

SIEMENS

SITRAIN

Training for Automation and Drives

SIMATIC S7

Программирование 2

Курс ST-7PRO2

1. Инструкции, зависящие от битов слова статуса (14)

2. Функции с аккумуляторами (15)

3. Инструкции для чисел типа REAL (7)

4. Косвенная адресация (23)

5. Переменные и типы данных STEP7 (28)

6. Вызов блоков и модель мультитекстов (25)

7. Использование библиотек (23)

8. Обработка синхронных и асинхронных ошибок (15)

9. Создание программы в текстовом редакторе (13)

10. Базовые и расширенные S7 коммуникации (34)

11. S7-400 (34)

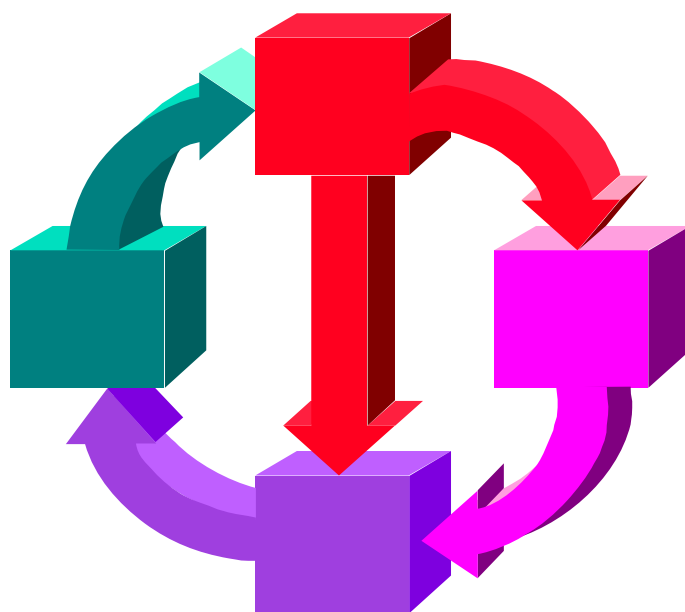
12. Распределённый ввод-вывод и назначение параметров (17)

13. Инжиниринговые пакеты для S7/M7 (53)

14. Решения упражнений (52)

15. Приложение: Косвенный доступ к параметрам (18)

Инструкции, зависимые от битов слова статуса



SIMATIC S7

Siemens AG 1999. All rights reserved.

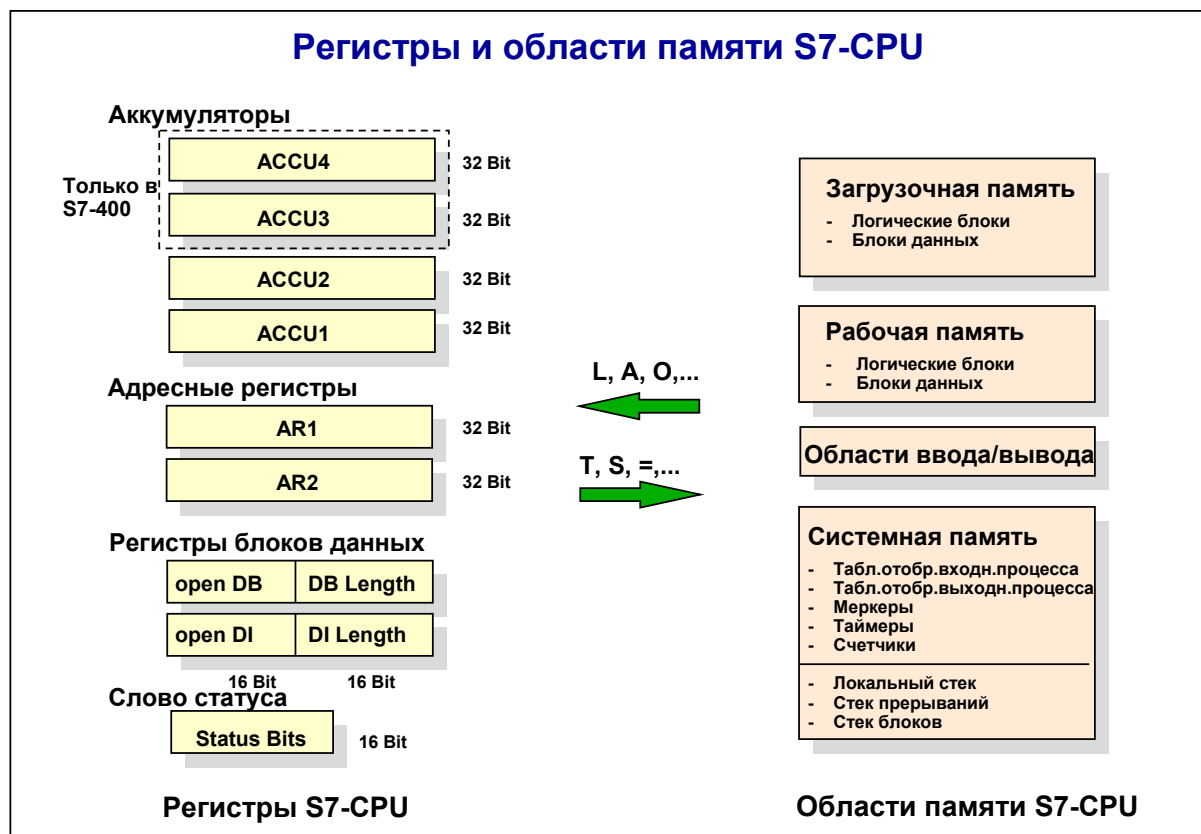
Date: 04.11.2005
File: PRO2_01E.1Information and Training Center
Knowledge for Automation

Содержание

Стр

Регистры и области памяти S7-CPU	2
Структура слова статуса	3
Проверка битов слова статуса	4
Инструкции с битами слова статуса	5
BR бит и ENO в вызовах блока и в сложных функциях	6
Инструкции переходов, зависящие от слова статуса	7
Функции перехода, зависящие от кодов состояния	8
Программирование распределенных переходов	9
Программирование инструкции цикла	10
Инструкции окончания блока	11
Упражнение 1.1: Переход после вычитания	12
Упражнение 1.2: Переход после умножения	13
Упражнение 1.3: Программирование распределенного перехода	14

Регистры и области памяти S7-CPU



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.2Information and Training Center
Knowledge for Automation

Регистры CPU

Регистры CPU используются для адресации или обработки данных. Данные могут с помощью соответствующих команд (L, T, ...) быть обменены между областями памяти CPU и регистрами:

- **Аккумуляторы:** Два (в S7-300) или четыре (в S7-400) аккумулятора используются для арифметики, сравнений с байтами, словами или двойными словами.
- **Адресные регистры:** Два адресных регистра используются как указатели для косвенной адресации памяти.
- **Регистры блоков данных:** Регистры блоков данных содержат номера открытых блоков данных. Таким образом возможно, что открыты одновременно два DB: один DB с помощью регистра DB, другой как экземпляр DB с помощью регистра DI. Когда DB открыт, его длина (в байтах) автоматически загружается в связанный с ним регистр.
- **Слово статуса:** Содержит различные биты, которые отражают результат или статус отдельных инструкций во время выполнения программы.

Области памяти

Память S7-CPU может быть разделена на четыре области:

- **Загрузочная память** используется, чтобы хранить программу пользователя без символов и комментариев. Загрузочная память может быть выполнена в виде RAM или FLASH EPROM.
- **Рабочая память** (встроенная RAM) используется, чтобы хранить соответствующую часть S7-программы, необходимую для выполнения программы. Программа выполняется исключительно в рабочей памяти.
- **Область ввода - вывода** разрешает прямой доступ ко входам и выходам, связанных с ней сигнальных модулей.
- **Системная память** (RAM) содержит области отображения входного и выходного процессов, меркеры, таймеры и счетчики. Кроме того, она содержит локальный стек, стек блоков и стек прерываний.

Структура слова статуса

Значение битов в слове статуса

Бит	Название	Величина	Значение
0	/FC	2^0	Бит первичного опроса
1	RLO	2^1	Результат логической операции
2	STA	2^2	Статус
3	OR	2^3	Или
4	OS	2^4	Переполнение с запоминанием
5	OV	2^5	Переполнение
6	CC 0	2^6	Код состояния
7	CC 1	2^7	Код состояния
8	BR	2^8	Двоичный результат
9...15	Не используются	$2^9 \dots 2^{15}$	

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.3Information and Training Center
Knowledge for Automation

Слово статуса

Отдельные биты слова статуса, дают информацию относительно результата или статуса инструкций, а также ошибок, которые возникли. Вы можете использовать биты слова статуса непосредственно в Вашей программе, используя двоичные логические операции и, таким образом, управлять выполнением программы.

Первичный опрос

Бит 0 слова статуса называется битом первичного опроса. Состояние сигнала "0" в бите /FC указывает, что со следующей логической инструкции начинается новая логическая цепочка в Вашей программе. Диагональная черта перед сокращением FC указывает, что /FC -бит инвертирован.

Результат логической операции

Бит 1 слова статуса - RLO -бит (RLO = "Result of Logic Operation"). Он используется как временная память в двоичных логических операциях.

Все бинарные логические команды в цепочке логических операций, начиная со второй, опросив сигнал на контакте, используют результат опроса в качестве своего первого операнда, а значение RLO в качестве второго. Результат логического действия в свою очередь сохраняют снова в RLO-бите.

Бит статуса

Бит статуса (бит 2) сохраняет значение адресованного бита. Бит статуса всегда показывает, для команд опроса (A, O, ...) или записи (=, S, R,) статус адресованного бита (для инструкции записи, статус показывается после выполнения инструкции).

Бит OR

OR бит требуется, когда Вы выполняете операцию И перед логической инструкцией ИЛИ (Команда O без операндов). OR бит указывает, что предварительно выполненная логическая операция И поставила значение "1", посредством чего результат логического действия ИЛИ уже определен как "1".

OV бит

OV бит (переполнение) показывает ошибку в математической инструкции или инструкции сравнения чисел с плавающей точкой. Бит устанавливается согласно результату выполненной математической инструкции или инструкции сравнения.

Проверка битов слова статуса

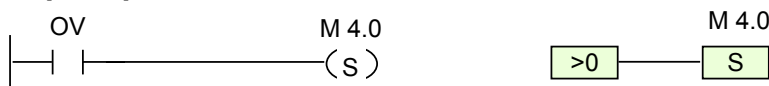
Проверка в STL

- A OV Просмотр переполнения
- A OS Просмотр переполнения с запоминанием
- A BR Просмотр BR-флага

Проверка кода состояния (CC0, CC1)

- A == 0 Результат равен 0
- A > 0 Результат больше 0
- A <> 0 Результат не равен 0
- A <= 0 Результат меньше либо равен 0
и т.д.
- A UO Потеря порядка

Проверка в LAD и в FBD



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.4Information and Training Center
Knowledge for Automation

OS бит

OS бит (переполнение с запоминанием) устанавливается вместе с OV битом. OS бит остается установленным после математической инструкции, выполненной без переполнения, то есть она не изменяется результатом следующей математической инструкции.

Таким образом, Вы имеете возможность, в любом месте блока после инструкции, вызвавшей переполнение, оценивать переполнение. OS бит сбрасывается командами: JOS (переход, если OS = 1), вызов блока CALL и конец блока.

CC1 и CC0

Биты CC1 и CC0 (коды состояния) информируют относительно результатов:

- математических инструкций
- инструкций сравнения.
- логических инструкций со словами
- инструкций сдвигов

Коды состояния CC1 и CC0 можно проверить, используя следующие инструкции.

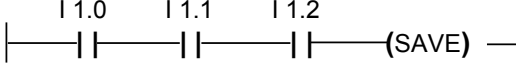
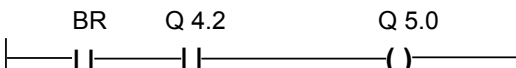
CC1	CC0	Проверка выполнена, если:	
0	0	A ==0	Результат = 0 (ACCU2 = ACCU1)
1	0	A >0	Результат > 0 (ACCU2 > ACCU1)
0	1	A <0	Результат < 0 (ACCU2 < ACCU1)
1	1	A UO	Потеря порядка (напр., деление на 0).

Кроме того, существуют команды переходов, которые оценивают цифровые результаты и, таким образом, разрешают необходимый переход в программе.

LAD/FBD

Вы можете найти команды проверки на языках LAD или FBD в Каталоге в разделе Status Bits.

Инструкции с битами слова статуса

Инструкция	Значение	Пример
• SET	Установка RLO в "1"	SET //RLO-1 = M 0.1
• CLR	Установка RLO в "0"	CLR //RLO-0
• NOT	Инвертирование RLO	O Manual_operation O Automatic_operation NOT = Operating_modes = M0.0
• SAVE	Сохранить RLO в BR (в бинарном результате)	
• A BR	Опросить BR	

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.5Information and Training Center
Knowledge for Automation**L STW/T STW**

Можно также загрузить полное слово статуса и сохранить его для более поздней проверки.

- L STW Загрузка слова статуса
- T MW 114 Сохранение в меркерном слове 114

С помощью инструкции T STW, в слово статуса может, например, быть загружено с предварительно сохраненное значение. На биты 0, 2, 3, 9 .. 15 эта инструкция не воздействует.

Изменение RLO

В STEP7 имеется множество инструкций, чтобы изменить RLO.

С помощью инструкции SET Вы устанавливаете результат OR и /FC, то есть после них начинается новая логическая цепочка.

Инструкция NOT инвертирует результат логической операции.

BR бит

BR бит представляет бит внутренней памяти, в котором может быть сохранен RLO перед инструкциями, изменяющими его. Это делается для того, чтобы RLO был впоследствии снова доступен для возобновления прерванной цепочки.

Если Вы пишете функциональный блок или функцию и хотите вызвать его в LAD, Вы должны управлять BR битом. BR бит соответствует выходу ENO в LAD-представлении.

Установка и сброс BR

Командой SAVE Вы сохраняете RLO в BR-бите.

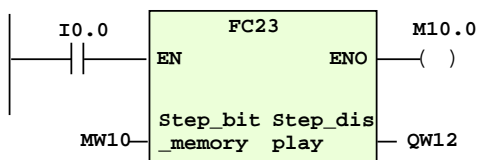
SAVE передает сигнал состояния из RLO в бит статуса BR.

SAVE выполняется независимо от любых условий и не затрагивает никакие другие биты статуса.

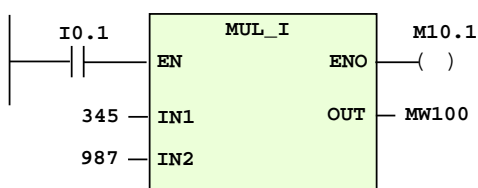
BR бит и ENO в вызовах блока и в сложных функциях

LAD

Network 1: Cyclic Program



Network 2: ???



STL

Network 1: Cyclic Program

```

A      I      0.0
JNB    _001
CALL   FC      23
      Step_bit_memory :=MW10
      Step_display    :=QW12
_001:  A      BR
      =      M      10.0

```

Network 2: ???

```

A      I      0.1
JNB    _002
L      345
L      987
*I
T      MW      100
AN     OV
SAVE
CLR
_002:  A      BR
      =      M      10.1

```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.6Information and Training Center
Knowledge for Automation

EN = Enable input

Вы можете управлять вызовом блока с помощью входа EN. Он существует в каждом блоке или сложной функции в LAD.

- если EN не активизирован, (то есть состояние сигнала - "0"), то блок не выполняет свою функцию. Разрешающий выход ENO также соответственно не активизирован.
- если EN активизирован (то есть состояние сигнала - "1"), то функция блока выполнена.

ENO = Enable output

Вызываемый блок или сложная функция может с помощью разрешающего выхода ENO подать сигнал, выполнена она с ошибкой или нет.

Вы можете использовать BR-бит слова статуса, чтобы сохранить ошибки. BR-бит изменяется только в соответствии с программой пользователя, а не системой.

Если ошибка происходит в течение выполнения, Вы можете сохранить это состояние ошибки, сбрасывая BR-бит. После выполнения блока в LAD/FBD, состояние BR-бита копируется в "выходной параметр" ENO. Однородный механизм для статуса ошибки таким образом доступен в STEP 7. Таким образом, например, вызываемый блок может сообщать вызывающему блоку, была ли обработка выполнена без ошибок или нет.

Обратите внимание Параметр EN - не настоящий параметр входа. Если он назначен, то автоматически создается инструкция условного перехода на метку, расположенную после выполнения блока.

Так же ENO - не настоящий выходной параметр. Если ENO назначен, то автоматически создается две инструкции для копирования BR-бита в данный выходной параметр.

Инструкции переходов, зависящие от слова статуса

- JU Label¹⁾ Безусловный переход
- JC Label¹⁾ Переход, если RLO =1
- JCN Label¹⁾ Переход, если RLO = 0
- JCB Label¹⁾ Переход, если RLO = 1 и запомнить RLO в BR
- JNB Label¹⁾ Переход, если RLO = 0 и запомнить RLO в BR
- JBI Label¹⁾ Переход, если BR = 1
- JNBI Label¹⁾ Переход, если BR= 0
- JO Label¹⁾ Переход, если OV =1
- JOS Label¹⁾ Переход, если OS =1

1) Метка может состоять макс. из 4 символов: букв и цифр. Первый - буква или _

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.7



Information and Training Center
Knowledge for Automation

Функции перехода С помощью функции перехода, Вы можете прерывать линейную обработку программы и продолжать в другом месте блока. Переход в программе может быть выполнен независимо от условий или только тогда, когда выполнены определенные условия.

