (*Array — переменная-массив*)
Missing] (Отсутствует «]»)	Не был указан индекс массива для переменной-массива.	X:=Array[2; (*Array — переменная-массив*)
Missing constant (Отсутствует константа)	В строке варианта значения в выражении CASE отсутствует целочисленная константа.	CASE A OF 2..: X:=1; 2,: X:=2; END_CASE;
NOT operation not supported on a literal number (Операция NOT не может быть выполнена над числовым литералом)	Оператор NOT был использован для числового значения.	Result:=NOT 1;
Negation not supported by %s data type (Отрицание не поддерживается для типа данных %s)	Был использован знак минус для переменных, чей тип данных не допускает отрицательных значений (UINT, UDINT, ULINT).	Y:=-X; (*X — переменная типа UINT, Y — переменная типа INT*)
There must be one line of valid code (excluding comments) (Нет ни одной строки с допустимым исполняемым кодом (не включая комментарии))	Текст программы не содержит ни одной строки с допустимым исполняемым кодом (не включая комментарии).	
Too many variables specified for Function (Избыточное число переменных в функции)	Для функции указано избыточное число параметров.	Y:=SIN(X1,X2);

Сообщение об ошибке	Причина ошибки	Пример
Undefined identifier '%s' (Неопределенный идентификатор '%s')	Была использована переменная, которая не была определена в таблице переменных.	
Unexpected syntax '%s' (Непредусмотренный синтаксис '%s')	Ключевое (зарезервированное) слово или переменная были использованы недопустимым образом.	FOR I:=1 TO 100 DO BY -1 (*Недопустимое расположение ключевого слова DO*) X:=X+1; END_FOR;
Usage mismatch in Function variable (Несоответствующее применение переменной функции)	Параметр функции был использован недопустимым образом.	Y:=SIN(X1,EN=>BOOL1); (*Входной параметр EN был использован в качестве выходного параметра*)
Value out of range (Значение вне диапазона)	Переменной присвоено значение, лежащее за пределами области допустимых значений, соответствующей типу данных этой переменной.	X:=32768; (*X — переменная типа INT*)
Variable '%s' is not a Function parameter (Переменная '%s' не является параметром функции)	В параметре указана переменная, которую не допускается указывать в качестве параметра функции.	Y:=SIN(Z:=X); (*X и Y — переменные типа REAL, а Z не является параметром функции SIN *)

Предупреждающие сообщения

Предупреждающее сообщение	Причина предупреждения	Пример
Keyword '%s' is redundant (Ключевое слово '%s' является лишним)	Ключевое слово было использовано в недопустимом месте. Например, выражение EXIT было использовано за пределами синтаксиса цикла.	
Conversion from '%s' to '%s', possible loss of data (Преобразование '%s' в '%s' может привести к потере данных)	Преобразование значения с типом данных большей размерности в значение с типом данных меньшей размерности может привести к потере данных.	Y:=DINT_TO_INT(X); (*X — переменная типа DINT, Y — переменная типа INT*)

Приложение С

Описание функций

Стандартные функции

Функции для работы с текстовыми строками

LEN: определение длины строки

- Функция
Определение длины указанной текстовой строки.
- Применение
 $Return_value := LEN(string);$
- Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
string	STRING	Указывает текстовую строку, длину которой требуется узнать.
Return_value	INT	Возвращает размер указанной текстовой строки.

- Пример

Переменные
STRING Message
INT Result

Message

A	B	C	D	E	F	G	H
---	---	---	---	---	---	---	---

Result:=LEN(Message);
→ В переменную **Result** записывается «8».

LEFT: извлечение символов слева

- Функция
Извлечение указанного количества символов, расположенных в начале (слева) указанной текстовой строки.
- Применение
 $Return_value := LEFT(Source_string, Number_of_characters);$
- Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Source_string	STRING	Указывает текстовую строку для извлечения символов.
Number_of_characters	INT, UINT	Указывает количество извлекаемых символов.
Return_value	STRING	Возвращает извлеченные символы.

- Пример

Переменные
STRING Message
STRING Result

Message

A	B	C	D	E	F	G	H
---	---	---	---	---	---	---	---

Result:=LEFT(Message,3);
→ В переменную **Result** записывается «ABC»

RIGHT: извлечение символов справа

- Функция
Извлечение указанного количества символов, расположенных в конце (справа) указанной текстовой строки.
- Применение
 $Return_value := RIGHT(Source_string, Number_of_characters);$

- Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Source_string	STRING	Указывает текстовую строку для извлечения символов.
Number_of_characters	INT, UINT	Указывает количество извлекаемых символов.
Return_value	STRING	Возвращает извлеченные символы.

- Пример

Переменные
STRING Message
STRING Result



Result:=RIGHT(Message,3);
→ В переменную **Result** записывается «FGH».

MID: извлечение символов внутри строки

- Функция

Извлечение указанного количества символов, начиная с указанной позиции, в пределах указанной текстовой строки.

- Применение

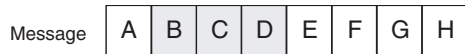
Return_value := MID (Source_string, Number_of_characters, Position);

- Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Source_string	STRING	Указывает текстовую строку для извлечения символов.
Number_of_characters	INT, UINT	Указывает количество извлекаемых символов.
Position	INT, UINT	Указывает позицию, в которой должно начаться извлечение символов. Первому символу соответствует позиция 1 (напр., в сообщении на рисунке ниже в позиции 1 находится символ «А»).
Return_value	STRING	Возвращает извлеченные символы.

- Пример

Переменные
STRING Message
STRING Result



Result:=MID(Message,3,2);
→ В переменную **Result** записывается «BCD».

CONCAT: конкатенация строк

- Функция

Объединение указанных текстовых строк.
Может быть указано до 31 текстовых строк.

- Применение

Return_value := CONCAT(Source_string_1, Source_string_2, ...);

- Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Source_string_1	STRING	Указывает присоединяемую текстовую строку.
Source_string_2	STRING	Указывает присоединяемую текстовую строку.
:		
Return_value	STRING	Возвращает текстовую строку, полученную в результате объединения указанных текстовых строк.

• Пример

Переменные
 STRING Message1
 STRING Message2
 STRING Message3
 STRING Result

Message 1

A	B	C
---	---	---

 Message 2

D	E
---	---

 Message 3

F	G	H
---	---	---

Result:=CONCAT(Message1,Message2,Message3);
 → В переменную **Result** записывается «ABCDEFGH».

Result

A	B	C	D	E	F	G	H
---	---	---	---	---	---	---	---

INSERT: вставка символов

• Функция

Вставка указанных символов в текстовую строку.

• Применение

Return_value := INSERT(Source_string, Insert_string, Position);

• Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Source_string	STRING	Указывает текстовую строку, в которую должны быть вставлены символы.
Insert_string	STRING	Указывает вставляемую строку.
Position	INT, UINT	Указывает позицию, в которой должны быть вставлены символы. Первому символу соответствует позиция 1 (напр., в сообщении на рисунке ниже в позиции 1 находится символ «А»).
Return_value	STRING	Возвращает текстовую строку, полученную в результате вставки символов.

• Пример

Переменные
 STRING Message1
 STRING Message2
 STRING Result

Message 1

A	B	C	D
---	---	---	---

 Message 2

E	F	G	H
---	---	---	---

Result := INSERT(Message1, Message2, 2);
 → В переменную **Result** записывается «ABEFGHC».

Result

A	B	E	F	G	H	C	D
---	---	---	---	---	---	---	---

DELETE: удаление символов

• Функция

Удаление указанного количества символов из указанной текстовой строки, начиная с указанной позиции.

• Применение

Return_value := DEL (Source_string, Number_of_characters, Position);

• Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Source_string	STRING	Указывает текстовую строку, из которой должны быть удалены символы.
Number_of_characters	INT, UINT	Указывает количество удаляемых символов.
Position	INT, UINT	Указывает позицию, в которой должны быть удалены символы. Первому символу соответствует позиция 1 (напр., в сообщении на рисунке ниже в позиции 1 находится символ «А»).
Return_value	STRING	Возвращает текстовую строку, полученную в результате удаления указанного количества символов.

• Пример

Переменные
STRING Message1
STRING Result



Result:=DEL(Message1,4,2);
→ В переменную **Result** записывается «AFGH».



REPLACE: замена символов

• Функция

Замещение указанного количества символов указанной текстовой строки, начиная с указанной позиции.

• Применение

Return_value := REPLACE(*Source_string*, *Replace_string*, *Number_of_characters*, *Position*);

• Аргументы и возвращаемые значения

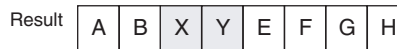
Имя переменной	Тип данных	Описание
Source_string	STRING	Указывает текстовую строку, в которой должны быть заменены символы.
Replace_string	STRING	Указывает замещающую текстовую строку.
Number_of_characters	INT, UINT	Указывает количество заменяемых символов.
Position	INT, UINT	Указывает позицию, в которой должна быть выполнена замена символов. Первому символу соответствует позиция 1 (напр., в сообщении на рисунке ниже в позиции 1 находится символ «А»).
Return_value	STRING	Возвращает текстовую строку, полученную в результате замены символов.

• Пример

Переменные
STRING Message1
STRING Message2
STRING Result



Result:=REPLACE(Message1,Message2,2,3);
→ В переменную **Result** записывается «ABXYEFGH».



FIND: поиск символов

• Функция

Поиск указанной текстовой строки в другой текстовой строке и возвращение позиции первого вхождения.

В случае необнаружения искомой строки возвращается 0.

• Применение

Return_value := FIND(*Source_string*, *Find_string*);

• Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Source_string	STRING	Указывает текстовую строку, в которой производится поиск.
Find_string	STRING	Указывает искомую текстовую строку.
Return_value	INT	Возвращает позицию первого вхождения искомой текстовой строки. Первому символу соответствует позиция 1 (напр., в сообщении на рисунке ниже в позиции 1 находится символ «А»).

• Пример

Переменные
 STRING Message1
 STRING Message2
 INT Result



Result:=FIND(Message1,Message2);
 → В переменную **Result** записывается «2».

Функции сдвига данных

SHL: битовый сдвиг влево

• Функция

Сдвигает строку битов влево на n битов.

В освобождающиеся в результате сдвига разряды в правой части строки записываются нули.

• Применение

Return_value := SHL (Shift_target_data, Number_of_bits);

• Аргументы и возвращаемые значения

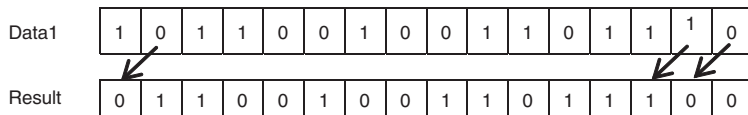
Имя переменной	Тип данных	Описание
Shift_target_data (S1)	BOOL, WORD, DWORD, LWORD	Указывает сдвигаемые данные.
Number_of_bits (n)	INT, UINT, UDINT, ULINT, DINT, LINT	Указывает количество битов, на которое должна быть сдвинута строка битов.
Return_value	BOOL, WORD, DWORD, LWORD	Возвращает выходное значение.

• Примечание

Для 1-го аргумента и возвращаемого значения должен быть выбран один и тот же тип данных.

• Пример

[Переменные]
 WORD Data1
 INT N
 WORD Result
 Data1 = B26E (hex) → 1011 0010 0110 1110 (двоичн.)
 N = 1 (десятичн.)



Result := SHL(Data1,N);
 → В переменную **Result** записывается «64DC».

SHR: битовый сдвиг вправо

• Функция

Сдвигает строку битов вправо на n битов.

В освобождающиеся в результате сдвига разряды в левой части строки записываются нули.

• Применение

Return_value := SHR (Shift_target_data, Number_of_bits);

• Аргументы и возвращаемые значения

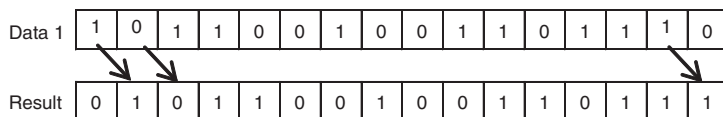
Имя переменной	Тип данных	Описание
Shift_target_data (S1)	BOOL, WORD, DWORD, LWORD	Указывает сдвигаемые данные.
Number_of_bits (n)	INT, UINT, UDINT, ULINT, DINT, LINT	Указывает количество битов, на которое должна быть сдвинута строка битов.
Return_value	BOOL, WORD, DWORD, LWORD	Возвращает выходное значение.

• Примечание

Для 1-го аргумента и возвращаемого значения должен быть выбран один и тот же тип данных.

• Пример

[Переменные]
 WORD Data1 Data1 = B26E (hex) → 1011 0010 0110 1110 (двоичн.)
 INT N N = 1 (десятичн.)
 WORD Result



Result := SHR(Data1,N);
 → В переменную **Result** записывается «5937».

ROL: битовый циклический сдвиг влево

• Функция

Циклический сдвиг строки битов на n битов влево.

• Применение

Return_value := ROL (Rotation_target_data, Number_of_bits);

• Аргументы и возвращаемые значения

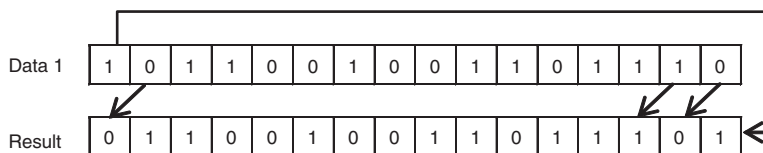
Имя переменной	Тип данных	Описание
Rotation_target_data (S1)	BOOL, WORD, DWORD, LWORD	Указывает циклически сдвигаемые данные.
Number_of_bits (n)	INT, UINT, UDINT, ULINT, DINT, LINT	Указывает количество битов, на которое должна быть циклически сдвинута строка битов.
Return_value	BOOL, WORD, DWORD, LWORD	Возвращает выходное значение.

• Примечание

Для 1-го аргумента и возвращаемого значения должен быть выбран один и тот же тип данных.

• Пример

[Переменные]
 WORD Data1 Data1 = B26E (hex) → 1011 0010 0110 1110 (двоичн.)
 INT N N = 1 (десятичн.)
 WORD Result



Result := ROL(Data1,N);
 → В переменную **Result** записывается «64DD».

ROR: битовый циклический сдвиг вправо

• Функция

Циклический сдвиг строки битов на n битов вправо.

• Применение

Return_value := ROR (Rotation_target_data, Number_of_bits);

• Аргументы и возвращаемые значения

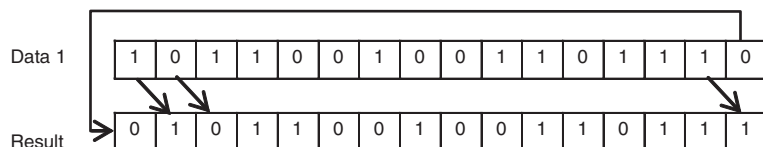
Имя переменной	Тип данных	Описание
Rotation_target_data (S1)	BOOL, WORD, DWORD, LWORD	Указывает циклически сдвигаемые данные.
Number_of_bits (n)	INT, UINT, UDINT, ULINT, DINT, LINT	Указывает количество битов, на которое должна быть циклически сдвинута строка битов.
Return_value	BOOL, WORD, DWORD, LWORD	Возвращает выходное значение.

• Примечание

Для 1-го аргумента и возвращаемого значения должен быть выбран один и тот же тип данных.

• Пример

[Переменные]
 WORD Data1 Data1 = B26E (hex) → 1011 0010 0110 1110 (двоичн.)
 INT N N = 1 (десятичн.)
 WORD Result



Result := ROR(Data1,N);
 → В переменную **Result** записывается «5937».

Функции управления данными

LIMIT: ограничение сверху/снизу

• Функция

Определение принадлежности входного значения диапазону, заданному верхним и нижним предельными значениями, и возврат соответствующего значения.