Безусловный переход Функция перехода JU выполняется всегда, то есть независимо от любых условий. JU прерывает линейную обработку программы и возобновляет ее с метки перехода. JU не изменяет биты слова статуса.

Функции перехода с RLO и BR Переход в программе может быть зависящим от RLO и BR битов. Кроме того, можно совершать переход, зависящий от RLO и одновременно сохранить RLO в BR бите.
RLO-зависимые функции переходов (JC, JCN) устанавливают не только для выполненного, но также и для невыполненных условий, биты статуса STA и RLO в "1" и биты OR и /FC в "0".
Функции перехода (JCB, JNB), сохраняющие RLO, запоминают состояние RLO в BR-бите. Оставшиеся биты, STA, RLO, OR и /FC, обрабатываются тем же самым способом, как и в функциях переходов, которые не сохраняют RLO.
Функции перехода (JBI, JNBI), зависящие от BR бита, не только для выполненного, но также и для невыполненных условий, устанавливают биты слова статуса STA в "1" и OR и /FC биты в "0". Биты RLO и BR остаются неизменными.

Функции переходов, зависящие от кодов состояния

- JZ Label¹⁾ Переход, если CC 1=0 и CC 0=0
(Результат = 0)
- JN Label¹⁾ Переход, если CC1 не равен CC0
(Результат <> 0)
- JP Label¹⁾ Переход, если CC 1=1 и CC 0=0
(Result > 0)
- JM Label¹⁾ Переход, если CC 1=0 и CC 0=1
(Результат < 0)
- JPZ Label¹⁾ Комбинация переходов JZ и JP
(Результат >= 0)
- JMZ Label¹⁾ Комбинация переходов JM и JZ
(Результат <= 0)
- JUO Label¹⁾ Переход, если: неправильное число типа REAL или
деление на 0

1) Метка может состоять макс. из 4 символов: букв и цифр. Первый - буква или _

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.8



Information and Training Center
Knowledge for Automation

Функции перехода с OV и OS

Переходы JO и JOS выполняются, если произошло переполнение. В цепочке с несколькими последовательно выполняемыми инструкциями оценка OV бита должна иметь место после каждой математической функции. Математическая инструкция, которая вызвала переполнение, устанавливает OV бит. Если следующая за ней инструкция выполнена корректно (без переполнения), то бит OV будет сброшен. Чтобы оценить переполнение в группе инструкций, Вы должны проверять OS бит. OS бит сбрасывается только при вызове блока и окончании блока, а также командой перехода JOS. Остальные биты слова состояния не изменяются командами JO и JOS.

Функции перехода CC0 и CC1

Выполнение программы можно сделать зависящим от состояния битов CC0 и CC1. Таким образом, Вы можете, например, проверять, является ли результат вычисления положительным, нулевым или отрицательным. Функции перехода, зависящие от состояния битов CC0 и CC1 не изменяют никакие биты слова состояния. Результат логической операции не изменяется при переходе и может, таким образом, использоваться для дальнейших логических действий в программе пользователя (бит /FC не изменяется).

Пример

Вычитание двух целых чисел с последующей оценкой :

L MW2

L MW8

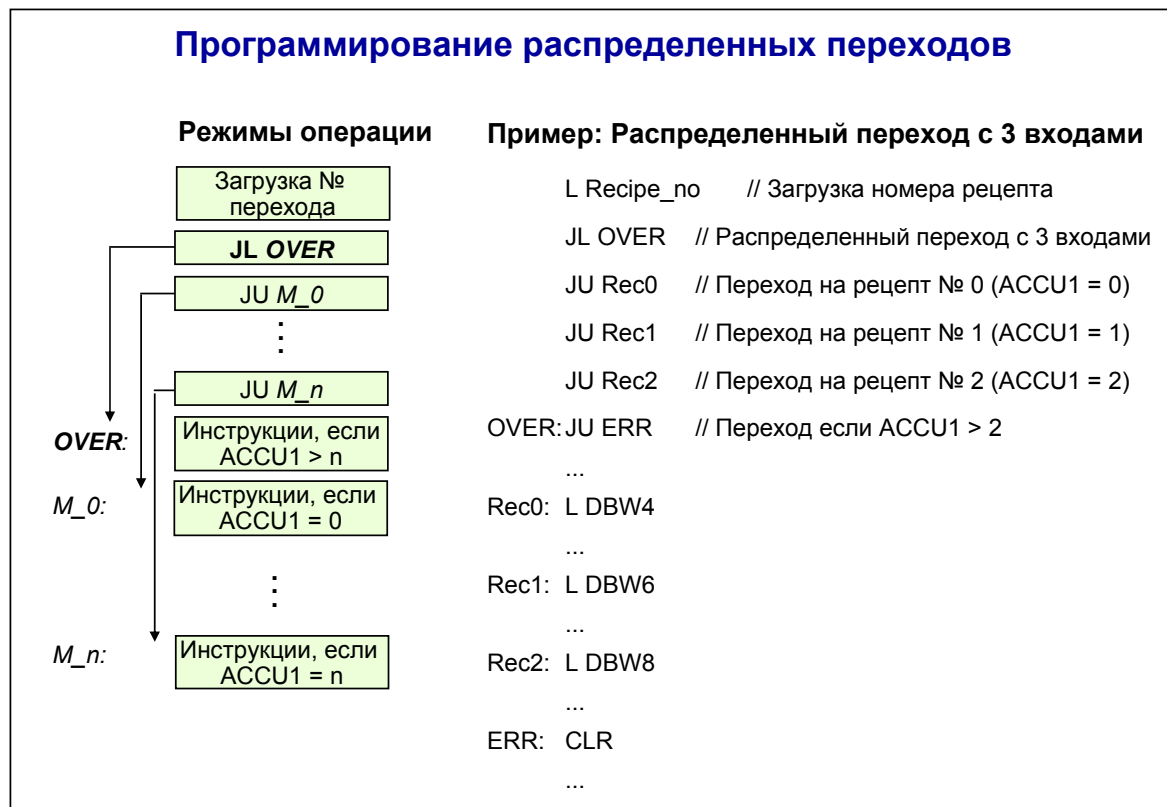
-I

JZ ZERO // Переход происходит, если результат равняется "0"

// Инструкции, если результат не равен "0"

ZERO: // Инструкции для реакции, когда результат равняется "0"

Программирование распределенных переходов



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.9Information and Training Center
Knowledge for Automation

Распределитель переходов

Распределитель переходов JL разрешает переход к программной секции в блоке, зависящий от номера перехода. JL инструкция работает вместе со списком функций переходов JU.

Этот список следует немедленно после JL и может включать максимум 256 переходов. Команда JL имеет метку перехода, которая указывает на конец списка, то есть стоит у первой инструкции после списка.

Только JU инструкции могут быть помещены между JL < метка_перехода > и < метка_перехода >: <инструкция>. Если в ACCU1-L-L расположен "0", выполняется первая инструкция скачка, при "1" - вторая, и т.д. Если число больше, чем длина списка (больше числа команд JU, следующих за JL), JL передает управление на команду, следующую за ветвями перехода. Инструкция JL выполняется независимо от любых условий и не изменяет биты слова состояния.

Обратите внимание

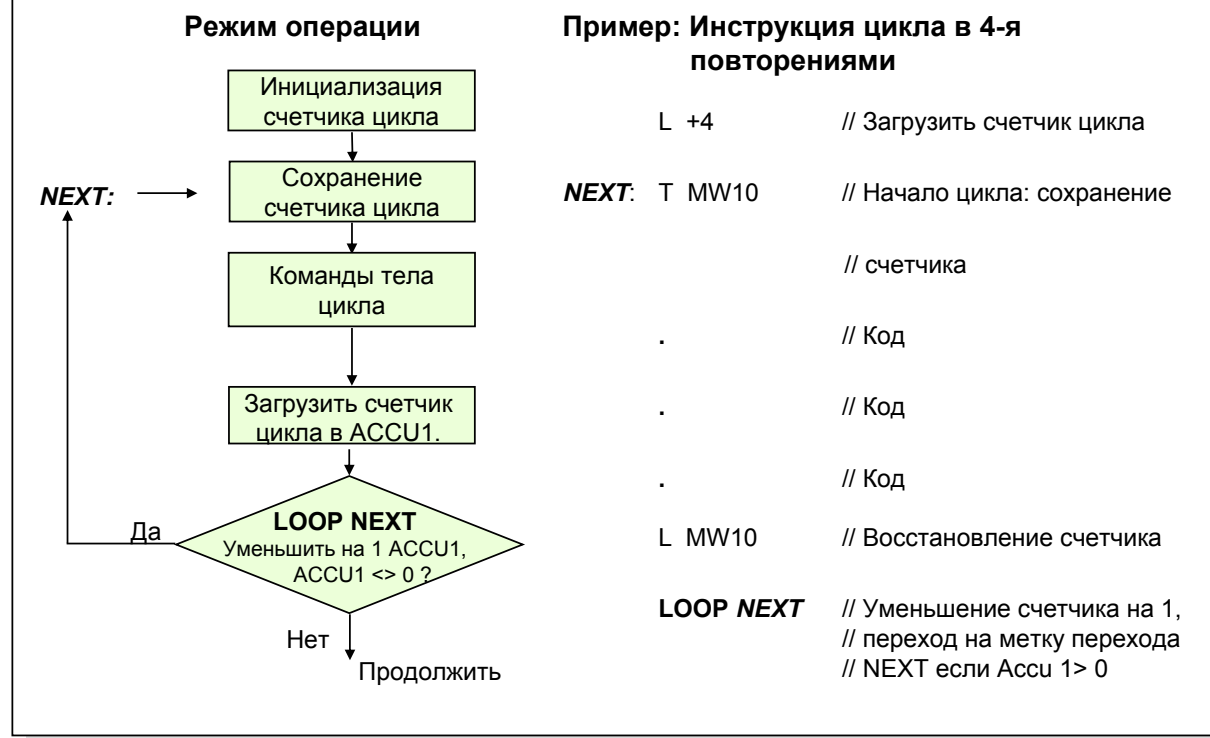
Переходы могут быть сделаны как вперед, так и назад. Переходы могут быть выполнены только в пределах одного блока, то есть инструкция перехода, и метка перехода должны быть в пределах одного и того же блока.

Метка перехода может быть указана только однажды в пределах блока. Максимальная длина перехода находится между -32768 или + 32767 инструкций кода программы. Фактическое максимальное число инструкций зависит от инструкций, используемых в Вашей программе (инструкции могут состоять из одного, двух и трех слов).

Длина метки перехода ограничена четырьмя алфавитно-цифровыми символами, при чем первый символ должен быть буквой или знаком "_" (Знак "_" применяет транслятор для своих меток, поэтому им пользоваться не рекомендуется). В метках различаются большие и малые буквы.

Инструкция, на которую совершается переход, должна всегда помещаться после метки перехода и отделяться от нее ":". Недопустим переход на метку без инструкции.

Программирование инструкции цикла



SIMATIC S7

Siemens AG 1999. All rights reserved.

 Date: 04.11.2005
 File: PRO2_01E.10

 Information and Training Center
 Knowledge for Automation

Повторение инструкций

Инструкция LOOP упрощает программирование циклов в программе.

Для программирования LOOP необходимо загрузить в ACCU1-L число повторов цикла. LOOP интерпретирует правое слово ACCU 1 как число без знака в диапазоне от 0 до 65535.

При каждом выполнении инструкции LOOP, значение в ACCU1-L уменьшается на 1. Далее значение сравнивается с нулем. Если значение не равно нулю, происходит переход на метку, указанную в инструкции LOOP. Если значение равно нулю, выполняется инструкция, следующая за командой LOOP.

Обратите внимание Счетчик циклов не должна быть инициализирован 0, потому что при этом цикл будет выполняться 65535 раз.

Инструкции окончания блока

- **BE** Конец блока

- **BEU** Безусловное окончание блока (В пределах блока)

- **BEC** Условное окончание блока (зависит от RLO)

—(RET) в представлении LAD

— RET в представлении FBD

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.11



Information and Training Center
Knowledge for Automation

Окончание блока

Вы можете закончить обработку блока командой BEC, действие которой зависит от результата логической операции.

Действие команд BEU или BE не зависит ни от каких условий.

BE

Инструкция BE заканчивает выполнение программы текущего блока. BE - всегда последняя инструкция блока. Она автоматически ставится PG, когда блок сохраняется. Таким образом, программист не должен ее писать. Команда BE не видима.

Операционная система осуществляет переход к блоку, вызвавшему текущий, и продолжает обработку программы с первой команды, следующей за командой вызова. Зарезервированная локальная область данных высвобождается. Блоки данных вызывающего блока снова станут действительными. Бит RLO также передается вызывающему блоку.

Предостережение : BE может также использоваться внутри блока. Это также возможно для пропуска инструкций, расположенных за BE. BE в блоке OB передает управление операционной системе.

BEU

BEU инструкция заканчивает выполнение программы в текущем программном блоке точно так же, как BE.

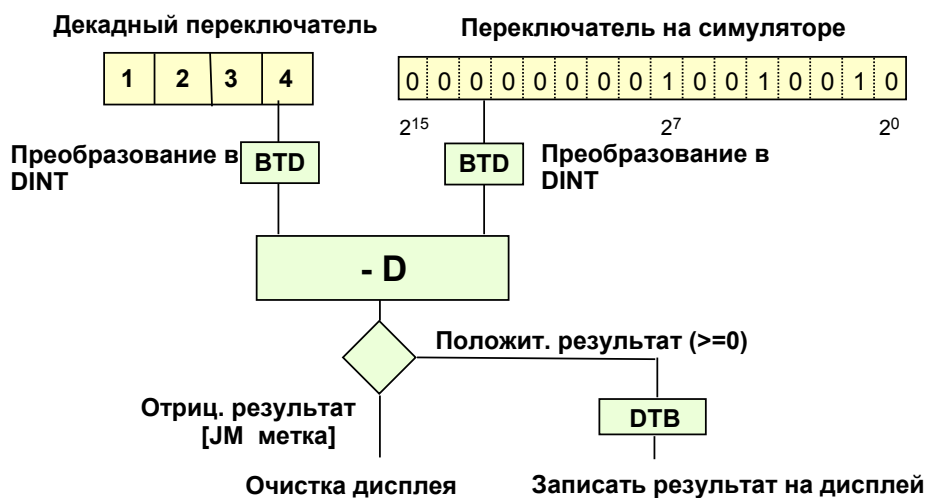
В отличие от инструкции BE, Вы можете программировать инструкцию BEU неоднократно в пределах блока. Секция программы после BEU только тогда обрабатывается, если на нее происходит передача управления командой перехода.

BEC

Конец блока, зависимый от RLO. Для RLO = 1, программа обработки текущего блока завершается и продолжается в блоке, вызвавшем текущий, с первой команды, следующей за командой вызова. Зарезервированная локальная область данных высвобождается. В вызывающий блок также передается признак RLO=1.

Для RLO=0 инструкция BEC не прекращает выполнение текущего блока. Выполняется инструкция, следующая за командой BEC. RLO устанавливается в 1 после команды BEC.

Упражнение 1.1: Переход после вычитания



Декадный переключатель : S7-300: IW4
S7-400: IW 30

Переключатель на симуляторе : S7-300: IW0
S7-400: IW28

Дисплей: S7-300: QW6
S7-400: QW38

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.12



Information and Training Center
Knowledge for Automation

Краткий обзор

Функции перехода прерывают линейную обработку программы. Такой переход может, в частности, быть сделан в зависимости от различных условий.

Цель упражнения

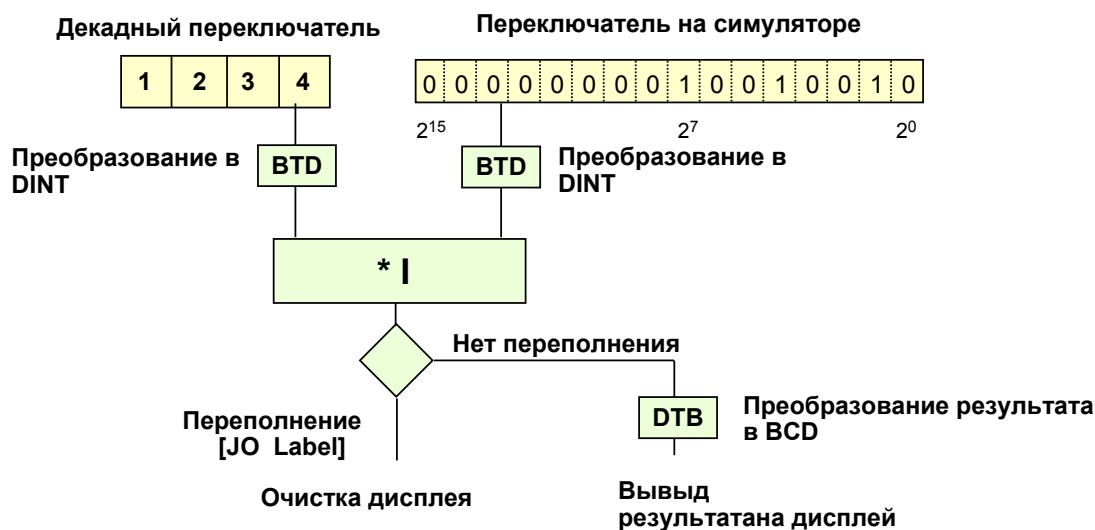
Программирование функции перехода, выполнение которой зависит от результата вычитания.

Задача

Создайте проект PRO2 и затем папку с именем S7-program EXERCISE. Создайте FC 11 со следующими функциональными возможностями:

1. Загрузите входные слова из декадного переключателя и переключателя на симуляторе как BCD -значения в ACCU1 и ACCU2.
2. Выполните преобразование значений к DINT. Для преобразования используйте команду BTD (BCD_TO_DINT). Эта команда преобразует числа в BCD-коде в двоичный код.
3. Вычитайте "значение" переключателя на симуляторе из "значения" декадного переключателя.
4. Выполните в зависимости от результата, следующие действия:
Результат < 0 : Очистите дисплей, то есть передайте на него "0"
Результат >= 0: Выведите выходное BCD -значение на дисплей
Примечание: Используйте команду перехода " JM [метка] ".
 Для маскировки ошибок преобразования во время установки цифр, создайте OB121 с инструкцией: NOP 0.
5. Вызовите FC11 из OB1, загрузите блоки (OB1, OB121 и FC11) в S7-CPU.
6. Протестируйте программу.

Упражнение 1.2: Переход после умножения



Декадный переключатель : S7-300: IW4
S7-400: IW 30

Переключатель на симуляторе : S7-300: IW0
S7-400: IW28

Дисплей: S7-300: QW6
S7-400: QW38

SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.13



Information and Training Center
Knowledge for Automation

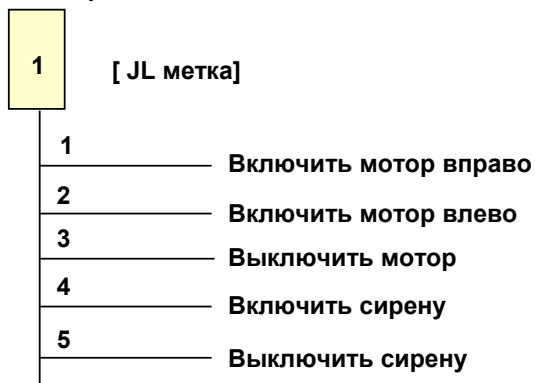
Цель упражнения Программирование функции перехода, выполнение которой зависит от результата умножения.

- Задача** Создайте FC 12 со следующими функциональными возможностями:
- Загрузите в аккумуляторы входные слова декадного переключателя и переключателя на симуляторе как BCD-кодированные значения (без знака) .
 - Выполните преобразование значений к DINT. Для преобразования используйте команду BTB (BCD_TO_DINT). Эта команда делает возможным интерпретировать прочитанные значения как положительные десятичные числа с четырьмя цифрами.
 - Выполните 16-разрядное умножение.
 - Проверьте Ваши результаты вычисления на переполнение и выполнит следующие действия:
Переполнение: Очистите дисплей
Нет переполнения: Выведите результат на дисплей (по крайней мере четыре младшие цифры) .
Примечания: Используйте команду перехода " JO [метка] " для проверки на переполнение.
 Для маскировки ошибок преобразования во время установки цифр создайте OB121 с инструкцией: NOP 0.
 - Вызовите FC12 в OB1 и загрузите программу (OB1, OB121 и FC12) в S7-CPU.
 - Проверьте Вашу программу.

Упражнение 1.3: Программирование распределенного перехода

Функция:

Декадный переключатель



Метка: Переход через список переходов

Адреса:	S7-300 (16-Bit)	S7-300 (32-Bit)	S7-400
Мотор вправо:	Q20.5	Q8.5	Q40.5
Мотор влево :	Q20.6	Q8.6	Q40.6
Сирена:	Q20.7	Q8.7	Q40.7

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.14



Information and Training Center
Knowledge for Automation

Цель упражнения Вы знакомитесь с использованием перехода по списку (распределенного перехода).

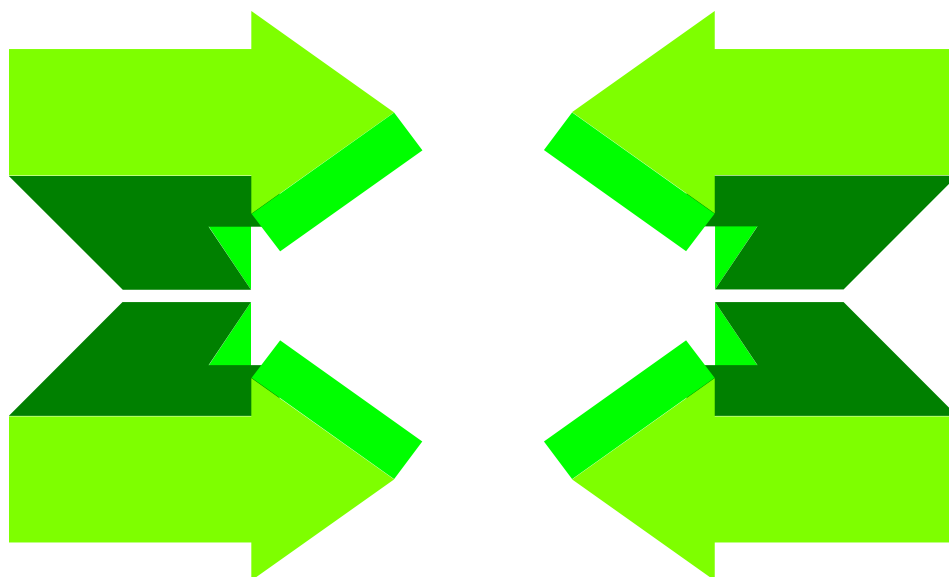
Задача Создайте FC 13 со следующими функциональными возможностями:

- Можно передать число от 1 до 5 , используя входной параметр #Select типа INT.
- В зависимости от числа, выполняются следующие действия :
 - 1: Конвейер движется вперед.
 - 2: Конвейер движется назад.
 - 3: Конвейер останавливается.
 - 4: Включается гудок.
 - 5: Выключается гудок.
- Все другие числа интерпретируются как ошибочные, то есть гудок выключен и конвейер остановлен, и "выходной параметр" ENO установлен в FALSE.

Что делать

1. Создайте FC13 с описанными выше функциональными возможностями. При выполнении перехода по списку заметьте, что могут использоваться только абсолютные переходы.
2. Создайте вызов FC13 в OB1, зависящий от I 0.0.
Входной параметр #Select получает фактическое значение со входа декадного переключателя , а "выходной параметр " ENO подключен на выход Q8. 0 (Q36.0).
3. Загрузите FC13 и OB1 и проверьте Вашу программу.

Функции с аккумуляторами



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.1Information and Training Center
Knowledge for Automation

Содержание

Стр

Обзор инструкций, использующих аккумуляторы	2
Инструкция TAK	3
Инструкции PUSH и POP	4
Инструкции ENT и LEAVE (только для S7-400)	5
Арифметические инструкции	6
Логические инструкции для слов	7
Инструкции обмена байтов в ACCU1	8
Инструкции инкремента и декремента для ACCU1	9
Формирование дополнений	10
Инверсия знака (двойное дополнение)	11
Инструкции циклического 32-битного сдвига через бит CC1	12
Упражнение 2.1: Вычисление степени	13
Упражнение 2.2 : Обмен данных в ACCU1	14
Упражнение 2.3 : Формирование дополнения	15

Обзор инструкций, использующих аккумуляторы

Инструкции, которые используют несколько аккумуляторов

- TAK: Обмен содержимого ACCU1 и ACCU2
- PUSH: Сдвиг содержимого ACCU "вверх"
- POP: Сдвиг содержимого ACCU "вниз"
- ENT: Сдвиг содержимого ACCU "вверх", без ACCU1
- LEAVE: Сдвиг содержимого ACCU "вниз", без ACCU1
- Арифметические инструкции и инструкции побитовой логики для слов

Инструкции, использующие только ACCU1

- INC: Инкрементирование содержимого ACCU1-LL
- DEC: Декрементирование содержимого ACCU1-LL
- SAW: Изменение порядка байтов в ACCU1-L (симметричное отображение)
- CAD: Изменение порядка младших байтов в ACCU1-L
- INVI, INVD: Формирование дополнения (инвертирование разрядов)
- NEGI, NEGD, NEGR: Формирование двойного дополнения (отрицание)
- RLDA, RRDA: Сдвиг содержимого ACCU1 влево или вправо через бит CC1

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.2Information and Training Center
Knowledge for Automation

Краткий обзор

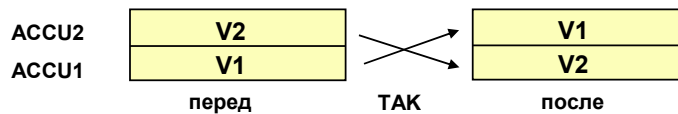
Функции, использующие аккумуляторы, передают значения между аккумуляторами или обменивают байты в ACCU1. Выполнение чистых аккумуляторных функций не зависит от результата логической операции или других битов слова состояния сигнала.

Данные операции также не изменяют битов слова состояния.

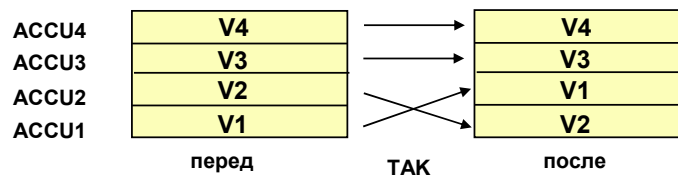
Аккумуляторные функции позволяют создавать оптимальные по времени исполнения программы для задач автоматизации.

Инструкция TAK

S7-300:



S7-400:



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.3



Information and Training Center
Knowledge for Automation

ТАК

Инструкция ТАК обменивает содержимое ACCU1 с содержимым ACCU2. Инструкция выполняется независимо от битов слова состояния и не меняет биты слова состояния. Содержание ACCU3 и ACCU4 остается неизменным для CPU с четырьмя ACCU. (Для S7-400).

Пример

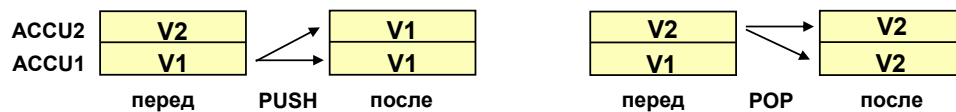
Вычитание меньшего значения из большего :

```

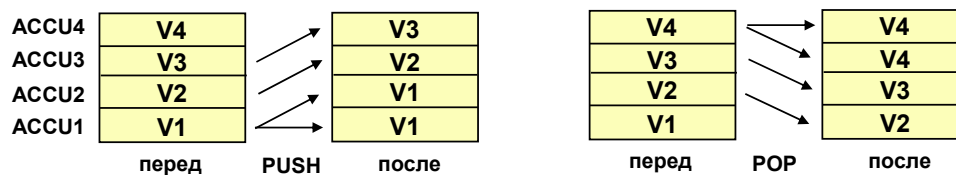
L   MW10    // Загрузка содержимого MW10 в ACCU1-L.
L   MW12    // Загрузка содержимого ACCU1-L в ACCU2-L, а
              // содержимого MW12 в ACCU1-L.
>I              // Проверка: ACCU2-L (MW10) больше, чем ACCU1-L
              //(MW12).
JC   NEXT   // Переход на метку NEXT, если ACCU2 (MW10)
              // больше, чем ACCU1 (MW12).
TAK              // Обмен содержанием ACCU1 и ACCU2.
NEXT: -I              // Вычитание содержимого ACCU1-L из содержимого
              // ACCU2-L.
T   MW14    // Передача результата (= большее значение минус
              // меньшее значение) в MW14
  
```

Инструкции PUSH и POP

S7-300:



S7-400:



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.4



Information and Training Center
Knowledge for Automation

PUSH

Инструкция PUSH сдвигает содержимое аккумулятора в следующий более высокий аккумулятор (с большим номером). PUSH обычно используется, чтобы продублировать значение ACCU1, без потери первоначального содержания ACCU2 или ACCU3 (только для S7-400).

- PUSH (S7-300): Инструкция PUSH копирует содержимое ACCU1 в ACCU2. ACCU1 остается неизменным.
- PUSH (S7-400): Инструкция PUSH копирует содержимое ACCU3 в ACCU4, содержимое ACCU2 в ACCU3 и содержимое ACCU1 в ACCU2. ACCU1 остается неизменным.

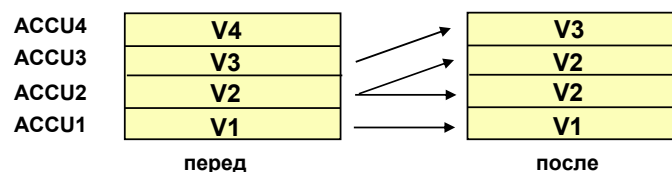
POP

Инструкция POP сдвигает значения аккумуляторов от 2-го до 4-го в. Эта инструкция обычно выполняется после инструкций передачи, когда содержимое ACCU1 больше не требуется и обработка должна продолжиться со значениями, сохраненными в верхних аккумуляторах.

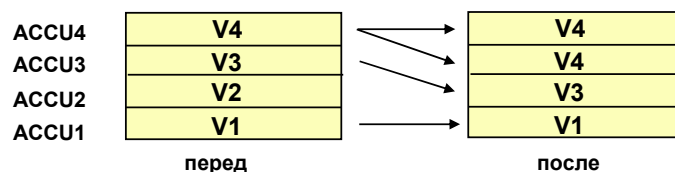
- POP (S7-300): Инструкция POP копирует содержимое ACCU2 в ACCU1. ACCU2 остается неизменным.
- POP (S7-400): Инструкция POP копирует содержимое ACCU2 в ACCU1, содержимое ACCU3 в ACCU2, и содержимое ACCU4 в ACCU3. ACCU4 остается неизменным.

Инструкции ENT и LEAVE (только для S7-400)

ENT:



LEAVE:



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.5



Information and Training Center
Knowledge for Automation

ENT

Инструкция ENT перемещает содержимое аккумуляторов 2 и 3 в следующий более высокий аккумулятор. Содержимое аккумуляторов 1 и 2 остается неизменным.

ENT можно использовать перед функцией загрузки, чтобы сохранить содержимое ACCU2:

ENT

L. ...

Инструкция выполняется независимо от битов слова состояния и не меняет биты слова состояния.

LEAVE

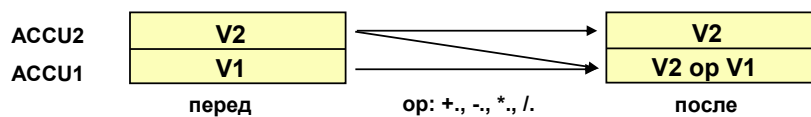
Инструкция LEAVE перемещает содержимое аккумуляторов 3 и 4 в "верхние" аккумуляторы. Содержание аккумуляторов 4 и 1 остается неизменным.

Арифметические функции содержат функциональные возможности LEAVE. С помощью инструкции LEAVE Вы можете подражать тем же самым функциональным возможностям в других цифровых логических действиях (например, побитовые логические инструкции для слова).

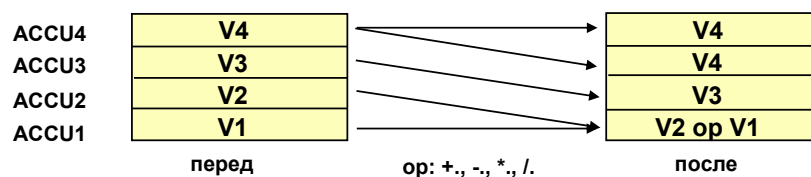
LEAVE, запрограммированная после цифрового логического действия, приводит содержимое аккумуляторов 3 и 4 в аккумуляторы 2 и 3. Результат цифрового логического действия остается неизменным в аккумуляторе 1.

Арифметические инструкции

S7-300:



S7-400:



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.6



Information and Training Center
Knowledge for Automation

Арифметические инструкции

Арифметические инструкции используют два цифровых значения, помещенные в аккумуляторах 1 и 2, согласно основным действиям арифметики. Результат вычисления помещается в ACCU1. Биты CC0, CC1, OV и OS слова состояния, дают информацию относительно результата или промежуточного результата вычисления.

S7-300

В CPU S7-300 содержание ACCU2 остается неизменным при выполнении арифметической функции.

S7-400

В CPU S7-400 содержание ACCU2 переписывается содержанием ACCU3. Содержание ACCU4 передается ACCU3.