• Применение

Return_value := LIMIT (Lower_limit_data, Input_data, Upper_limit_data);

• Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Lower_limit_data	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает нижнее предельное значение.
Input_data	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает входное значение.
Upper_limit_data	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает верхнее предельное значение.
Return_value	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Возвращает выходное значение.

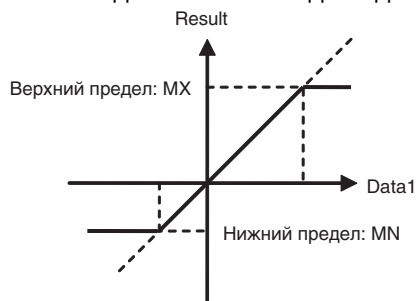
• Примечание

Если входное значение меньше, чем нижнее предельное значение, в качестве результата возвращается нижнее предельное значение.

Если входное значение больше, чем верхнее предельное значение, в качестве результата возвращается верхнее предельное значение.

Если входное значение находится между верхним и нижним предельными значениями, в качестве результата возвращается само входное значение.

Для аргументов и возвращаемого значения должен быть задан один и тот же тип данных.



- Пример

```
[Переменные]
INT MN      MN = 123 Data1 = 456 MX = 789
INT Data1
INT MX      Result := LIMIT(MN,Data1,MX);
INT Result  → В переменную Result записывается «456».
```

Функции выбора данных

SEL: выбор значения

- Функция

Выбор одного из двух значений на основании состояния условия выбора.

- Применение

```
Return_value := SEL (Selection_condition, Selection_target_data1, Selection_target_data2);
```

- Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Selection_condition (g)	BOOL	Если условие выбора g = ИСТИНА («1»), выбирается значение S2. Если условие выбора g = ЛОЖЬ («0»), выбирается значение S1.
Selection_target_data1 (S1)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из возможных значений для выбора.
Selection_target_data2 (S2)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из возможных значений для выбора.
Return_value	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Возвращает выходное значение.

- Примечание

Для обоих аргументов Selection_target_data должен быть выбран один и тот же тип данных.

- Пример

```
[Переменные]
BOOL G
INT Data1
INT Data2
INT Result

G = 1
Data1 = 123 Data2 = 456

G =
  0   1
  ┌───┬───┐
  │ 123 │ 456 │
  └───┴───┘
    ↓
  Result

Result := SEL(G,Data1,Data2);
→ В переменную Result записывается «456».
```

MUX: мультиплексор

- Функция

Выбор одного из нескольких значений в соответствии со значением условия выбора. Может быть указано до 30 возможных значений.

- Применение

```
Return_value := MUX (Extraction_condition, Extraction_target_data1, Extraction_target_data2, ...);
```

• Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Extraction_condition (n)	INT, UINT, UDINT, ULINT, DINT, LINT	Указывает, какое значение должно быть выбрано. Условие выбора $n = 0, 1, 2, \dots, 29$ Если $n = 0$, возвращается заданное значение S1. Если $n = 1$, возвращается заданное значение S2. : : Если $n = n$, возвращается заданное значение S(n+1).
Extraction_target_data1 (S1)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из возможных значений для извлечения.
Extraction_target_data2 (S2)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из возможных значений для извлечения.
:		
Return_value	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Возвращает выходное значение.

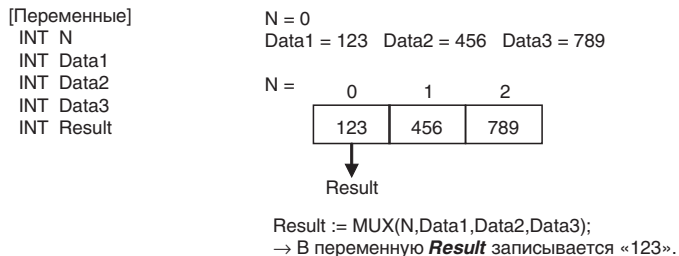
• Примечание

Return_value := MUX(Selection_condition, Extraction_target_data1, Extraction_target_data2, ...);

Для возвращаемого значения и всех аргументов, за исключением первого (Extraction_condition), должен быть задан один и тот же тип данных.

Если в условии выбора вводится значение вне диапазона 0...29, функция возвращает неопределенное значение.

• Пример



MAX: максимальное значение

• Функция

Выбор максимального значения среди нескольких заданных значений.
Может быть задано до 31 возможных значений.

• Применение

Return_value := MAX (Target_data1, Target_data2, Target_data3, ..., Target_data31);

• Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Target_data1 (S1)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из значений.
Target_data2 (S2)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из значений.
Target_data3 (S3)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из значений.
:		

Имя переменной	Тип данных	Описание
Target_data31 (S31)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из значений.
Return_value	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Возвращает выходное значение.

• Примечание

$Return_value := MAX(Target_data1, Target_data2, Target_data3, \dots, Target_data31);$

Для всех аргументов и возвращаемого значения должен быть задан один и тот же тип данных.

• Пример

[Переменные]

INT Data1
INT Data2
INT Data3
INT Result

Data1 = 123 Data2 = 456 Data3 = 789



Result := MAX(Data1,Data2,Data3);

→ В переменную **Result** записывается «789».

MIN: минимальное значение

• Функция

Выбор минимального значения среди нескольких заданных значений.

Может быть задано до 31 возможных значений.

• Применение

$Return_value := MIN(Target_data1, Target_data2, Target_data3, \dots, Target_data31);$

• Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Target_data1 (S1)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из значений.
Target_data2 (S2)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из значений.
Target_data3 (S3)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из значений.
:		
Target_data31 (S31)	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Указывает одно из значений.
Return_value	BOOL, INT, UINT, UDINT, ULINT, DINT, LINT, WORD, DWORD, LWORD, REAL, LREAL	Возвращает выходное значение.

• Примечание

$Return_value := MIN(Target_data1, Target_data2, Target_data3, \dots, Target_data31);$

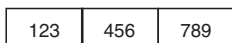
Для всех аргументов и возвращаемого значения должен быть задан один и тот же тип данных.

• Пример

[Переменные]

INT Data1
INT Data2
INT Data3
INT Result

Data1 = 123 Data2 = 456 Data3 = 789



Result := MIN(Data1,Data2,Data3);

→ В переменную **Result** записывается «123».

Расширенные функции OMRON

Функции для работы с картой памяти

WRITE_TEXT: создание текстового файла

- Функция
Запись указанной текстовой строки в указанный файл на карте памяти.
- Применение
Write_Text (Write_string, Directory_name_and_file_name, Delimiter, Parameter);
- Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Write_string	STRING	Указывает текстовую строку, которая должна быть записана в файл.
Directory_name_and_file_name	STRING	Указывает имя каталога и имя файла, включая корневой каталог (\). Длина имени файла не должна превышать 8 символов. К имени файла всегда добавляется расширение TXT. К примеру, при значении переменной «/LINE_A» в корневом каталоге будет создан файл с именем LINE_A.TXT.
Delimiter	STRING	"": пустой символ «,»: запятая «\$L» или «\$I»: перевод строки (ASCII 0A) «\$N» или «\$n»: возврат каретки + перевод строки (ASCII 0D 0A) «\$P» или «\$p»: новая страница (ASCII 0C) «\$R» или «\$r»: возврат каретки (ASCII 0D) «\$T» или «\$t»: символ табуляции (ASCII 09)
Параметр	INT, UINT, WORD	0: присоединить; 1: создать новый файл.