Пример

Следующий фрагмент программы дает различные результаты, в зависимости от того, выполняется ли код на S7-300 CPU или на S7-400 CPU:

```
L 0 // загружается целое число 0 в ACCU1
L 5 // загружается целое число 5 в ACCU1, 0 - в ACCU2
PUSH // перемещается в ACCU2; (S7-400: ACCU2 -> ACCU3)
*I // умножается ACCU1 на ACCU2; (S7-400: ACCU3 -> ACCU2)
*I // умножается ACCU1 на ACCU2; (S7-400: ACCU3 -> ACCU2)
```

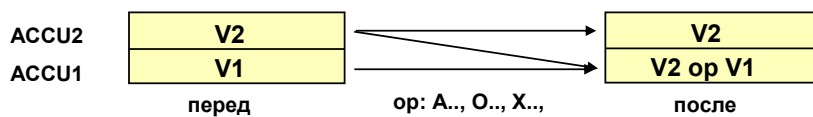
Результат:

S7-300: ACCU1 = 125

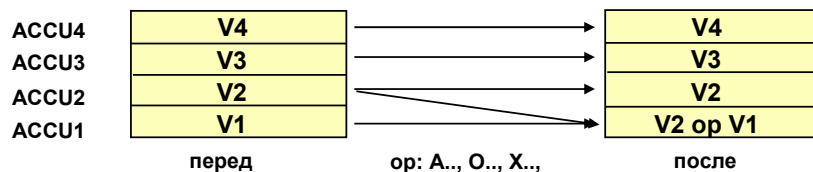
S7-400: ACCU1 = 0

Логические инструкции для слов

S7-300:



S7-400:



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.7

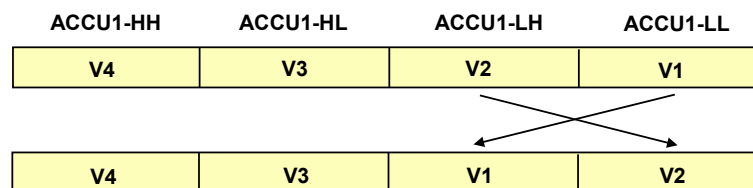
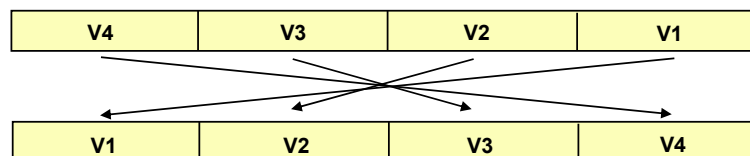


Information and Training Center
Knowledge for Automation

Логические инструкции для слов

Логические инструкции для слов комбинируют соответствующие биты из ACCU1 с битами постоянной или содержимого ACCU2, результат помещается в ACCU1. Содержимое остальных аккумуляторов (ACCU2 для S7-300, или ACCU2, ACCU3 и ACCU4 для S7-400) остается неизменным. Логическое действие может быть выполненное для слов или для двойных слов. Существуют следующие инструкции: И (AW, AD), ИЛИ (OW, OD) и ИСКЛЮЧАЮЩЕЕ ИЛИ (XOW, XOD).

Инструкции обмена байтов в ACCU1

CAW:**CAD:****SIMATIC S7**

Siemens AG 1999. All rights reserved.

 Date: 04.11.2005
 File: PRO2_02E.8

 Information and Training Center
 Knowledge for Automation
CAW

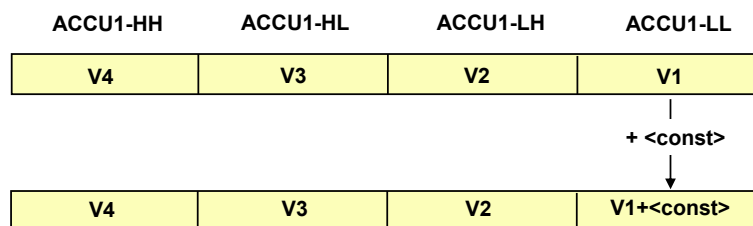
Инструкция CAW обменивает байты в правом слове данных ACCU1, то есть содержание ACCU1-L-H перемещается в ACCU1-L-L и наоборот. С помощью этой инструкции можно преобразовать формат 16-разрядных чисел (INT, и WORD) на языке программирования SIMATIC к формату чисел языков программирования для процессоров INTEL (данные для передачи в PC).

CAD

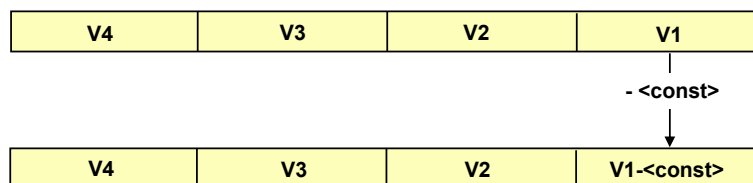
Инструкция CAD обменивает байты в ACCU1, то есть содержание ACCU1-H-H, перемещается в ACCU1-L-L и наоборот и содержимое ACCU1-H-L в ACCU1-L-H и наоборот. Этой инструкцией можно преобразовать формат 32-разрядных чисел (DINT, DWORD и REAL) на языке программирования SIMATIC к формату чисел языков программирования процессора INTEL (данные для передачи в PC).

Инструкции инкремента и декремента для ACCU1

INC <const>:



DEC <const>:



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.9



Information and Training Center
Knowledge for Automation

INC

Инструкция INC < целое число, 8-бит > прибавляет целое число 8-битовое к содержимому ACCU1-L-L и сохраняет результат в ACCU1-L-L. ACCU1-L-H, ACCU1-H, ACCU2, ACCU3, ACCU4 остаются неизменными.

DEC

Инструкция DEC < целое число, 8-бит > вычитает целое 8-битное число из содержимого ACCU1-L-L и сохраняет результат в ACCU1-L-L. ACCU1-L-H, ACCU1-H, ACCU2, ACCU3, ACCU4 остаются неизменным.

Обратите внимание Инструкции INC и DEC так называемые "инструкции низкого уровня", то есть в случае переполнения, процессор не устанавливает бит переполнения в слове состояния.

Вместо инструкции INC, Вы можете также использовать следующие инструкции для INT- или DINT- дополнения:

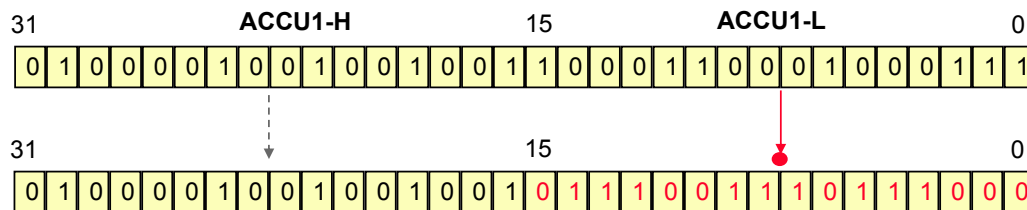
- + <const> (Прибавляет 16-разрядную константу к ACCU1)
- + L # <const> (Прибавляет 32-разрядную константу к ACCU1)

Вместо инструкции DEC, Вы можете использовать следующие инструкции для INT- или DINT- вычитания:

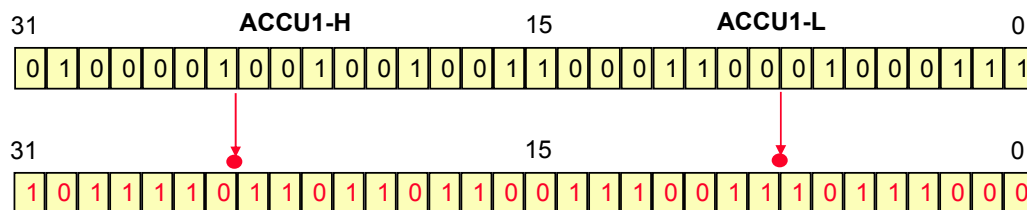
- + - <const> (Вычитает 16-разрядную константу из содержимого ACCU1)
- + L # - <const> (Вычитает 32-разрядную константу из содержимого ACCU1).

Формирование дополнений

INVI (Дополнение для ACCU1-L):



INVD (Дополнение для ACCU1):



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.10Information and Training Center
Knowledge for Automation

INVI

Инструкция INVI инвертирует биты, содержащиеся (биты от 0 до 15) в правом слове ACCU1. Т.е. заменяет нули на единицы, а единицы на нули. Содержимое левого слова (биты от 16 до 31) остается неизменным. Инструкция INVI не изменяет никакие биты слова состояния.

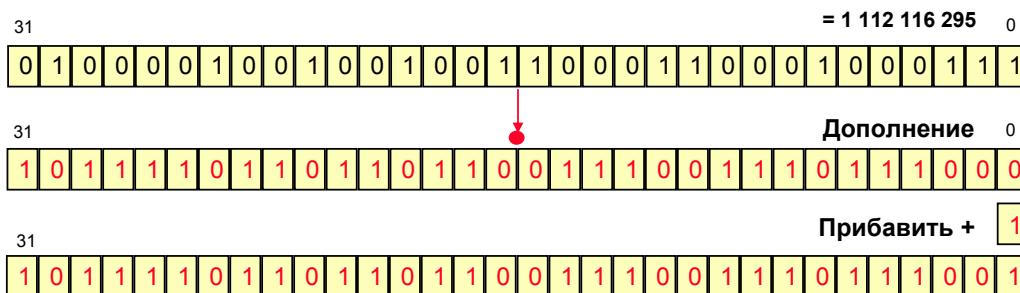
INVD

Инструкция INVD инвертирует биты, содержащиеся (биты от 0 до 31) в правом слове ACCU1. Т.е. заменяет нули на единицы, а единицы на нули. Инструкция INVD не изменяет никакие биты слова состояния.

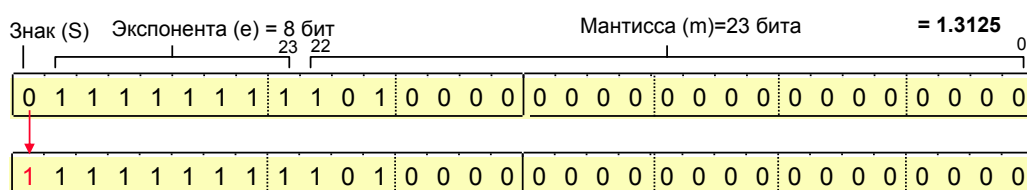
Инверсия знака (двойное дополнение)

NEGI (Инверсия знака в числе типа INT)

NEGD (Инверсия знака в числе типа DINT):



NEGR (Инверсия знака в числе типа REAL):



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.11



Information and Training Center
Knowledge for Automation

NEGI

Инструкция NEGI интерпретирует значение (биты от 0 до 15), находящееся в правом слове ACCU1, как число типа INT и формируя двойное дополнение, т.е.изменяет его знак.

Эта инструкция эквивалентна умножению на "-1". Левое слово ACCU1 (биты от 16 до 31) остается неизменным.

Биты слова состояния CC1, CC0, OS и OV устанавливаются в соответствии с результатом операции.

NEGD

Инструкция NEGD интерпретирует значение, находящееся в ACCU1 как число типа REAL и умножает его на "-1".

Формирование двойного дополнения может также быть достигнуто, формированием дополнения и прибавлением 1 (для двоичного числа, это эквивалентно вычитанию).

Биты слова состояния CC1, CC0, OS и OV устанавливаются в соответствии с результатом операции..

NEGR

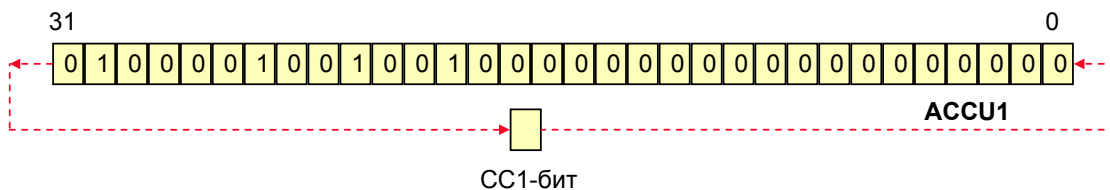
Инструкция NEGR интерпретирует значение, находящееся в ACCU1 как число типа REAL (32 бита, IEEE-FP) и умножает этот число на "-1".

Инструкция инвертирует состояние бита 31 в ACCU1 (знак мантииссы).

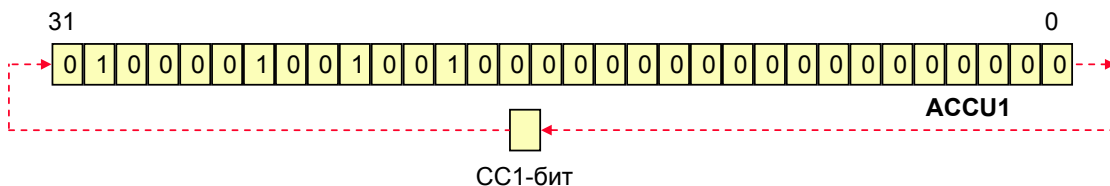
Инструкция NEGR не изменяет биты слова состояния.

Инструкции циклического 32-битного сдвига через бит CC1

RLDA (Циклический 32-битный сдвиг влево через бит CC1):



RRDA (Циклический 32-битный сдвиг вправо через бит CC1):



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.12



Information and Training Center
Knowledge for Automation

RLDA

Функция сдвига RLDA сдвигает содержимое ACCU1 на 1 бит влево. В содержимое бита (бит 0), который становится свободным во время сдвига, записывается значение бита CC1 слова состояния. Бит CC1 слова состояния получает значение вытолкнутого бита (бит 31); Биты CC0 и OV слова состояния устанавливаются в "0".

- Пример:

ACCU1: 0100 0100 1100 0100

CC1: 1

RLDA

ACCU1: 1000 1001 1000 1001

CC1: 0

RRDA

Функция сдвига RRDA сдвигает содержимое ACCU1 на 1 бит вправо. В содержимое бита (бит 31), который становится свободным во время сдвига, записывается значение бита CC1 слова состояния. Бит CC1 слова состояния получает значение вытолкнутого бита (бит 0); Биты CC0 и OV слова состояния устанавливаются в "0".

- Пример:

ACCU1: 0100 0100 1100 0100

CC1: 1

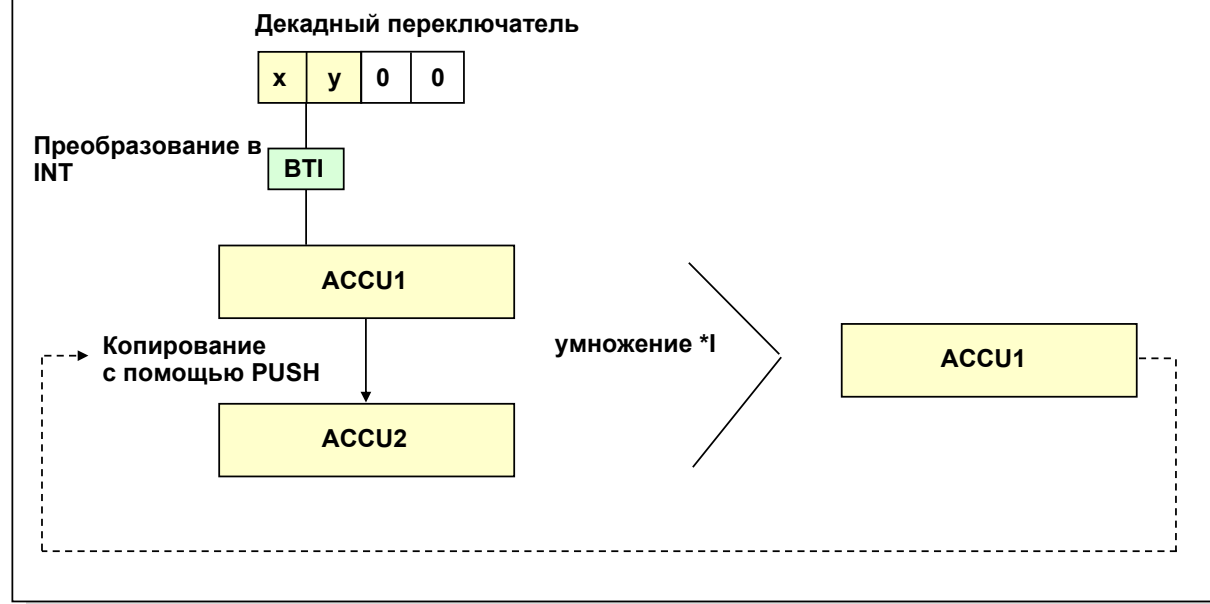
RRDA

ACCU1: 1010 0010 0110 0010

CC1: 0

Упражнение 2.1: Вычисление степени

Пример: Вычисление 6-ой степени целого числа через последовательное использование команд PUSH и *I



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.13



Information and Training Center
Knowledge for Automation

Цель упражнения Чтобы познакомиться с функциями, изменяющими ACCU, вычислим степень целого числа.