• Пример

Переменные		
BOOL	P_MemCardBusyFlag	(* Флаг операции над памятью файлов *) AT A343.13
BOOL	P_MemCardAskFlag	(* Флаг обнаружения карты памяти *) AT A343.15
STRING	FileName	(* Имя файла *)
INT	LogData1 2 3	(* Номер журнала *)
STRING	FiledStr1 2 3	(* Текстовая строка номера журнала *)
STRING	CsvLineStr	(* Журнал в формате CSV, 1–строчная текстовая строка *)

```

FileName := '\LOGFILE';
LogData1 := 12;
LogData2 := 345;
LogData3 := 6789;

(* Вывод данных в текстовый файл, *)
(* если соблюдаются условия записи на карту памяти. *)
IF ( P_MemCardAckFlag AND (NOT P_MemCardBusyFlag) ) THEN

(* Преобразование из числового формата в формат текстовой строки *)
FieldStr1 := INT_TO_STRING( LogData1 );
FieldStr2 := INT_TO_STRING( LogData2 );
FieldStr3 := INT_TO_STRING( LogData3 );
(* Создание 1–строчной текстовой строки в формате CSV, *)
(* содержащей числовое значение *)
CsvLineStr := FieldStr1 + ',' + FieldStr2 + ',' + FieldStr3;
(* Вывод одной строки числовых данных в файл *)
WRITE_TEXT( CsvLineStr, FileName, '$n', 0 );
END_IF;
    
```

Содержание выходного файла

LOGFILE.TXT

```

12,345,6789
    
```

Сопутствующий флаг вспомогательной области	Адрес	Описание
Флаг операции над памятью файлов	A343.13	Включен во время выполнения любой из следующих операций: • команда CMND передает команду FINS локальному модулю ЦПУ; • выполняется команда, оперирующая памятью файлов; • выполняется замена программы с помощью управляющего бита во вспомогательной области; • выполняется операция простого резервного копирования.
Флаг обнаружения карты памяти	A343.15	Включен, если модулем ЦПУ обнаружена карта памяти.

Дополнительные сведения и меры предосторожности относительно сопутствующих флагов вспомогательной области см. в описании команды для работы с памятью файлов FWRIT в Справочном руководстве по командам программирования ПЛК серии CS/CJ.

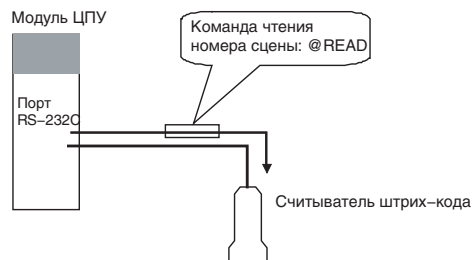
ФУНКЦИИ СВЯЗИ

TXD_CPU: передача строки через порт RS-232C модуля ЦПУ

- Функция
 Передача текстовой строки через порт RS-232C модуля ЦПУ.
- Применение
 TXD_CPU (Send_string);
- Условия
 Для порта RS-232C должен быть выбран режим беспротokolной связи.
- Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Send_string	STRING	Указывает передаваемую текстовую строку.

• Пример



Переменные

BOOL DoSendData (* Переменная для управления функцией передачи *)

INT iProcess (* Номер процедуры *)

STRING Message (* Сообщение передачи *)

BOOL SendEnableCPUPort (* Флаг готовности к передаче *) AT A392.05

```
(* Данные передаются, если DoSendData = 1, а iProcess = 0 *)
IF (DoSendData = TRUE) AND (iProcess = 0) THEN
    iProcess := 1;
    DoSendData := FALSE;
END_IF;
(* Выполнение процедуры передачи с указанным номером процедуры *)
CASE iProcess OF
    1: (* Формирование передаваемых текстовых данных *)
        Message := '@READ';
        iProcess := 2;
    2: (* Выполнение функции передачи, если передача разрешена *)
        IF SendEnableCPUPort = TRUE THEN
            TXD_CPU(Message);
            iProcess := 3;
        END_IF;
    3: (* Если флаг готовности к передаче = 1, передача завершена *)
        IF SendEnableCPUPort = TRUE THEN
            iProcess := 0;
        END_IF;
END_CASE;
```

Сопутствующий флаг вспомогательной области	Адрес	Описание
Флаг готовности к передаче порта RS-232C	A392.05	Включен, если есть возможность передачи данных в беспроточольном режиме.

Дополнительные сведения и меры предосторожности относительно сопутствующих флагов вспомогательной области см. в описании команды связи по последовательному интерфейсу TXD в Справочном руководстве по командам программирования ПЛК серии CS/CJ.

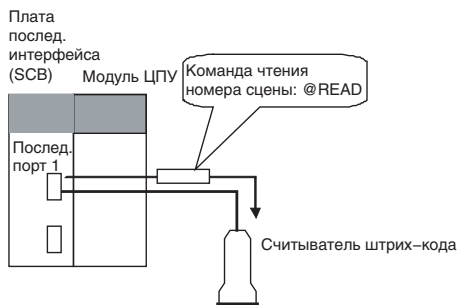
TXD_SCB: передача строки через последовательный порт платы последовательного интерфейса

- Функция
 - Передача строки через последовательный порт платы последовательного интерфейса (SCB).
- Применение
 - TXD_SCB (*Send_string, Serial_port*);
- Условия
 - Для последовательного порта должен быть выбран режим беспроточольной связи.

• Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Send_string	STRING	Указывает передаваемую текстовую строку.
Serial_port	INT, UINT, WORD	Указывает номер последовательного порта. 1: последовательный порт 1; 2: последовательный порт 2.

• Пример



Переменные		
BOOL	P_DoSendData	(* Переменная для управления функцией передачи *)
INT	iProcess	(* Номер процедуры *)
STRING	Message	(* Сообщение передачи *)
BOOL	P_SendEnableSCBPort1	(* Флаг готовности к передаче *) AT A356.05 Используется последовательный порт 1.

```
(* Используется послед. порт номер 1 *)

(* Данные передаются, если P_DoSendData = 1, а iProcess = 0 *)
IF (P_DoSendData = TRUE) AND (iProcess = 0) THEN
    iProcess := 1;
    P_DoSendData := FALSE;
END_IF;

(* Выполнение процедуры передачи с указанным номером процедуры *)
CASE iProcess OF
    1: (* Формирование передаваемых текстовых данных *)
        Message := '@READ';
        iProcess := 2;
    2: (* Выполнение функции передачи, если передача разрешена *)
        IF P_SendEnableSCBPort1 = TRUE THEN
            TXD_SCB(Message, 1);
            iProcess := 3;
        END_IF;
    3: (* Если флаг готовности к передаче = 1, передача завершена *)
        IF P_SendEnableSCBPort1 = TRUE THEN
            iProcess := 0;
        END_IF;
END_CASE;
```

Сопутствующий флаг вспомогательной области	Адрес	Описание
Флаг готовности к передаче порта 1	A356.05	Включен, если есть возможность передачи данных в беспроточном режиме.
Флаг готовности к передаче порта 2	A356.13	Включен, если есть возможность передачи данных в беспроточном режиме.

Дополнительные сведения и меры предосторожности относительно сопутствующих флагов вспомогательной области см. в описании команды связи по последовательному интерфейсу TXD в Справочном руководстве по командам программирования ПЛК серии CS/CJ.

TXD_SCU: передача строки через последовательный порт модуля последовательного интерфейса

- **Функция**
Передача текстовой строки через последовательный порт модуля последовательного интерфейса (SCU).
- **Применение**
TXD_SCU (Send_string, SCU_unit_number, Serial_port, Internal_logic_port);
- **Условия**
Для последовательного порта должен быть выбран режим беспrotocolной связи.
- **Аргументы и возвращаемые значения**

Имя переменной	Тип данных	Описание
Send_string	STRING	Указывает передаваемую текстовую строку.
SCU_unit_number	INT, UINT, WORD	Указывает номер модуля последовательного интерфейса.
Serial_port	INT, UINT, WORD	1: последовательный порт 1; 2: последовательный порт 2.
Internal_logic_port	INT, UINT, WORD	0...7: указан номер внутреннего логического порта. 16#F: автоматическое назначение внутреннего логического порта.

• **Пример**



Переменные		
BOOL	P_DoSendData	(* Переменная для управления функцией передачи *)
INT	iProcess	(* Номер процедуры *)
STRING	Message	(* Сообщение передачи *)
BOOL	P_TXDU_Exe	(* Флаг выполнения TXDU *) AT 1519.05 Номер модуля 0, используется последовательный порт 2.
BOOL	P_ComInstEnable	(* Флаг доступности порта связи*) AT A202.07 Используется порт 7.

```
(* Использовать следующие параметры: номер модуля: 0; номер послед. порта: 2; номер логического порта: 7 *)

(* Данные передаются, если P_DoSendData = 1, а iProcess = 0 *)
IF (P_DoSendData = TRUE) AND (iProcess = 0) THEN
    iProcess := 1;
    P_DoSendData := FALSE;
END_IF;

(* Выполнение процедуры передачи с указанным номером процедуры *)
CASE iProcess OF
    1: (* Формирование передаваемых текстовых данных *)
        Message := '@READ';
        iProcess := 2;
    2: (* Функция передачи выполняется, если установлены флаг доступности порта связи
        и флаг выполнения TXDU *)
        IF (P_ComInstEnable = TRUE) AND (P_TXDU_Exe = FALSE) THEN
            TXD_SCU(Message, 0, 2, 7);
            iProcess := 3;
        END_IF;
    3: (* Передача считается завершенной, если установлен флаг доступности порта связи *)
        IF P_ComInstEnable = TRUE THEN
            iProcess := 0;
        END_IF;
END_CASE;
```

Сопутствующий флаг вспомогательной области	Адрес	
Флаги доступности портов связи	A202.00...A202.07	Включен, если есть возможность обмена данными по сети. Номера битов соответствуют непосредственно номерам внутренних логических портов. Биты 00...07: внутренние логические порты 0...7

Сопутствующие биты области модулей шины ЦПУ	Бит	
n = CIO 150 + 25 x номер модуля Порт 1: n+9 Порт 2: n+19	05	Включен во время выполнения команды TXDU.

Дополнительные сведения и меры предосторожности относительно сопутствующих флагов вспомогательной области см. в описании команды связи по последовательному интерфейсу TXDU в Справочном руководстве по командам программирования ПЛК серии CS/CJ.

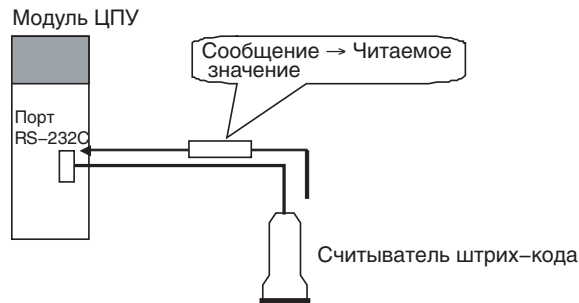
RXD_CPU: прием строки через порт RS-232C модуля ЦПУ

- Функция
 - Прием текстовой строки через порт RS-232C модуля ЦПУ.
- Применение
 - RXD_CPU (*Storage_location, Number_of_characters*)
- Условия
 - Для порта RS-232C должен быть выбран режим беспротokolной связи.

• Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Storage_location	STRING	Указывает переменную для записи принятой текстовой строки.
Number_of_characters	INT, UINT, WORD	Указывает количество принимаемых символов. 0...255

• Пример



Переменные		
BOOL	P_DoRecvData	(* Переменная для управления функцией приема *)
STRING	Message	(* Переменная для сохранения принятого сообщения *)
BOOL	P_EndRecvCPUPort	(* Флаг завершения приема *) AT A392.06

(* Данные принимаются, если P_DoRecvData = 1 и завершен прием*)

```
IF (P_DoRecvData = TRUE) AND (P_EndRecvCPUPort = TRUE) THEN
  (* Получить 16 символов *)
  RXD_CPU(Message, 16);
  P_DoRecvData := FALSE;
END_IF;
```

Сопутствующий флаг вспомогательной области	Адрес	Описание
Флаг завершения приема порта RS-232C	A392.06	Включен, если в беспrotocolном режиме завершился прием данных.
Флаг переполнения при приеме порта RS-232C	A392.07	Включен, если во время приема данных в беспrotocolном режиме произошло переполнение буфера приема.
Счетчик приема порта RS-232C	A393	Содержит значение количества символов, принятых в беспrotocolном режиме.

Дополнительные сведения и меры предосторожности относительно сопутствующих флагов вспомогательной области см. в описании команды связи по последовательному интерфейсу RXD в Справочном руководстве по командам программирования ПЛК серии CS/CJ.

RXD_SCB: прием строки через последовательный порт платы последовательного интерфейса

• Функция

Прием текстовой строки через последовательный порт платы последовательного интерфейса (SCB).

• Применение

RXD_SCB (Storage_location, Number_of_characters, Serial_port)

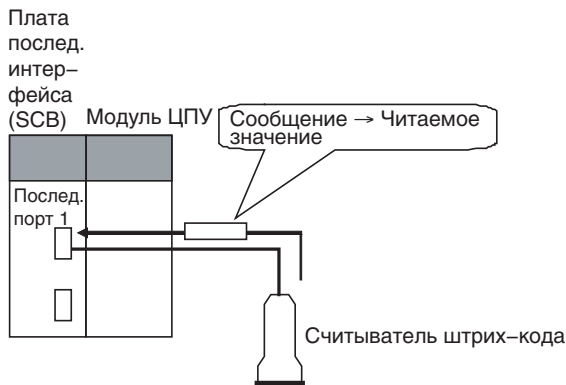
• Условия

Для последовательного порта должен быть выбран режим беспrotocolной связи.

• Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Storage_location	STRING	Указывает переменную для записи принятой текстовой строки.
Number_of_characters	INT, UINT, WORD	Указывает количество принимаемых символов. 0...255
Serial_port	INT, UINT, WORD	Указывает номер последовательного порта. 1: последовательный порт 1; 2: последовательный порт 2.

• Пример



```

Переменные
BOOL P_DoRecvData (* Переменная для управления функцией приема *)
STRING Message (* Переменная для сохранения принятого сообщения *)
BOOL P_EndRecvSCBPort1 (* Флаг завершения приема *) AT A356.06
Используется последовательный порт 1
    
```

```

(* Используется послед. порт номер 1 *)
(* Данные принимаются, если P_DoRecvData = 1 и завершен прием*)
IF (P_DoRecvData = TRUE) AND (P_EndRecvSCBPort1 = TRUE) THEN
    (* Получить 16 символов *)
    RXD_SCB(Message, 16, 1);
    P_DoRecvData := FALSE;
END_IF;
    
```

Сопутствующий флаг вспомогательной области	Адрес	Описание
Флаг завершения приема порта 1	A356.06	Включен, если в беспроточольном режиме завершился прием данных.
Флаг переполнения при приеме порта 1	A356.07	Включен, если во время приема данных в беспроточольном режиме произошло переполнение буфера приема.
Счетчик приема порта 1	A357	Содержит значение количества символов, принятых в беспроточольном режиме.
Флаг завершения приема порта 2	A356.14	Включен, если в беспроточольном режиме завершился прием данных.
Флаг переполнения при приеме порта 2	A356.15	Включен, если во время приема данных в беспроточольном режиме произошло переполнение буфера приема.
Счетчик приема порта 2	A358	Содержит значение количества символов, принятых в беспроточольном режиме.

Дополнительные сведения и меры предосторожности относительно сопутствующих флагов вспомогательной области см. в описании команды связи по последовательному интерфейсу RXD в Справочном руководстве по командам программирования ПЛК серии CS/CJ.

RXD_SCU: прием строки через последовательный порт модуля последовательного интерфейса

• Функция

Прием текстовой строки через последовательный порт модуля последовательного интерфейса (SCU).

• Применение

RXD_SCU (*Storage_location*, *Number_of_characters*, *SCU_unit_number*, *Serial_port*, *Internal_logic_port*);

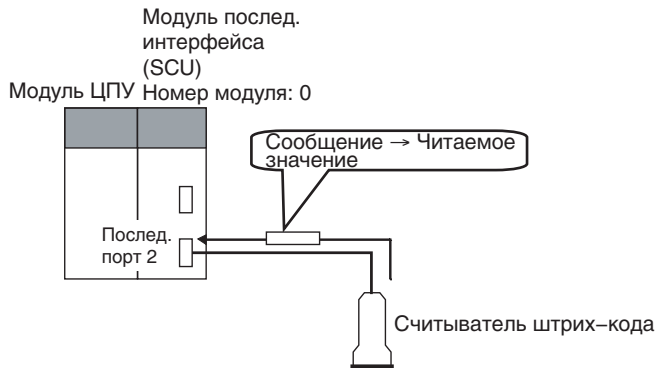
• Условия

Для последовательного порта должен быть выбран режим беспrotocolной связи.

• Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Storage_location	STRING	Указывает переменную для записи принятой текстовой строки.
Number_of_characters	INT, UINT, WORD	Указывает количество принимаемых символов. 0...255
SCU_unit_number	INT, UINT, WORD	Указывает номер модуля последовательного интерфейса.
Serial_port	INT, UINT, WORD	1: последовательный порт 1; 2: последовательный порт 2.
Internal_logic_port	INT, UINT, WORD	0...7: указан номер внутреннего логического порта. 16#F: автоматическое назначение внутреннего логического порта.

• Пример



Переменные

BOOL P_DoRecvData (* Переменная для управления функцией приема *)
 INT iProcess (* Номер процедуры *)
 STRING Message (* Переменная для сохранения принятого сообщения *)
 BOOL P_RXDU_Recv (* Состояние модуля последовательного интерфейса *)
 AT 1519.06 Номер модуля 0
 Используется последовательный порт 2
 BOOL P_ComInstEnable (* Флаг доступности порта связи *) AT A202.07 Используется порт 7

```

(* Использовать следующие параметры: номер модуля: 0; номер послед. порта: 2;
номер логического порта: 7 *)
(* Данные принимаются, если P_DoRecvData = 1, а iProcess = 0 *)
IF (P_DoRecvData = TRUE) AND (iProcess = 0) THEN
    iProcess := 1;
    P_DoRecvData := FALSE;
END_IF;

(* Выполнение процедуры приема с указанным номером процедуры *)
CASE iProcess OF
    1: (* Функция приема выполняется, если установлены флаг доступности порта связи
и флаг завершения приема. *);
        IF (P_ComInstEnable = TRUE) AND (P_RXDU_Recv = TRUE) THEN
            RXD_SCU(Message, 16, 0, 2, 7);
            iProcess := 2;
        END_IF;
    2: (* Прием считается завершенным, если установлен флаг доступности порта связи *)
        IF P_ComInstEnable = TRUE THEN
            iProcess := 0;
        END_IF;
END_CASE;
    
```

Сопутствующий флаг вспомогательной области	Адрес	
Флаги доступности порта связи	A202.00... A202.07	Включен, если есть возможность обмена данными по сети. Номера битов соответствуют непосредственно номерам внутренних логических портов. Биты 00...07: внутренние логические порты 0...7

Сопутствующие биты области модулей шины ЦПУ	Бит	
n = CIO 150 + 25 x номер модуля Порт 1: n+9 Порт 2: n+19	06	Включен, если в беспrotocolном режиме завершился прием данных.

Сопутствующие биты области модулей шины ЦПУ	Бит	
n = CIO 150 + 25 x номер модуля Порт 1: n+9 Порт 2: n+19	07	Включен, если во время приема данных в беспротокольном режиме произошло переполнение буфера приема.
n = CIO 150 + 25 x номер модуля Порт 1: n+10 Порт 2: n+20	---	Содержит значение количества символов, принятых в беспротокольном режиме.

Дополнительные сведения и меры предосторожности относительно сопутствующих флагов вспомогательной области см. в описании команды связи по последовательному протоколу RXDU в Справочном руководстве по командам программирования ПЛК серии CS/CJ.

Функции преобразования угловых величин

DEG_TO_RAD: преобразование из градусов в радианы

- Функция
Перевод значения угла, выраженного в градусах, в радианы.
- Применение
Return_value := DEG_TO_RAD (argument)
- Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Argument	REAL, LREAL	Содержит значение угла в градусах.
Return_value	REAL, LREAL	Возвращает значение угла в радианах.

RAD_TO_DEG: преобразование из радианов в градусы

- Функция
Перевод значения угла, выраженного в радианах, в градусы.
- Применение
Return_value := RAD_TO_DEG (argument)
- Аргументы и возвращаемые значения

Имя переменной	Тип данных	Описание
Argument	REAL, LREAL	Содержит значение угла в радианах.
Return_value	REAL, LREAL	Возвращает значение угла в градусах.

Функции для работы с таймерами/счетчиками

TIMX: 100 мс ТАЙМЕР

- Функция
Реализация таймера, отсчитывающего время в обратном направлении с шагом 100 мс (эквивалент команды TIMX (550) языка LD).
- Когда условие выполнения переключается из «0» (ЛОЖЬ) в «1» (ИСТИНА), запускается таймер с указанным адресом. Текущее значение таймера, начиная с заданной уставки, уменьшается на 1 через каждые 100 мс.
- Обратный отсчет ведется, пока остается включенным («1») условие выполнения. Когда текущее значение достигает 0, устанавливается флаг завершения для таймера с указанным адресом. Если текущее значение не равно нулю, флаг завершения таймера сброшен.
- Пока условие выполнения выключено («0»), текущее значение таймера с указанным адресом остается равным уставке таймера, флаг завершения таймера при этом сброшен.
- Применение
TIMX(Execution_condition, Timer_address, Timer_set_value);

• Аргументы

Имя переменной	Тип данных	Описание
Execution_condition	BOOL	Таймер работает, когда его условие выполнения = ИСТИНА («1»).
Timer_address	TIMER	Указывает используемый адрес таймера (T0...T4095).
Timer_set_value	UINT	Указывает время задержки с дискретностью 100 мс. (&0...&65535, #0...#FFFF)

• Примечание

- Только если в настройках ПЛК выбран параметр *Apply the same spec. as T0-2047 to T2048-4095 (Применить к T2048...4095 настройки T0...2047)*: текущее значение обновляется по завершении всех циклических задач, а также через каждые 80 мс.
- При обращении к флагу завершения таймера из программы пользователя следует учитывать, что изменение состояния данного флага, в зависимости от момента обращения к нему, может отражаться в программе пользователя с задержкой в один цикл.

TIMHX: 10 мс ТАЙМЕР

• Функция

Реализация таймера, отсчитывающего время в обратном направлении с шагом в 10 мс (эквивалент команды TIMHX (551) языка LD).

- Когда условие выполнения переключается из «0» (ЛОЖЬ) в «1» (ИСТИНА), запускается таймер с указанным адресом. Текущее значение таймера, начиная с заданной уставки, уменьшается на 1 через каждые 10 мс.
- Обратный отсчет ведется, пока остается включенным («1») условие выполнения. Когда текущее значение достигает 0, устанавливается флаг завершения для таймера с указанным адресом. Если текущее значение не равно нулю, флаг завершения таймера сброшен.
- Пока условие выполнения выключено («0»), текущее значение таймера с указанным адресом остается равным уставке таймера, флаг завершения таймера при этом сброшен.

• Применение

TIMHX(Execution_condition, Timer_address, Timer_set_value);

• Аргументы

Имя переменной	Тип данных	Описание
Execution_condition	BOOL	Таймер работает, когда его условие выполнения = ИСТИНА («1»).
Timer_address	TIMER	Указывает используемый адрес таймера (T0...T4095).
Timer_set_value	UINT	Указывает время задержки с дискретностью 10 мс. (&0...&65535, #0...#FFFF)

• Примечание

- Только если в настройках ПЛК выбран параметр *Apply the same spec. as T0-2047 to T2048-4095 (Применить к T2048...4095 настройки T0...2047)*: текущее значение обновляется по завершении всех циклических задач, а также через каждые 80 мс.
- При обращении к текущему значению таймера из программы пользователя следует учитывать, что текущие значения таймеров с номерами 0...255, 256...2047 и 2048...4095 могут отличаться друг от друга в связи с тем, что значения этих таймеров обновляются в разные моменты времени.
- При обращении к флагу завершения таймера из программы пользователя следует учитывать, что изменение состояния данного флага, в зависимости от момента обращения к нему, может отражаться в программе пользователя с задержкой в один цикл.

TMNHX: 1 мс ТАЙМЕР

• Функция

Реализация таймера, отсчитывающего время в обратном направлении с шагом в 1 мс (эквивалент команды TMNHX (552) языка LD).