Задача Создайте FC21, выполняющую следующее:

- Прочитать в левый байт декадного переключателя и преобразовать значение из BCD-формата в формат целого числа (BTI).
- Вычислить 6-ую степень прочитанного значения:

Что делать

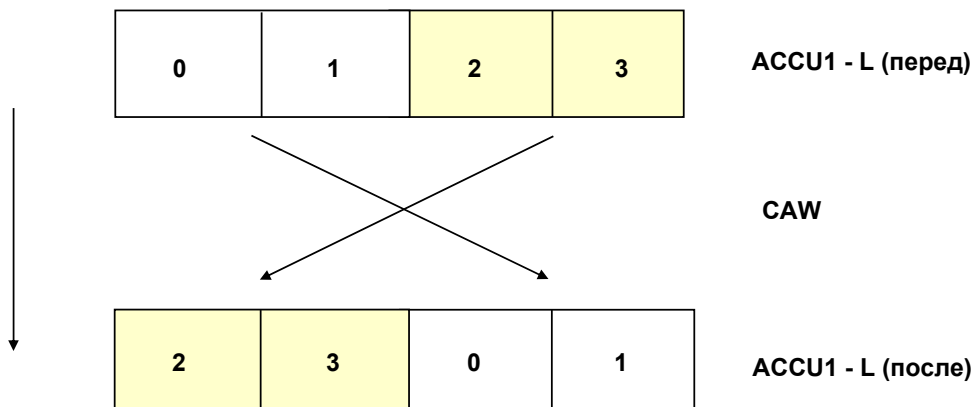
1. Копировать содержание ACCU1 в ACCU2 с помощью команды PUSH.
2. Умножить ACCU1 на ACCU2 (вычисление квадрата).
3. Копировать содержание ACCU1 в ACCU2 с помощью команды PUSH.
4. И т.д., и т.д., и т.д.

Осторожно: Как Вы должны выполнить четвертый шаг так, чтобы FC21 давала правильный результат на S7-300-CPU также как и на S7-400-CPU?

5. Показать результат на цифровом дисплее.
6. Вызвать FC21 в OB1 и загрузить программу в S7-CPU.
7. Проверить программу.

Обратите внимание Чтобы читаемое значение не стало слишком большим, Вы должны использовать только правую декаду.

Упражнение 2.2 : Обмен данных в ACCU1



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.14



Information and Training Center
Knowledge for Automation

Цель упражнения

Вы знакомитесь с инструкцией для обмена байтов в пределах ACCU1.
Применение: Преобразование представления числа на SIMATIC в представление на PC, использующем INTEL-CPU (80486, Pentium, ...). Эти преобразования должны всегда тогда выполняться, когда требуется обмен числами между SIMATIC и PC.

Задача

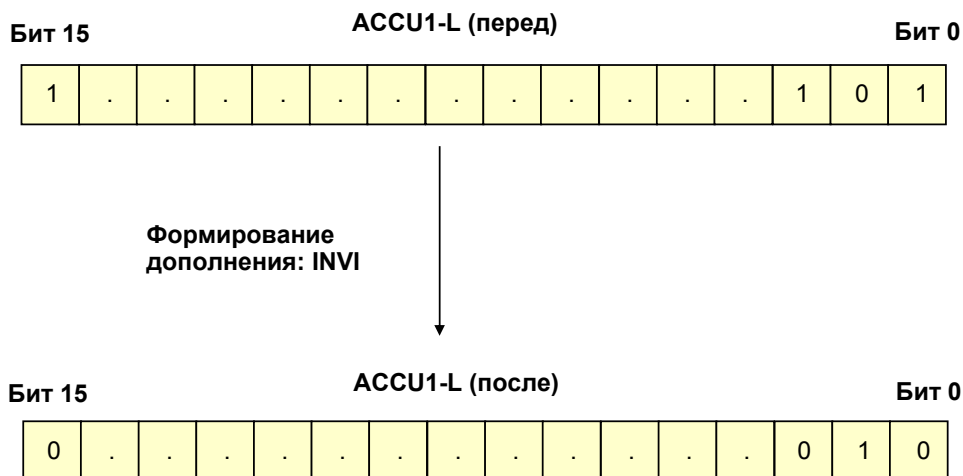
Создайте FC22 со следующими функциональными возможностями:

- Загрузка значения декадного переключателя в ACCU1.
- В ACCU1 - L обменять два байта с помощью команды CAW.
- Показывают содержание ACCU1 на цифровом дисплее.

Что делать

1. Создать FC22.
2. Вызвать FC22 в OB1.
3. Разгрузить программу в S7-CPU.
4. Проверить программу.

Упражнение 2.3 : Формирование дополнения



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_02E.15



Information and Training Center
Knowledge for Automation

Цель упражнения

Вы познакомитесь с инструкцией для формирования дополнения в SIMATIC S7.

Применение: Преобразование "отрицательной" логики в "положительную" логику

Такие преобразования предпринимаются тогда, когда используются 0-активные сигналы, однако программа пользователя продолжает работать с положительной логикой.

Задача

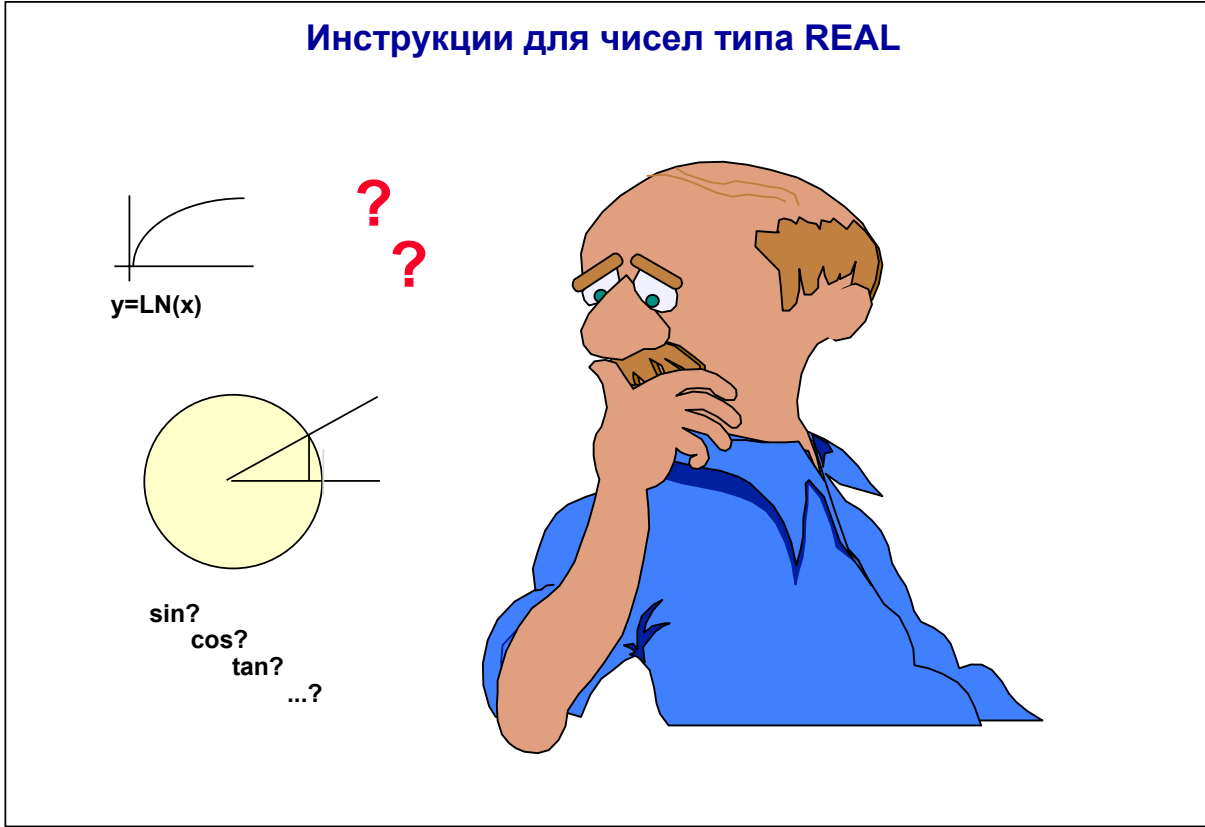
Создать FC23 со следующей функциональностью:

- Загрузить значение слова с переключателя тренажера в ACCU1.
- Формировать дополнение.
- Выходной результат вывести на светодиоды тренажера.

Что делать

1. Создать FC23
2. Вызвать FC23 в OB1
3. Загрузить программу в S7-CPU.
4. Проверить программу с помощью просмотра статуса переменных.

Инструкции для чисел типа REAL



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_03E.1

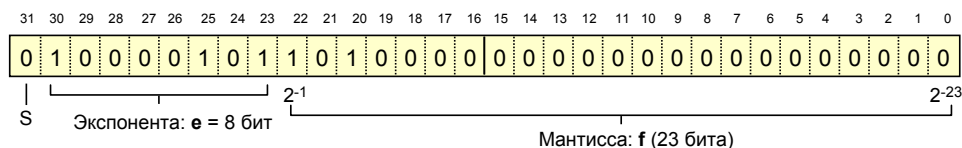


Information and Training Center
Knowledge for Automation

Содержание	Стр
Представление числа типа REAL (числа с плавающей точкой) в SIMATIC S7	2
Основные инструкции для чисел типа REAL	3
Дополнительные математические функции	4
Тригонометрические функции и обратные к ним	5
Другие инструкции для чисел типа REAL	6
Упражнение 3.1: Вычисление расстояния	7

Представление числа типа REAL (числа с плавающей точкой) в SIMATIC S7

- Формат представления числа типа REAL (IEEE FP 32-разрядный формат):



- Представление нормализованного числа типа REAL:

$$S \times (1.f) \times 2^{(e-127)}$$

S = Знаковый бит, (0 соответствует +, 1 соответствует -)

f = 23 битная мантисса с MSB = 2^{-1} и LSB = 2^{-23}

e = экспонента - двоичное целое число ($0 < e < 255$)

- Пример:

$$S = 0$$

$$e = 1000\ 0101 = 133$$

$$f = 1010\ 0000... = 0.5 + 0.125$$



$$R = +1.625 \times 2^{(133-127)} = 1.625 \times 64 = 104.0$$

- Диапазон значений числа типа REAL:

$$-3.402\ 823 \times 10^{+38} \dots -1.175\ 494 \times 10^{-38}, 0, 1.175\ 494 \times 10^{-38} \dots 3.402\ 823 \times 10^{+38}$$

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_03E.2



Information and Training Center
Knowledge for Automation

Число типа REAL

Числа типа REAL (числа с плавающей точкой) позволяют выполнять сложные математические вычисления в алгоритмах управления производственным процессом типа замкнутых контуров управления. Тип данных REAL - это описание структуры, состоящей из трех компонентов: знака, экспоненты из 8 бит и мантиссы из 23 бит. Знак может иметь значение "0" (положительный) или "1" (отрицательный). Экспонента увеличена на постоянное значение +127 для того, чтобы ее значение находилось в диапазоне от 0 до 255. Мантисса представляет собой дробную часть. Целую часть числа мантисс не содержит, так как это всегда или 1 (для нормализованного числа с плавающей точкой) или 0 (для денормализованного числа с плавающей точкой).

Ограничения

Описание	Значение e	Мантисса f	Значение	CC1	CC0	OV	OS
Нет представл.	255	$\neq 0$	[qNaN]	1	1	1	1
Переполнение	255	0	$>(2 \cdot 2^{-23}) 2^{127}$ $<(-2 \cdot 2^{-23}) 2^{127}$	1 0	0 1	1 1	1 1
Нормализ. число	1.. 254	любая	$(1.f) 2^{e-127}$ $(-1.f) 2^{e-127}$	1 0	0 1	0 0	- -
Денормализ. число	0	$\neq 0$	$(0.f) 2^{-126}$ $(-0.f) 2^{-126}$	0 0	0 0	1 1	1 1
Нуль	00	+0	00	0	-		

Обратите внимание CPU вычисляют числа с плавающей точкой с полной точностью. Показ на PG может отклоняться от точного представления, из-за ошибок округления при преобразовании. Числа типа REAL округляются до шестого знака после запятой.

Основные инструкции для чисел типа REAL

• REAL - сложение:

L	MD10	// Загрузка 1-го REAL -числа
L	MD20	// Загрузка 2-го REAL -числа
+R		// Сложение REAL -чисел (MD10 + MD20)
T	MD30	// Перенос результата в MD30

• REAL - вычитание:

L	MD10	// Загрузка 1-го REAL -числа
L	MD20	// Загрузка 2-го REAL -числа
-R		// Вычитание REAL -чисел (MD10 - MD20)
T	MD30	// Перенос результата в MD30

• REAL - умножение:

L	MD10	// Загрузка 1-го REAL -числа
L	MD20	// Загрузка 2-го REAL -числа
*R		// Умножение REAL -чисел (MD10 * MD20)
T	MD30	// Перенос результата в MD30

• REAL - деление:

L	MD10	// Загрузка 1-го REAL -числа
L	MD20	// Загрузка 2-го REAL -числа
/R		// Деление REAL -чисел (MD10 / MD20)
T	MD30	// Перенос результата в MD30

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_03E.3



Information and Training Center
Knowledge for Automation

Краткий обзор

Функции + R, -R, *R, /R интерпретирует значения, в ACCU1 и ACCU2 как числа типа REAL. Они выполняют запрограммированную арифметическую операцию (+ R, -R, *R и /R) и сохраняют результат в ACCU1. После того, как вычисление выполнено, биты CC0 и CC1 слова состояния указывают: если CC1 = 0 и CC0 = 1 - результат отрицательный; нуль - CC1 = 0 и CC0 = 0; положительный - CC1 = 1 и CC0 = 0. Биты слова состояния OV и OS сигнализируют о переполнении.

Недействительные REAL-числа

Недействительное вычисление - это когда одна из двух входных величин - недействительное REAL-число. В этом случае результат в ACCU1 - также недействительное REAL-число.

Недействительные REAL-числа также сохраняются в ACCU1, если Вы попытаете обработать недействительные значения следующими инструкциями:

Сложение: Сложение "+ бесконечности" и "- бесконечности".

Вычитание: Вычитание "+ бесконечности" и "+ бесконечности" или "- бесконечности" и "- бесконечности"

Умножение: Умножение 0 на "бесконечность"

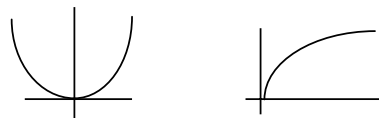
Деление: Деление "бесконечности" на "бесконечности" или 0 на 0. Результат деления действительных REAL-чисел на 0, в зависимости от знака числа равен "+ бесконечности" или "- бесконечности".

Обратите внимание 16-ичное число D # 16 # FFFF FFFF представляет собой, например, недействительное REAL -число.

Дополнительные математические функции

● Математические функции:

SQR	Вычисление квадрата
SQRT	Вычисление квадратного корня
EXP	Показательная функция по основанию e
LN	Натуральный логарифм (e=2.718282)

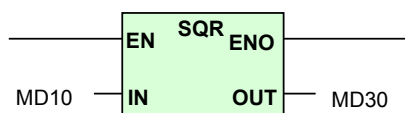


● Пример:

```

L      MD10    // Загрузка REAL-числа
SQR    // Вычисление квадрата
T      MD30    // Перенос результата в MD30
  
```

(STL)



(LAD)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_03E.4



Information and Training Center
Knowledge for Automation

Краткий обзор

Математические функции используют число из ACCU1 как входное значение, результат вновь возвращается в ACCU1.

Математические функции изменяют содержимое только ACCU1. Содержимое ACCU2, ACCU3 и ACCU4 (последние два - для S7-400), остается неизменным.

В зависимости от результата, математические функции устанавливают биты CC0, CC1, OV и OS слова состояния.

Если в ACCU1 находится недействительное REAL-число, то математическая функция возвращает недействительное REAL-число и устанавливает соответствующие биты слова состояния.

SQR

Функция SQR вычисляет квадрат содержимого ACCU1.

SQRT

Функция SQRT вычисляет квадратный корень значения в ACCU1. Если значение в ACCU1 отрицательное, SQRT устанавливает биты CC0, CC1, OV и OS слова состояния в "1" и возвращает недействительное REAL-число. Если в ACCU1 находится -0 (минус ноль), то функция возвращает также -0.

EXP

Функция EXP вычисляет степень числа e (= 2.71828), показателем служит значение из ACCU1.

LN

Функция LN вычисляет натуральный логарифм, по основанию e значения из ACCU1. Если значение в ACCU1 меньше или равно нулю, LN устанавливает биты слова состояния CC0, CC1, OV и OS в "1" и возвращает недействительное REAL-число.

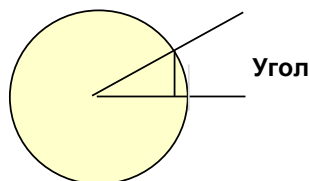
Натуральный логарифм - обратная функция к показательной функции:

Если $y = e^x$, то $x = \ln y$

Тригонометрические функции и обратные к ним

- Тригонометрические функции:

SIN	Синус
COS	Косинус
TAN	Тангенс



- Обратные к тригонометрическим:

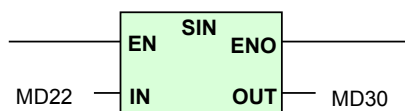
ASIN	Арксинус
ACOS	Аркосинус
ATAN	Арктангенс

- Пример:

```

L      MD10      // Загрузить REAL-число
SIN      // Вычислить синус
T      MD30      // Передать результат в MD30
  
```

(STL)



(LAD)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_03E.5



Information and Training Center
Knowledge for Automation

Тригонометрические функции

Тригонометрические функции используют REAL-значение в ACCU1 как угол в радианной мере.

Для входного угла (00 ... 3600), Вы должны, если необходимо, выполнить преобразование к радианной мере ($0 \dots 2\pi$, с $\pi = 3.141593$). Во время выполнения функции, значение, меньшее, чем 0 или большее, чем 2π , многократным прибавлением 2π или вычитанием из 2π автоматически приводится в диапазон 0 и 2π (автоматическое вычисление по модулю 2π).

Функции, обратные к тригонометрическим

Эти функции используют REAL-число в определенном диапазоне значений в ACCU1 и возвращают угол в радианной мере :

Функция	Диапазон аргумента	Диапазон результата
ASIN	$[-1, +1]$	$[-\pi/2, +\pi/2]$
ACOS	$[-1, +1]$	$[0, \pi]$
ATAN	Полный диапазон	$[-\pi/2, +\pi/2]$

При выходе за пределы разрешенного диапазона, функции возвращают действительное REAL-число и устанавливают биты слова состояния CC0 CC1, OV и OS в "1".

Другие инструкции для чисел типа REAL

- Инструкции преобразования типа REAL в DINT:

RND+	Округление до следующего большего числа типа DINT (с избытком)
RND-	Округление до ближайшего меньшего числа типа DINT (с недостатком)
RND	Округление до следующего целого числа
TRUNC	Целая часть числа

- Инструкции преобразования типа DINT в REAL:

DTR	Округление
-----	------------

- Другие инструкции преобразования типа REAL в REAL:

ABS	Вычисление абсолютного значения (модуля)
NEGR	Инверсия знака REAL-числа (умножение на -1)

- Пример:

L	MD10	// Загрузить REAL-число	
RND+		// Преобразовать в следующее большее	(STL)
		//DINT-число	
T	MD30	// Перенести результат в MD30	

	EN	RND+	ENO	
MD22	IN	OUT		MD30

(LAD)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_03E.6Information and Training Center
Knowledge for Automation

Краткий обзор

Функции преобразования конвертируют тип данных значения, находящегося в ACCU1 в другие типы данных, и сохраняют результат в ACCU1. Содержание других ACCU остается неизменным.

Если, в одной из инструкций (RND +, RND -, RND или TRUNC), значение, находящееся в ACCU1, выходит за границы допустимого диапазона типа DINT или не соответствует REAL-числу, инструкция устанавливает биты слова состояния OV и OS в "1". Преобразование тогда не происходит.

RND +

Инструкция RND + преобразовывает содержимое ACCU1 как REAL-число в целое число (DINT), который является большим или равным числу, которое конвертируется.

MD10 = " 100.5" => RND + => MD20 = " + 101 "

MD10 = " -100.5" => RND + => MD20 = " -100 "

RND-

Инструкция RND- конвертирует содержимое ACCU1 как REAL-число в целое число (DINT), который является меньшим или равным числу, которое конвертируется.

MD10 = " 100.5" => RND- > = MD20 = " + 100 "

MD10 = " -100.5" => RND- > = MD20 = " -101 "

RND

Инструкция RND преобразовывает содержимое ACCU1 как REAL-число в следующее возможное целое число (DINT).

MD10 = " 100.3" => RND > = MD20 = " + 100 "

MD10 = " 100.7" => RND > = MD20 = " + 101 "

MD10 = " -100.3" => RND > = MD20 = " -100 "

MD10 = " -100.7" => RND > = MD20 = " -101 "

TRUNC

Инструкция TRUNC возвращает целую часть числа; дробная компонента компонент отбрасывается.

MD10 = " 100.5" => TRUNC > = MD20 = " + 100 "

MD10 = " -100.5" => TRUNC > = MD20 = " -100 "

DTR

Инструкция DTR преобразовывает число из формата DINT в формат REAL. Если необходимо, операция округляет результат.

ABS

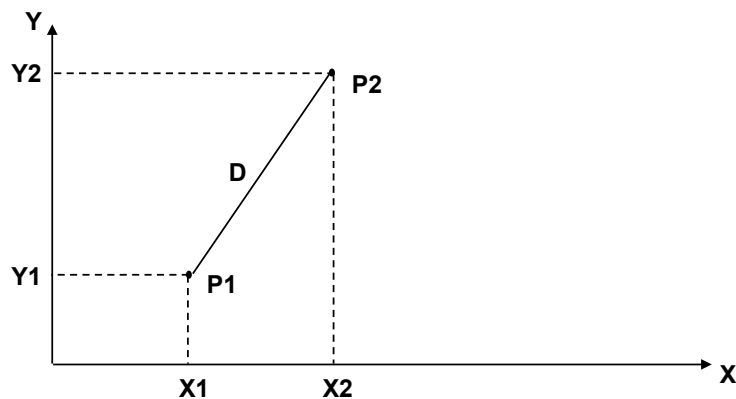
Инструкция ABS формирует абсолютное значение REAL-числа, находящегося в ACCU1, то есть устанавливает в "0" знак (бит 31) (даже для действительного REAL-числа).

NEGR

Инструкция NEGR инвертирует REAL-число в ACCU1, то есть инвертирует знак (бит 31) (даже для действительного REAL-числа). Инструкции DTR, ABS и NEGR не изменяют биты слова состояния.

Упражнение 3.1: Вычисление расстояния

Пример: Вычисление расстояния D между двумя точками в прямоугольной системе координат



Функция: FC31 с $D = \sqrt{(X2 - X1)^2 + (Y2 - Y1)^2}$

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_03E.7



Information and Training Center
Knowledge for Automation

Цель упражнения Применение математических функций для вычисления расстояния между двумя точками.

Задача Создайте FC31 со следующими функциональными возможностями:

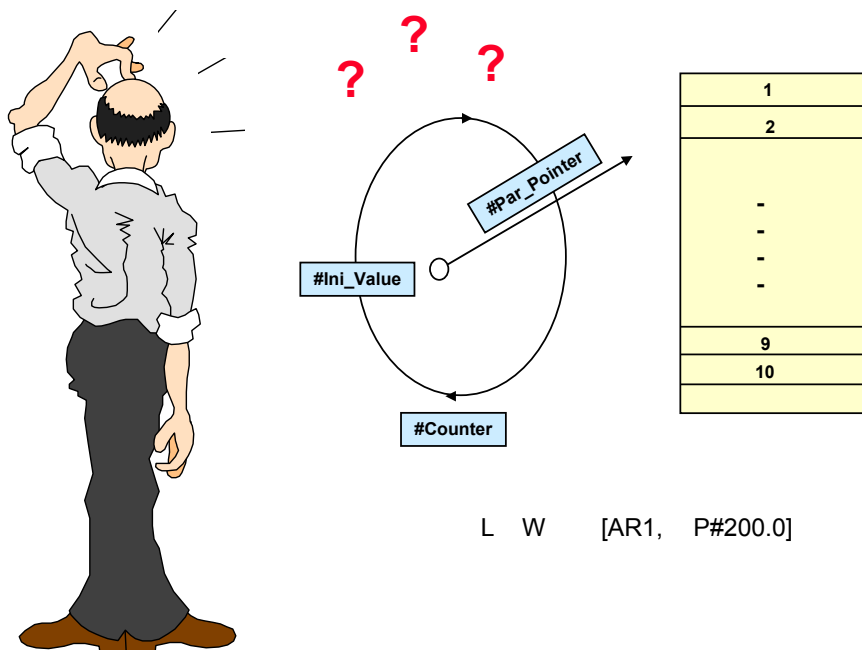
- FC31 принимает координаты (X1, Y1) и (X2, Y2) двух точек P1 и P2 во входных параметрах.
- FC31 возвращает расстояние между точками в выходном параметре RET_VAL.
- FC31 должна быть выполняемой как в системе S7-300, так и в системе S7-400. Она не должна использовать глобальные адреса CPU для сохранения промежуточных результатов.

Что делать

1. Создать FC31 с вышеупомянутыми функциональными возможностями.
2. Вызвать FC31 из OB1, и назначить ей входные и выходные параметры следующим образом:
 $X1 = MD0$, $Y1 = MD4$
 $X2 = MD8$, $Y2 = MD12$
 $RET_VAL = MD16$
3. Разгрузить программу в S7-CPU.
4. Проверить FC31 с помощью утилиты "Monitor/Modify Variable".

Дополнительная задача Создать оптимальную по времени выполнения версию FC31 для S7-400, которая работает без использования временных переменных.

Косвенная адресация и инструкции с адресными регистрами



SIMATIC S7

Siemens AG 1999. All rights reserved.

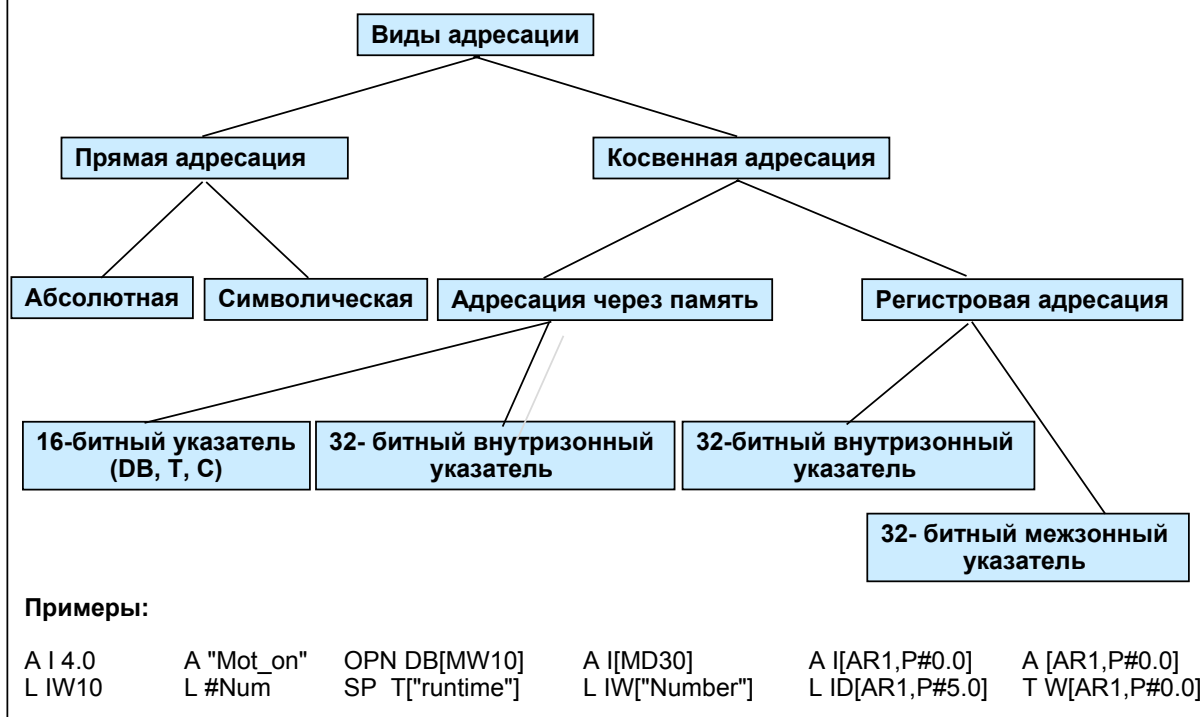
Date: 04.11.2005
File: PRO2_04E.1Information and Training Center
Knowledge for Automation

Содержание

Стр

Виды адресации, доступные в STEP 7	2
Прямая адресация переменных	3
Адресные идентификаторы прямой адресации для DB	4
Оценка информации о DB в программе	5
Косвенная адресация через память	6
Структура указателя при косвенной адресации через память	7
Специальные особенности косвенной адресации через память	8
Пример косвенной адресации	9
Упражнение 4.1: Программирование цикла с косвенной адресацией	10
Внутризонная регистровая косвенная адресация	11
Межзональная регистровая косвенная адресация	12
Инструкции для загрузки адресных регистров	13
Другие инструкции для адресных регистров	14
Специальные особенности адресных регистров	15
Упражнение 4.2: Программирование цикла с регистровой косвенной адресацией	16
Типы указателей в STEP 7	17
Структура и назначение типа данных POINTER	18
Структура типа данных ANY	19
Назначение параметров с типом данных ANY	20
Косвенное назначение параметра типа ANY	21
Использование переданного указателя ANY	22
Упражнение 4.3: Функция вычисления суммы и среднего значения	23

Виды адресации, доступные в STEP 7



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.2Information and Training Center
Knowledge for Automation

Прямая адресация

При прямой адресации местоположение операнда в памяти закодировано в инструкции, то есть указан идентификатор адреса, определяющий адрес величины, с которой инструкция должна работать.

Символическая адресация

В программе управления, адреса могут быть указаны абсолютные адреса (например, I 1.0) или символические (например "start signal"). Символический адрес использует имена вместо абсолютных адресов. Программа более легка для чтения, когда используются значения имен. Символические адреса подразделяются на локальные символы (определяются в части декларации блока) и глобальные символы (определяются в символической таблице).

Косвенная адресация

С помощью косвенной адресации Вы можете адресовать идентификаторы адреса которых определяются только при выполнении программы. С помощью косвенной адресации, часть программы, например, может быть выполнена неоднократно (программирование цикла), а используемые адреса назначаются в каждом проходе различными.

Косвенная адресация подразделяется на:

- Косвенную адресацию через память: указатель на адрес находится в ячейке памяти пользователя (например, MD30).

Ячейкам памяти, где хранятся указатели можно давать символические имена.

- Регистровую косвенную адресацию: указатель на адрес загружен в одном из двух адресных регистраторов (AR1 или AR2) S7- процессора.

Предостережение

Так как при косвенной адресации адрес вычисляется только во время выполнения, существует опасность, что могут быть неумышленно испорчены области памяти и, таким образом, произойти неожиданная реакция PLC.

Прямая адресация переменных

Адрес	Местоположение в памяти (например).	Ширина доступа	Значение
I	37.4	Байт, слово, двойное слово	Входы
Q	27.7	Байт, слово, двойное слово	Выходы
PIB	655	Байт, слово, двойное слово	Периферийные входы
PQB	653	Байт, слово, двойное слово	Периферийные выходы
M	55.0	Байт, слово, двойное слово	Меркеры
T	114	--	Таймеры
C	13	--	Счетчики
DBX	2001.6	Байт (DBB), слово (DBW), двойное слово (DBD)	Данные адресуются через DB регистр
DIX	406.1	Байт (DIB), слово (DIW), двойное слово (DID)	Данные адресуются через DI регистр
L	88.5	Байт (LB), слово (LW), двойное слово (LD)	Локальный стек

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.3Information and Training Center
Knowledge for Automation

Прямая адресация переменных

С помощью прямой адресации могут быть адресованы простые (элементарных) переменных, то есть переменные длиной максимум до 4 байтов.

Простые переменные состоят из:

- Идентификатора адреса (например: "IB" для байта входа)
- Точного адреса (местоположение в памяти) (адрес байта или бита) в пределах области памяти, которая определена идентификатором адреса. Адреса или простые переменные могут также быть адресованы через глобальные, символические имена (таблица символов).

Периферия

При доступе к периферии теперь, в отличие от S5, необходимо делать различие между входами и выходами. Однако, возможен доступ только на чтение (L PIW) для периферийных входов и только на запись (T PQW) для периферийных выходов.

Локальные данные

STEP 7 позволяет иметь абсолютный доступ к собственным данным локального стека блока, например:

- A L 12.6 (Опрос локального бита данных с адресом 12.6)
- L LW 12 (Загрузка локального слова данных в ACCU1)

DBX/DIX

Возможен также непосредственный доступ к простым переменным в пределах блоков данных:

- DBX 12.6 (Опрос бита данных с адресом 12.6 из DB 1, DB должен быть открыт заранее).
- L DB5. DBW10 (Загружают DW10 из DB5)

Комплексные переменные

Вы можете получить доступ к местным переменным, которые имеют сложный тип данных: структура или массив, но только символически. Абсолютный доступ возможен только с компонентами сложных переменных, которые имеют элементарный тип данных.

Адресные идентификаторы прямой адресации для DB

Открыть блок данных

Загрузка и перенос в блоках данных

OPN DB 19	L DBB 1	Загрузить байт данных 1
OPN "Values"	L DBW 2	Загрузить слово данных 2 (байты 2 и 3)
	L 5	Загрузить число 5
	T DBW 4	Перенести в слово 4
OPN DI 20	L 'A'	Загрузить ASCII-символ A
	L DIB28	Загрузить байт данных 28
	==I	Сравнить
	A DBX 0.0	Опросить бит 0 из байта 0
<hr/>		
Комбинация инструкций (содержит OPN DB..)	L DB19.DBW4	Загрузить слово данных 4 из DB 19
	L "Values".Number_1	Символический доступ к переменной Number_1. DB19, имеющей символьное имя "Values"
	A DB10.DBX4.7	Опросить бит 7 из байта 4 DB 10

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.4Information and Training Center
Knowledge for Automation

Краткий обзор

CPU имеет два регистра блока данных для обработки адреса данных. Открываются блоки данных, номера которых в настоящее время находятся в этих регистрах.

Прежде, чем Вы получите доступ к блоку данных, Вы должны сначала открыть его через один из этих двух регистров.

Вы можете открыть блок данных, используя следующие инструкции:

- OPN DBn или OPN DI n

или с помощью объединенной команды типа:

- L DBn.DBWm (L DI n.DIWm не возможна!)

В этом случае номер блока данных n, также загружается в регистр DB.

Адресация

Блоки данных в STEP7 имеют байтовую организацию. Прямой доступ возможен для переменных с длиной BIT, BYTE, WORD или DWORD; в адресе в каждом случае указывается номер первого байта (как с I/Q/M).

Символический доступ

При символическом доступе Вы вносите в список символов символическое имя блока данных.

Вы назначаете символические имена в блоке данных индивидуальным переменным, используя DB- редактор.

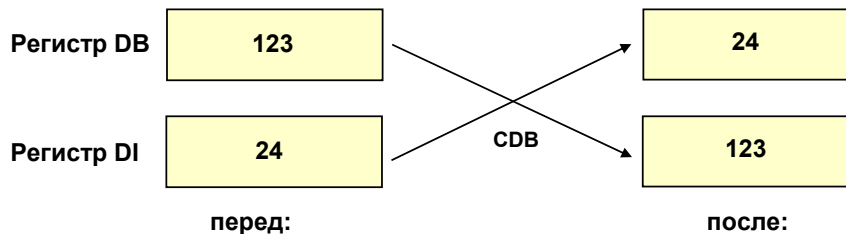
Теперь полный символический доступ элемента данных возможен с инструкцией L "Values". Number_1. Здесь открыт DB19 ("Values" - символическое имя DB 19) и загружено DW 2 (Number_1 -символическое имя DW2), т.е. это эквивалентно L DB19.DBW2.

Заметьте, что нельзя смешивать в языках LAD, FBD, STL абсолютную и символьную адресации. Например, команда L DB19.Number_1 недопустима!

Оценка информации о DB в программе

Инструкции с регистрами DB:

- **CDB: Обмен содержимого DB - регистров**



- **Загрузить DB-регистр в ACCU1**
 - L DBNO (загрузить номер открытого DB в ACCU1)
 - L DINO (загрузить номер открытого DI в ACCU1)
- **Загрузить длину блока данных**
 - L DBLG (загрузить длину (в байтах) блока данных, открытого через DB, в ACCU1)
 - L DILG (загрузить длину (в байтах) блока данных, открытого через DI, в ACCU1)

SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2 04E.5

Information and Training Center
Knowledge for Automation

Регистры DB, DI

Эти регистры содержат текущие значения номеров открытых блоков данных. Одновременно для запросов могут быть открыты два блока данных.

STL использует первый регистр DB обычно для открытия глобального DB и второй регистр DB - для открытия экземпляра DB. По этой причине эти регистры также называются регистром DB и регистром DI (instance). CPU обращается с этими регистраторами одинаково. Каждый блок данных может быть открыт, используя один из этих двух регистров (даже через оба одновременно).

CDB

CDB (Exchange DB registers) обменивает содержание регистров DI и DB. Содержание DB регистра перемещается в регистр DI и наоборот. Эта инструкция не воздействует ни на содержание ACCU1, ни биты слова состояния.

L DBLG, L DILG:

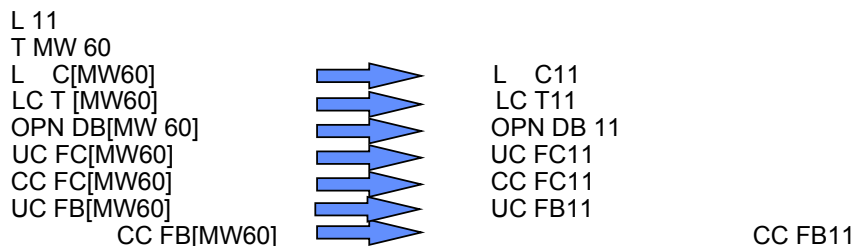
Эти инструкции загружают в ACCU1 длину блока данных, открытого в настоящее время, в байтах . С помощью этой информации, программа пользователя может проверить, имеет ли DB необходимую длину прежде, чем обратиться к DB.

L DBNO, L DINO:

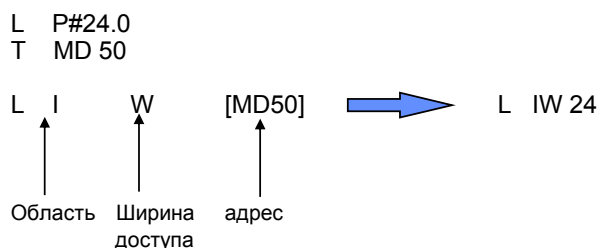
Эти инструкции загружают в ACCU1 номера открытых в настоящее время блоков данных.

Косвенная адресация через память

- **16-битный указатель в формате слова (адресация DB, T, C)**



- **32-битный указатель в формате двойного слова (адресация I, Q, M, ...)**



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.6Information and Training Center
Knowledge for Automation

Краткий обзор При косвенной адресации через память, адрес переменной, которая должна использоваться в команде, находится в ячейке памяти, которая указана в команде. Команды программы, которые используют косвенную адресацию памяти, содержат:

- Инструкцию (например: OPN, A, L, и т.д.)
- Идентификатор адреса (DB, C, T, я, QW, MD, и т.д.)
- Переменную, которая должна быть заключена в квадратные скобки.

Эта переменная содержит адрес (указатель) операнда который использует инструкция.

В зависимости от используемого идентификатора адреса, инструкция интерпретирует данные, находящиеся в переменной, заключенной в [], или как указатель в форме слова или в форме двойного слова.

Инструкции с 16-битовыми указателями Вы можете использовать 16-битовый указатель для адресации таймеров, счетчиков или блоков (DB, FC, FB).

Все инструкции для таймеров и счетчиков могут использовать косвенную адресацию. При адресации таймеров, счетчиков или блоков идентификаторы области используют формы T, C, DB, DI, FB, FC. Адрес (местоположение в памяти) адресованного операнда сохраняется в слове. Блок данных может быть открыт через регистры DB или DI. Если в указателе находится нуль, когда Вы косвенно открываете блок данных (DB, DI), то в регистр DB/DI загружается значение "0". Ошибка при такой загрузке не возникает. Вызов логических блоков может быть осуществлен при косвенной адресации с помощью инструкций UC или CC (не CALL !). Вызываемые так блоки не должны содержать никаких параметров или статических переменных. Этот указатель в формате слова интерпретируется как целое число (0 ... 65 535). Это относится к номерам таймера (T), счетчика (C), блока данных (DB, DI) или логического блока (FC, FB).

Специальные особенности косвенной адресации через память

Области памяти для сохранения 16- и 32-битовых указателей:

- Меркеры (адресуются абсолютно или символически, напр.: OPN DB[MW30], OPN DI["Motor_1"], и т.д.
A I[MD30], T QD["Speed_1"], и т.д.)
- Локальный стек данных (адресуются абсолютно или символически, напр.: OPN DB[LW10], OPN DI[#DB_NO], и т.д.
A I[LD10], T QD[#Par_Pointer], и т.д.)
- Глобальный (общий) блок данных (адресация может быть только абсолютной, DB должен быть предварительно открыт, напр.: OPN DB[DBW0] (переписывается регистр DB !!!), OPN DI[DBW22], напр.: A I[DBD10], T QD[DBD22], и т.д.)
- Экземпляр блока данных (адресация может быть только абсолютной, DI должен быть предварительно открыт, напр.: OPN DB[DIW20], OPN DI[DIW0] (переписывается регистр DI !!!), напр.: A I[DID10], T QD[DID22], и т.д.)

Характеристики в передаче указателей для FB и FC

- ❑ Указатели, используемые в параметрах, не могут использоваться непосредственно для косвенной адресации через память.
- ❑ Указатели для косвенной адресации, помещенные в память, перед вызовом должны быть скопированы во временные переменные.

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.8



Information and Training Center
Knowledge for Automation

Области памяти для хранения указателей

При косвенной адресации через память, адрес (местоположение в памяти) может быть 16-разрядным или 32-разрядным. Этот адрес может сохраняться в одной из следующих областей:

- Память меркеров: как абсолютно адресованный операнд или как переменная, символически адресованная через таблицу символов.
- Локальный стек данных: как абсолютно адресованный операнд или как временная переменная, объявленная в секции деклараций блока
- Глобальный (общий) блок данных: как абсолютно адресованный операнд. Когда глобальный (общий) DB используются как место для хранения указателей, должно быть учтено, что "правильный" блок данных должен быть открыт через регистр DB (например OPN DBn) перед вызовом
- Экземпляр блока данных: как абсолютно адресованный операнд. Когда данные экземпляра используются, следующие моменты должны быть соблюдены:

ОВ и функции: В пределах функций или ОВ, указатель, который сохранен в экземпляре блока данных, может использоваться, как будто он был сохранен в глобальном (общем) DB. Нужно просто помнить, что вместо регистра DB теперь используется регистр DI.

FB: В пределах FB, данных экземпляра, который являются параметрами или статические переменными, не могут вообще использоваться символически для косвенной адресации через память.

Для абсолютного доступа к локальным данным в пределах FB, в принципе, возможно использование "адреса", введенного в секцию деклараций, однако, это нужно делать, когда в FB используются мультиэкземпляры и этот адрес - не абсолютный адрес, как в случае DB, а фактически адрес относительно AR2.

Обратите внимание

Когда Вы передаете указатели для косвенной адресации через память в блоки или держите значение постоянно в статической переменной, тогда Вы должны копировать значение указателя из параметра или статической переменной во временную переменную и затем организовывать доступ, используя эту временную переменную.

Пример косвенной адресации

FC30: Пример для косвенной адресации

Network 1: Открыть DB с помощью косвенной адресации

```
L   #dbnumber           // Скопировать номер DB в MW100
T   MW 100              //
OPN DB[MW 100]          // Открыть DB
```

Network 2: Цикл удаления

```
L   P#18.0              // Сохранить конечный адрес (DBW18) как указатель
T   MD 40               // в MD 40;
L   10                  // Установить счетчик цикла на 10
next: T MB 50            // и сохранить его в MB 50;
L   0                   // Загрузить инициализирующее значение
T   DBW[MD 40]          // и перенести его в DB;
L   MD 40               // Загрузить указатель,
L   P#2.0               // уменьшить его на 2 байта
-D                          // и перенести результат назад
T   MD 40               // в MD 40;
L   MB 50               // Загрузить счетчик цикла
LOOP next                // Уменьшение счетчика и
                        // если, если он не равен 0, то переход;
```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.9



Information and Training Center
Knowledge for Automation

Описание

Этот пример показывает функцию для инициализации элементов блока данных с значением "0". Номер DB передается функции во входном параметре.

Адресованный блок данных открывается в сети 1. Для этого, переданный номер блока (входной параметр: #dbnumber) копируется в слово памяти (MW100) и затем DB открывается, используя это слово памяти.

В сети 2 первые 10 слов данных DB установлены в "0" с помощью цикла. Цикл использует инструкцию LOOP, сохраняя счетчик цикла в MB50.

Передача значения "0" в отдельные слова блока данных имеет место с помощью косвенной адресации через память (MD 40). Перед входом в цикл указатель с адресом последнего слова данных (DBW 18) загружен в MD 40. При каждом выполнении цикла, адрес доступа в MD40 уменьшается на P#2.0, так как значение передается в DB словами, а байтам.

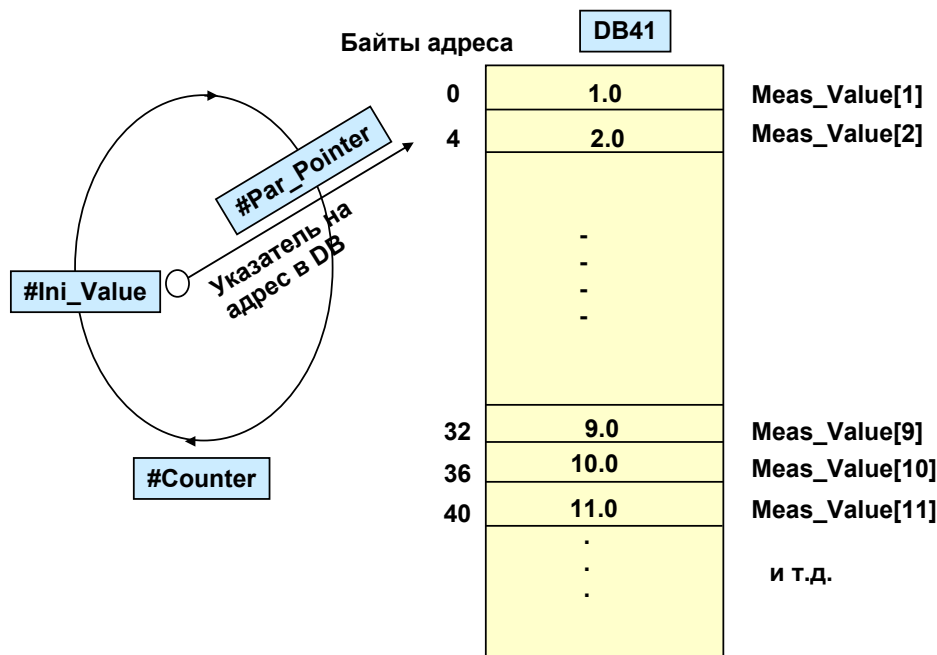
Обратите внимание Чтобы сделать пример коротким, проверка номера блока данных не выполняется.

Практически это также имело бы смысл сделать, как и спроектировать начальный адрес и длину области инициализации, параметризуемыми и проверять перед открытием DB, если даже существует DB с необходимой длиной.

Дополнительная слабость в вышеупомянутом примере - то, что указатель для косвенного доступа сохраняется в области памяти маркеров.

Что было бы лучше здесь сделать? Почему?

Упражнение 4.1: Программирование цикла с косвенной адресацией



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.10Information and Training Center
Knowledge for Automation

Цель упражнения Вы знакомитесь с использованием косвенной адресации через память в цикле, делая практический пример.

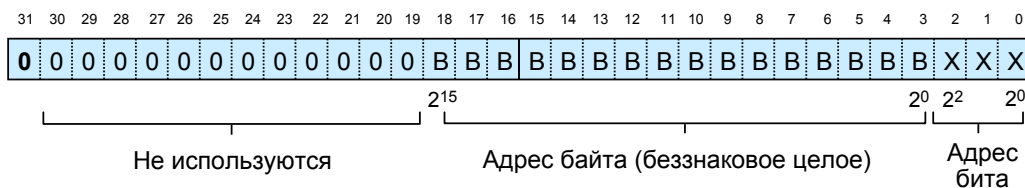
Задача

Используется косвенная адресация через память для программирования цикла. Нужно проинициализировать 100 последовательных ячеек в блоке данных значениями, возрастающими от 1.0 до 100.0.

1. Создать FC41 и DB41.
2. В части декларации DB41, определите переменную #Meas_Value типа ARRAY [1 .. 100] REAL.
3. В части декларации FC41, определите входной параметр #DB_NUM типа WORD и четыре временных переменных # L_Counter типа INT, #Ini_Value типа REAL, # I_DB_Num типа WORD, а также #Par_Pointer типа DWORD.
4. В FC41, сначала откройте блок данных, чей номер передается на вход # DB_NUM. Используйте для этого временную переменную # I_DB_NUM.
5. Заполните поля массива от # Meas_Value [1] до # Meas_Value [100] в DB41 в порядке возрастания номеров числами от 1.0 до 100.0. Используйте для этого программирование цикла (инструкция: LOOP):
 - Сохраняйте счетчик цикла в переменной # L_Counter, а инициализирующее значение для индивидуальных компонентов Meas_Value[.] в переменной # Ini_Value.
 - Используйте косвенную адресацию через память для адресации компонентов # Meas_Value [..]. Сохраняйте адрес для доступа в переменной # Par_Pointer.
6. Вызовите FC41 в OB1, и назначьте фактическое значение для входного параметра # DB_NUM. Загрузите блоки в CPU и испытайте Вашу программу.

Внутризонная регистровая косвенная адресация

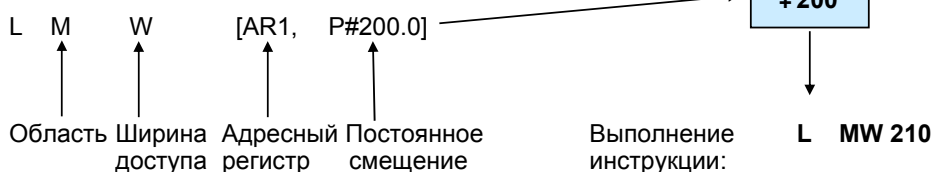
Внутризонный указатель в AR 1 или AR2:



Синтаксис команды:

LAR1 P#10.0

AR1: 00000000 0000 0000 0000 0101 0000



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.11



Information and Training Center
Knowledge for Automation

Краткий обзор

При косвенной регистровой внутризонной адресации, адрес (местоположение в памяти) операнда, к которому нужно обращаться, находится в одном из двух адресных регистров (AR1, AR2). Адресные регистры в этом случае содержат 32-битовый указатель с той же самой и тем же самой структурой, что и при косвенной адресации через память.

Синтаксис

При регистровой косвенной внутризонной адресации команда состоит из:

- Инструкции (например: A, L, T, и т.д.)
- Идентификатора адреса (I, MB, QD, и т.д.), который является комбинацией идентификатора области (I, Q, M., DB, DI, и т.д.) и ширины доступа (B = Byte, W = WORD, D = DWORD).
- Адресного регистра, который наряду с постоянным смещением должен быть указан в квадратных скобках. Это смещение добавляется к содержанию указанного адресного регистра прежде, чем инструкция выполнится.

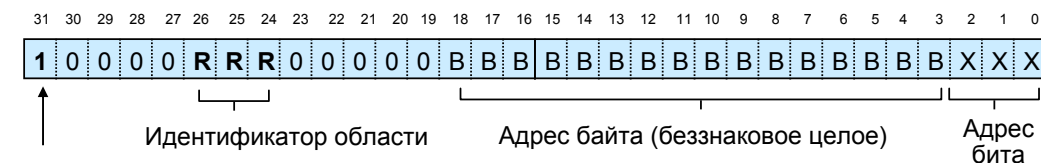
Содержание адресного регистра и смещения имеет формат внутризонных указателей, состоящих из адреса бита и адреса бита. Указание смещения (даже P#0.0) в синтаксисе команды обязательно.

Обратите внимание

- При косвенной адресации бита, слова или двойного слова, смещение должно иметь адрес бита "0", иначе возникнет ошибка времени выполнения.
- Если адресные регистры AR1 или AR2, используемые для косвенной внутризонной адресации содержат указатель с указанием области (см. следующую страницу), то идентификатор области в указателе во время выполнения инструкции не оценивается. Имеет силу идентификатор области в команде.

Межзонная регистровая косвенная адресация

Межзонный указатель в AR 1 или AR2:



Bit 31=0: внутризонная
Bit 31=1: межзонная

Идентификатор области:

000	Периферия (P)	001	Входы (PII)
010	Выходы (PIQ)	011	Память меркеров
100	Блок данных, регистр DB	101	Блок данных, регистр DI
110	Собственные локальные данные	111	Локальные данные вызывающего блока

Синтаксис команды:

LAR1 P#110.0

L W [AR1, P#200.0]

↑ ↑ ↑

Ширина Адресный Постоянное

доступа регистр смещение

AR1: 10000011 0000 0000 0000 0000 0101 0000

M

Выполнение
инструкции:

+ 200

L MW 210

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.12



Information and Training Center
Knowledge for Automation

Краткий обзор

При косвенной регистровой межзонной адресации, идентификатор области (I, Q, M, и т.д.) и адрес (местоположение в памяти) (byte.bit адрес) операнда, к которому должно происходить обращение, находится как указатель с указанием области в одном из двух адресных регистрах (AR1, AR2).

Синтаксис

При регистровой косвенной межзонной адресации, полная инструкция состоит из:

- инструкции (например: A, L, T, и т.д.)
- ширины доступа (B = БАЙТ, W = СЛОВО, D = DWORD).
- адресного регистра, который наряду с постоянным смещением должен быть указан в квадратных скобках.

Адресный регистр в этом случае должен содержать указатель с идентификатором области и адресом byte.bit, т.е. быть межзонным. Смещение имеет формат внутризонного указателя, состоящего из адреса байта и адреса бита.

Указание смещения (даже P#0.0) в синтаксисе команды обязательно.

Обратите внимание

- При косвенной адресации байта, слова или двойного слова, смещение должно иметь адрес бита "0", иначе произойдет ошибка времени выполнения.
- Доступ к собственным локальным данным (идентификатор: 110) не возможен с косвенной межзонной адресацией. Вызывается ошибка времени выполнения "unknown area identifier (неизвестный идентификатор области)".
- Доступ к собственным локальным данным возможен только с помощью внутризонной адресации.

Инструкции для загрузки адресных регистров

Загрузка адресных регистров

- | | |
|----------------------|--------------------------------------|
| • LARn (n = 1 or 2): | Загрузить содержимое ACCU1 в ARn |
| • LARn <Address> | Загрузить содержимое <Address> в ARn |
| • LARn P#<Address> | Загрузить адрес <Address> в ARn |

<Address>:

- | | |
|---|--|
| • Регистры процессора: | AR1, AR2 (напр., <i>LAR1 AR2</i> and <i>LAR2 AR1</i>) |
| • 32-битовые переменные: | MDn, LDn, DBDn, DIDn (напр., <i>L DBD5</i> , и т.д.) |
| • символн. 32- битовые переменные :
(глобальные и локальные) | 32- битовые глобальные переменные (напр., <i>LAR1 "Index"</i> , и т.д.)
и TEMP (временные) переменные OB, FB и FC
(напр., <i>LAR1 #Address</i> , и т.д.) |

P#<Address>

- | | |
|--|--|
| • Указатель с абсолютной битовой адресацией: | En.m, An.m, Mn.m, Ln.m, DBXn.m, DIXn.m
(напр., <i>LAR1 P#M5.3</i> , <i>LAR2 P#I3.6</i> , и т.д.) |
| • Указатель с локальной, символн. адресацией | <u>OB</u> : TEMP- переменные (напр.,: <i>LAR1 P##Par_Pointer</i> , и т.д.)
<u>FB</u> : IN-, OUT-, INOUT-, STAT- и TEMP- переменные.
<u>FC</u> : TEMP- переменные (<i>LAR1 P##Loop</i> , и т.д.) |

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.13Information and Training Center
Knowledge for Automation

Загрузка операндов

С помощью функций загрузки можно инициализировать адресные регистры новыми значениями. Функция загрузки LARn (n = 1, 2) загружает указатель в адресный регистр ARn. Как источник могут использоваться ACCU1, ARn или двойное слово из памяти меркеров, локального стека, глобальных (общих) блоков данных и экземпляров блока данных. Доступ может абсолютным или символическим. Если Вы не определяете адрес, в адресный регистр ARn автоматически загружается содержание ACCU1. Содержание загруженного регистра или двойного слова должно быть в формате указателя.

Загрузка указателей

Непосредственно указатели (адреса) также могут быть загружены в адресные регистры. С помощью инструкции:

- L P # < идентификатор области > n.m

Вы можете загружать указатель с указанием области непосредственно в указанный адресный регистр. Указатель с указанием области на локальную переменную (#Address) может, например, быть загружен с помощью следующей инструкции

- LARN P ## Address (n = 1, 2)

в один из двух адресных регистров. Указатель с указанием области, который создан таким образом, содержит начальный адрес переменной.

Этот доступ возможен на всех TEMP- переменных OB, FB и FC, а также для параметров с декларацией IN, OUT и INOUT и STAT- переменных FB.

Обратите внимание

Если Вы хотите загрузить указатели на IN, OUT и INOUT параметра (# Param) FC в адресный регистр, то это не возможно сделать прямым способом. Должен быть сделан промежуточный шаг:

- L P ## Param (Указатель на параметре # Param загружен в ACCU1)
LARn (Загрузка ACCU1 в ARn)

Другие инструкции для адресных регистров

Перенос из адресного регистра

- TARn (n =1 or 2): Перенос содержимого из ARn в ACCU1
- TARn <Address> Перенос содержимого из ARn в <Address>

<Address>:

- Процессорные регистры: AR2 (напр., TAR1 AR2)
- 32 -битовые абс. переменные: MDn, LDn, DBDn, DIDn (напр., TAR2 MD5, и т.д.)
- сиволич. 32 -битовые переменные: 32- битовые глобальные переменные (напр., TAR1 "Index", и т.д.) и TEMP- переменные OB, FB и FC (напр., TAR1 #Address, и т.д.)

Обмен адресных регистров

- TAR Обмен содержимого адресных регистров AR1 и AR2

Adding to Address Register

- +ARn Прибавить ACCU1-L к ARn
- +ARn P#x.m Прибавить указатель без указания области P#x.m к ARn

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.14



Information and Training Center
Knowledge for Automation

Передача из адресных регистров

Инструкция TARn передает полный указатель из адресного регистра ARn. В качестве цели может быть определен другой адресный регистр или двойное слово из памяти меркеров, локального стека, глобальных (общих) блоков данных и экземпляров блока данных. Если операнд в команде не определен, TARn передает содержание адресного регистра в ACCU1. Предыдущее содержание ACCU1 перемещается в ACCU2; содержание ACCU2 теряется. Содержание ACCU3 и ACCU4 (S7-400) остается неизменным.

Обмен адресных регистров

Инструкция TAR обменивает содержимое регистров адресных AR1 и AR2.

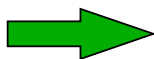
При добавление к регистрам адресным

Значение может быть прибавлено к адресным регистрам, например, чтобы увеличивать адрес адреса при каждом выполнении тела цикла. Значение может быть определено или как постоянная в форме внутризонного указателя в инструкции, или как содержание правого слова в ACCU1-L. Инструкции + AR1 и + AR2 интерпретируют значение, находящееся в ACCU1 как число в формате INT, расширяют его до 24 бит со знаком и добавляют к содержимому адресного регистра. Таким образом указатель может также быть уменьшен. Выход за пределы (верхний или нижний) максимальной области адресов (0 ... 65 535) не имеет никаких последствий; старшие биты просто "отрезаны" (отключены). Инструкции + ARn P # n.m добавляют внутризонный указатель к указанному адресному регистру. Указатель области P # n.m может иметь максимальный размер P # 4095.7. Ни одна из адресных инструкций, рассмотренных выше или на предыдущей странице, не изменяет битов в слове состояния.

Специальные особенности адресных регистров

Внутреннее использование AR1 STL/LAD/FBD-редактором

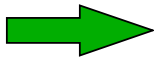
- ❑ При доступе к параметрам в FC, используются **регистры AR1 и DB**, если параметры имеют сложный тип данных (ARRAY, STRUCT, DATE_AND_TIME).
- ❑ При доступе к INOUT-параметрам FB, используются **AR1 и DB регистры**, если INOUT- параметр имеет сложный тип данных (ARRAY, STRUCT, DATE_AND_TIME)



Никакой доступ к локальным параметрам не возможен между командой загрузки в адресный регистр и командой косвенного доступа через регистр к желаемой переменной

Внутреннее использование AR2 STL/LAD/FBD-редактором

- ❑ **Регистр AR2 и регистр DI** используется как база адреса для адресации всех параметров и STAT-переменных в **FB**.



Если AR2 или DI - изменяются пользователем внутри FB, никакой доступ к собственным параметрам или STAT-переменным не может иметь место без восстановления обоих регистров.

- ❑ Никаких ограничений в отношении регистра AR2 и регистра DI в пределах FC нет.

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.15Information and Training Center
Knowledge for Automation

Адресный регистр AR1

Редактор STEP7 использует адресный, регистр AR1 для доступа к комплексным параметрам блока.

В пределах функций, регистры AR1 и DB изменяются при любом символическом доступе к параметрам блока типа "ARRAY" или типа "STRUCT".

Регистры AR1 и DB также изменяются при доступе к in / out параметрам типа "ARRAY" или типа "STRUCT" в FB.

Символический доступ в FB или FC к временным переменным не меняет ни AR1, ни регистр DB.

Адресный регистр AR2

Редактор STEP7 использует внутризонную косвенную адресацию через регистр для символического доступа данным экземпляра, то есть ко всем параметрам и статическим переменным FB. Регистр DI содержит номер экземпляра блока данных, а регистр AR2 соответствующее смещение адреса параметра или переменной в экземпляре блока данных. Никакой доступ к данным экземпляра не возможен после того, как регистры DI и AR2 изменены и если содержание обоих регистров не восстановлено. Если Вы хотите использовать регистр AR2 или регистр DI в пределах FB для ваших собственных целей, то рекомендуется следующая процедура :

1. Сохранить содержание DI и AR2 в переменных типа DWORD:

```
TAR2 # AR2_REG // Сохранение AR2 во временной переменной #AR2_REG
L DINO          // Сохранение DI в ACCU1
```

```
T # DI_REG      // Сохранение во временной переменной # DI_REG
```

2. Использование регистров DI и AR2 для Ваших собственных целей.

Никакой доступ к параметрам FB или статическим переменным не может иметь место в течение этой части программы.

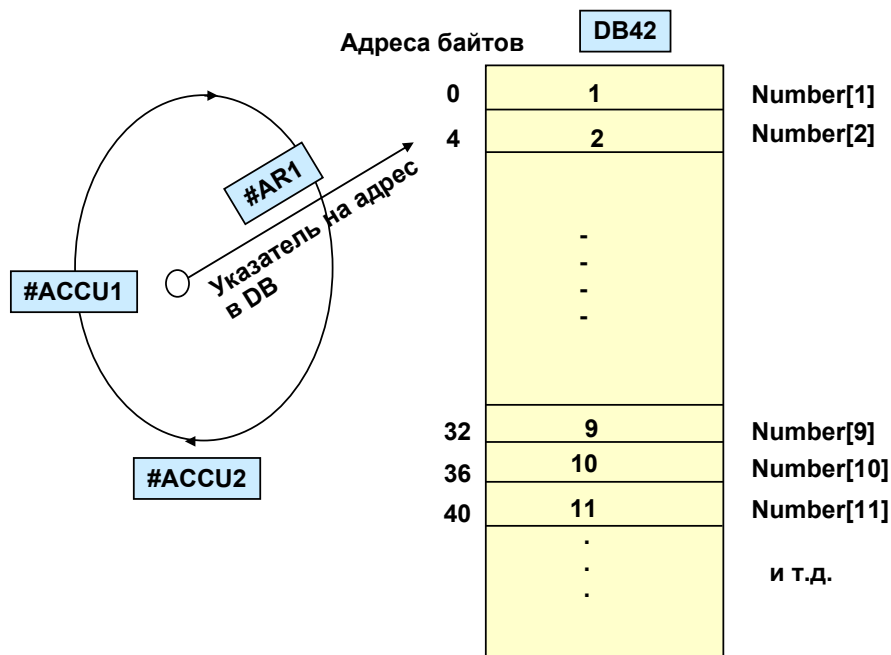
3. Восстановить первоначальное содержание регистра DI и регистра AR2:

```
LAR2 # AR2_REG // Загрузка AR2 содержимым # AR2_REG
```

```
OPN DI [# DI_REG] // Восстановление регистра DI
```

К параметрам FB и статические переменные можно опять обращаться символически.

Упражнение 4.2: Программирование цикла с регистровой косвенной адресацией



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.16Information and Training Center
Knowledge for Automation

Цель упражнения

Вы познакомитесь с использованием регистровой косвенной адресации в цикле, делая практический пример.

Задача

Регистровая косвенная адресация используется для программирования цикла. В этом цикле значения от 1 до 100 должны быть записаны в 100 последовательных ячеек блока данных.

Создать оптимальное по времени решение (без дополнительных временных переменных) упражнения 4.1. Сохранять значения счетчика цикла и инициализирующие значения в аккумуляторах.

Для адресации компонент #Number[.], используйте адресный, регистр AR1 (внутризонная регистровая косвенная адресация).

Что делать

1. Создайте FC42 и DB42.
2. В части декларации DB42, определите переменную #Number типа ARRAY[1..100] DINT.
3. В части декларации FC42, определите входной параметр #DB_Num типа WORD и временную переменную #I_DB_Num типа WORD.
4. В FC42 сначала откройте блок данных, чей номер получен на входе #DB_Num. Для этого используйте временную переменную #I_DB_Num.
5. Заполните поля от #Number [1] к #Number [100] в DB42 в возрастающем порядке числами от 1.0 до 100.0.
 - Для этого используйте программирование цикла (Инструкция: LOOP):
 - Используйте регистровую косвенную адресацию с AR1 для адресации компонентов #Number[.].
6. Вызовите FC42 в OB1, и назначьте фактические параметры на входной параметр #DB_Num. Загрузите блоки к CPU и испытайте Вашу программу.

Типы указателей в STEP 7

16-битовый указатель для косвенной адресации через память

- Для косвенного доступа через память к таймерам, счетчикам, для открытия блоков данных и для вызова FC без параметров и FB без параметров и STAT-переменные

32-битовый указатель для косвенной и регистровой адресации через память

- 32-битовый внутризонный указатель для косвенного доступа через память и регистры в области PI, PQ, I, Q, M, DB, DI и L (локальный стек данных)
- 32-битовый межзонный указатель для косвенного доступа через регистры в области PI, PQ, I, Q, M, DB, DI, L и V (локальный стек данных вызывающего блока)

48-битовый указатель (тип данных: POINTER)

- Тип данных для передачи параметров в блоки (FB и FC)
- В дополнение к 32-битовому межзонному указателю содержит номер DB

80-битовый указатель (тип данных: ANY)

- Тип данных для передачи параметров в блоки (FB и FC)
- В дополнение к 32-битовому межзонному указателю содержит номер DB, тип данных и коэффициент повторения

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.17



Information and Training Center
Knowledge for Automation

Типы указателей STEP7

Помимо типов указателя, описанных на предыдущей странице (16-битовый, 32-битовый внутризонный и 32-битовые внутризонный и межзонный), STEP7 имеет еще два дополнительных типа указателей:

- 48-битовый указатель (тип данных "POINTER")
- 80-битовый указатель (тип данных "ANY")

16- и 32-битовые типы указателя могут быть непосредственно загружены в аккумулятор или в адресный регистр и, таким образом, могут использоваться для косвенной адресации в пределах блока.

Типы указателя POINTER и ANY (размером больше, чем 32 бита) не могут быть загружены непосредственно в регистры и использоваться для косвенной адресации в пределах блока. Они используются исключительно для адресации фактических параметров при их передаче в блок.

Например, Вы можете объявлять параметр с типа данных POINTER или ANY в блоке и во время вызовы блока назначать параметр с адресом фактического параметра.

POINTER

Тип данных POINTER имеет менее важное значение для пользователя. Он главным образом используется STL/LAD/FBD-редактором для передачи фактических параметров сложного типа данных, типа ARRAY, STRUCT, и DT в вызываемый FB или FC.