- Когда условие выполнения переключается из «0» (ЛОЖЬ) в «1» (ИСТИНА), запускается таймер с указанным адресом. Текущее значение таймера, начиная с заданной уставки, уменьшается на 1 через каждые 1 мс.
- Обратный отсчет ведется, пока остается включенным («1») условие выполнения. Когда текущее значение достигает 0, устанавливается флаг завершения для таймера с указанным адресом. Если текущее значение не равно нулю, флаг завершения таймера сброшен.
- Пока условие выполнения выключено («0»), текущее значение таймера с указанным адресом остается равным уставке таймера, флаг завершения таймера при этом сброшен.
- Применение
 ТМННХ(*Execution_condition*, *Timer_address*, *Timer_set_value*);

- Аргументы

Имя переменной	Тип данных	Описание
Execution_condition	BOOL	Таймер работает, когда его условие выполнения = ИСТИНА («1»).
Timer_address	TIMER	Указывает используемый адрес таймера (T0...T4095).
Timer_set_value	UINT	Указывает время задержки с дискретностью 1 мс. (&0...&65535, #0...#FFFF)

- Примечание
- Только если в настройках ПЛК выбран параметр *Apply the same spec. as T0-2047 to T2048-4095* (*Применить к T2048...4095 настройки T0...2047*): текущее значение обновляется по завершении всех циклических задач.
- При обращении к текущему значению таймера из программы пользователя следует учитывать, что прочитанное текущее значение таймера с номером 16 и выше может отличаться от текущего значения таймера с номером от 0 до 15. Это связано с тем, что текущие значения таймера 16 и следующих за ним таймеров обновляются только во время выполнения команды, тогда как текущие значения таймеров с номерами от 0 до 15 обновляются с интервалом в 1 мс.
- При обращении к флагу завершения таймера из программы пользователя следует учитывать, что изменение состояния данного флага, в зависимости от момента обращения к нему, может отражаться в программе пользователя с задержкой в один цикл.

TIMUX: 0,1 мс ТАЙМЕР

- Функция

Реализация таймера, отсчитывающего время в обратном направлении с шагом 0,1 мс (эквивалент команды TIMUX (556) языка LD).

- Когда условие выполнения переключается из «0» (ЛОЖЬ) в «1» (ИСТИНА), запускается таймер с указанным адресом. Текущее значение таймера, начиная с заданной уставки, уменьшается на 1 через каждые 0,1 мс.
- Обратный отсчет ведется, пока остается включенным («1») условие выполнения. Когда текущее значение достигает 0, устанавливается флаг завершения для таймера с указанным адресом. Если текущее значение не равно нулю, флаг завершения таймера сброшен.
- Пока условие выполнения выключено («0»), текущее значение таймера с указанным адресом остается равным уставке таймера, флаг завершения таймера при этом сброшен.
- Применение
 TIMUX(*Execution_condition*, *Timer_address*, *Timer_set_value*);

- Аргумент

Имя переменной	Тип данных	Описание
Execution_condition	BOOL	Таймер работает, когда его условие выполнения = ИСТИНА («1»).
Timer_address	TIMER	Указывает используемый адрес таймера (T0...T4095).
Timer_set_value	UINT	Указывает время задержки с дискретностью 0,1 мс. (&0...&65535, #0...#FFFF)

- Примечание
- При длительности цикла в 100 мс или больше данный таймер может работать неправильно.
- При обращении к текущему значению таймера из программы пользователя следует учитывать, что считываемое значение, в зависимости от момента чтения, может отставать от фактического текущего значения на один цикл.
- При обращении к флагу завершения таймера из программы пользователя следует учитывать, что изменение состояния данного флага, в зависимости от момента обращения к нему, может отражаться в программе пользователя с задержкой в один цикл.

TMUX: 0,01 мс ТАЙМЕР

- Функция
Реализация таймера, отсчитывающего время в обратном направлении с шагом 0,01 мс (эквивалент команды TMUX (557) языка LD).
- Когда условие выполнения переключается из «0» (ЛОЖЬ) в «1» (ИСТИНА), запускается таймер с указанным адресом. Текущее значение таймера, начиная с заданной уставки, уменьшается на 1 через каждые 0,01 мс.
- Обратный отсчет ведется, пока остается включенным («1») условие выполнения. Когда текущее значение достигает 0, устанавливается флаг завершения для таймера с указанным адресом. Если текущее значение не равно нулю, флаг завершения таймера сброшен.
- Пока условие выполнения выключено («0»), текущее значение таймера с указанным адресом остается равным уставке таймера, флаг завершения таймера при этом сброшен.
- Применение
`TMUX(Execution_condition, Timer_address, Timer_set_value);`

- Аргументы

Имя переменной	Тип данных	Описание
Execution_condition	BOOL	Таймер работает, когда его условие выполнения = ИСТИНА («1»).
Timer_address	TIMER	Указывает используемый адрес таймера (T0...T4095).
Timer_set_value	UINT	Указывает время задержки с дискретностью 0,01 мс. (&0...&65535, #0...#FFFF)

- Примечание
- При длительности цикла в 10 мс или больше данный таймер может работать неправильно.
- При обращении к текущему значению таймера из программы пользователя следует учитывать, что считываемое значение, в зависимости от момента чтения, может отставать от фактического текущего значения на один цикл.
- При обращении к флагу завершения таймера из программы пользователя следует учитывать, что изменение состояния данного флага, в зависимости от момента обращения к нему, может отражаться в программе пользователя с задержкой в один цикл.

TTMX: НАКАПЛИВАЮЩИЙ ТАЙМЕР

- Функция
Реализация таймера, отсчитывающего время в прямом направлении с шагом 0,1 с (эквивалент команды TTMX (555) языка LD).
- Пока условие выполнения включено («1»), ведется прямой отсчет (т. е. текущее значение прирастает).
- Когда условие выполнения выключается («0»), отсчет времени приостанавливается, текущее значение перестает прирастать, но не сбрасывается (т. е. сохраняется). Когда условие выполнения вновь включается («1»), отсчет времени возобновляется (текущее значение вновь начинает увеличиваться).
- Когда текущее значение достигает заданной уставки, устанавливается флаг завершения работы таймера.
- После достижения таймером заданной уставки текущее значение таймера и флаг завершения работы таймера остаются неизменными.

- Включение входа сброса приводит к сбросу таймера.
- Применение
TTIMX(*Execution_condition, Reset_input, Timer_address, Timer_set_value*);

- Аргументы

Имя переменной	Тип данных	Описание
Execution_condition	BOOL	Пока условие выполнение = ИСТИНА («1»), текущее значение увеличивается (накапливается).
Reset_input	BOOL	При включении входа сброса текущее значение и флаг завершения таймера сбрасываются.
Timer_address	TIMER	Указывает используемый адрес таймера (T0...T4095).
Timer_set_value	UINT	Указывает время задержки с дискретностью 0,1 мс. (&0...&65535, #0...#FFFF)

- Примечание
- Поскольку текущее значение увеличивается только при выполнении команды, данный таймер может работать неправильно при длительности цикла программы ПЛК в 100 мс и больше.
- При обращении к флагу завершения таймера из программы пользователя следует учитывать, что изменение состояния данного флага, в зависимости от момента обращения к нему, может отражаться в программе пользователя с задержкой в один цикл.

CNTX: СЧЕТЧИК

- Функция

Реализация счетчика обратного счета
(эквивалент команды CNTX (546) языка LD).

- Текущее значение счетчика уменьшается на 1 каждый раз, когда включается (переходит из «0» в «1») вход счетчика.
Когда текущее значение достигает 0, устанавливается флаг завершения счетчика.
- Когда включается вход сброса, счетчик сбрасывается, а его текущее значение становится равно уставке. При этом также сбрасывается флаг завершения работы счетчика, а вход счетчика становится недоступным для приема сигналов.

- Применение

CNTX(*Count_input, Reset_input, Counter_address, Counter_set_value*);

- Аргументы

Имя переменной	Тип данных	Описание
Count_input	BOOL	При каждом включении входа счетчика текущее значение счетчика уменьшается на 1.
Reset_input	BOOL	Когда вход сброса включен, текущее значение и флаг завершения счетчика сброшены.
Counter_address	COUNTER	Указывает используемый адрес счетчика (C0...C4095).
Counter_set_value	UINT	Указывает начальное значение счетчика, с которого по умолчанию должен начинаться обратный счет. (&0...&65535, #0...#FFFF)

CNTRX: РЕВЕРСИВНЫЙ СЧЕТЧИК

- Функция

Реализация счетчика прямого/обратного счета
(эквивалент команды CNTRX (548) языка LD).

- При включении входа прямого счета текущее значение увеличивается на 1.
- При включении входа обратного счета текущее значение уменьшается на 1.
- При счете в прямом направлении флаг завершения работы счетчика устанавливается, когда текущее значение счетчика, достигнув заданной уставки, сбрасывается в 0. Затем, когда текущее значение увеличивается с 0 до 1, флаг завершения работы счетчика вновь сбрасывается.

- При счете в обратном направлении флаг завершения работы счетчика устанавливается, когда текущее значение счетчика, достигнув 0, возвращается к заданной уставке. Затем, когда текущее значение становится меньше уставки на 1, флаг завершения работы счетчика сбрасывается.
- Текущее значение не изменяется, если входы прямого и обратного счета включаются одновременно.
- При включенном входе сброса текущее значение счетчика равно 0, а входы счета недоступны для приема сигналов.
- Применение
CNTRX(*Increment_count, Decrement_count, Reset_input, Counter_address, Counter_set_value*);

- Аргументы

Имя переменной	Тип данных	Описание
Increment_count	BOOL	При каждом включении входа прямого счета текущее значение счетчика увеличивается на 1.
Decrement_count	BOOL	При каждом включении входа обратного счета текущее значение счетчика уменьшается на 1.
Reset_input	BOOL	Когда вход сброса включен, текущее значение и флаг завершения счетчика сброшены.
Counter_address	COUNTER	Указывает используемый адрес счетчика (C0...C4095).
Counter set value	UINT	Указывает начальное значение счетчика, с которого по умолчанию должен начинаться обратный счет. (&0...&65535, #0...#FFFF)

TRSET: СБРОС ТАЙМЕРА

- Функция
Сброс указанного таймера
(эквивалент команды TRSET (549) языка LD).
- Применение
TRSET(*Execution_condition, Timer_address*);
- Аргументы

Имя переменной	Тип данных	Описание
Execution_condition	BOOL	Включение условия выполнения приводит к сбросу таймера.
Timer_address	TIMER	Указывает используемый адрес таймера (T0...T4095).

Указатель

А

- Автоматическая генерация определений функциональных блоков, 111
- Адреса
 - Выбор используемых областей, 123
 - Выяснение автоматически назначенных адресов, 125
 - Используемые области, 53
- Алгоритм
 - Создание, 106

В

- Внешние переменные, 24, 47
- Внутренние переменные, 23, 44
- Входные переменные, 23, 41
- Входные-выходные переменные, 44
- Выходные переменные, 23, 42

И

- Имя переменной, 24
- Имя символа
 - Автоматическая генерация, 110

К

- Команды таймеров
 - Ограничения, 65
 - Работа, 92
- Компиляция, 130
- Консоли программирования, 67

М

- Меню, 9
 - Всплывающее, 13
 - Главное, 9
- Меры предосторожности, хxi
 - Обеспечение безопасности, хxii
 - Общие, хxii
 - Эксплуатация, хxiv
- Меры предосторожности и обеспечения безопасности, хxii
- Мониторинг функциональных блоков, 136
- МЭК 61131-3, 4, 8

О

- Области экземпляров, 26, 53
 - Настройка, 26, 123
- Определения функциональных блоков, 16
 - Выяснение оригинала по экземпляру, 128
 - Компиляция, 130
 - Создание, 100
 - Сохранение в файлы, 135
- Отладка функциональных блоков, 136
- Ошибки
 - Функциональные блоки, 67

П

- Параметр AT, 24, 48, 108
 - Ограничения, 63
 - Параметры
 - Общие сведения, 18
 - Параметры массива, 24, 49, 73, 109
 - Переменные
 - Введение, 22
 - Выбор используемых областей, 26
 - Выяснение автоматически назначенных адресов, 125
 - Использование, 23, 40
 - Назначение адресов, 26
 - Ограничения, 64
 - Определения, 39
 - Регистрация до создания программы, 104
 - Свойства, 23, 24, 47
 - Создание по мере необходимости, 107
 - Программирование на языке LD
 - Ограничения в функциональных блоках, 63
 - Определение функционального блока, 102
 - Программирование на языке ST
 - Ограничения, 66
 - Определение функционального блока, 103
 - Проекты
 - Создание, 100
- ## Р
- Различение фронтов
 - Ограничения, 64
 - Редактирование в режиме онлайн
 - Ограничения, 67
 - Определения функциональных блоков, 145

С

Свойства, 4

Т

Таблица глобальных символов, 22

Типы данных, 24, 47

 Определение, 69

Ф

Файлы

 Библиотека, 9

 Определения функциональных блоков, 135

 Текстовые файлы проекта, 9

Функции, 4

 Ограничения, 6

 Функциональные блоки, 7, 8

Функциональные блоки

 Мониторинг, 136

 Настройка параметров, 120

 Общие сведения, 14

 Ограничения, 63

 Определение, 104

 Отладка, 136

 Ошибки, 67

Повторное использование, 29

Порядок работы, 61

Преимущества, 15

Создание, 28

Структура, 16

Указания по использованию, 69

Характеристики, 7, 8, 38

Элементы, 39

Х

Характеристики

 Работа функционального блока, 61

 Экземпляры, 53

 CX-Programmer версии 5.0, 6

Э

Экземпляры

 Количество, 18

 Несколько, 58

 Общие сведения, 17

 Регистрация в таблице глобальных символов, 22

 Создание, 28, 117

 Характеристики, 53

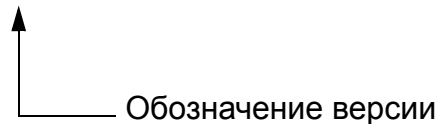
Эксплуатация

 Меры предосторожности, xxiv

Перечень версий

Версия руководства указывается в конце номера каталога на титульной странице руководства.

Cat. No. W447-RU2-12



В таблице ниже показаны изменения, внесенные в настоящее руководство после выхода его оригинальной версии. Номера страниц соответствуют предыдущим версиям.

Обозначение версии	Дата	Изменения
01	Февраль 2005	Оригинальная версия
02	Ноябрь 2005	Добавлена информация в связи с выходом версии 6.1, в частности информация о функциях имитации режима выполнения и мониторинга переменных программы на языке ST.
03	Июль 2006	Добавлена информация в связи с выходом версии 7.0.
04	Январь 2007	Стр. 17 и 18: слово «возможно» заменено словом «невозможно» в таблице (в двух местах), изменено примечание. Стр. 29: изменен текст в ячейке «входы», касающийся состояния значения при следующем выполнении. Стр. 213: изменен рисунок. Стр. 214, 216...218 и 202: изменены рисунок и программа. Стр. 215 и 219: изменен текст в правой нижней ячейке.
05	Июль 2007	Добавлена информация об обновлении версии: 7.0 -> 7.2.
07	Июнь 2008	Добавлена информация об обновлении версии: 7.2 -> 8.0.
08	Февраль 2009	Добавлена информация об обновлении версии: 8.0 -> 8.1.
09	Декабрь 2009	Добавлена информация об обновлении версии: 8.3 -> 9.0.
10	Февраль 2010	Добавлена информация об обновлении версии: 9.0 -> 9.1.
11	Октябрь 2010	Добавлена информация об обновлении версии: 9.1 -> 9.2.
12	Январь 2011	Добавлена информация об обновлении версии: 9.2 -> 9.3.

Перечень версий

Россия
ООО «Омрон Электроникс»
улица Правды, дом 26
Москва, Россия, 125040
Тел.: +7 495 648 94 50
Факс: +7 495 648 94 51
www.industrial.omron.ru

OMRON

Официальный дистрибьютор: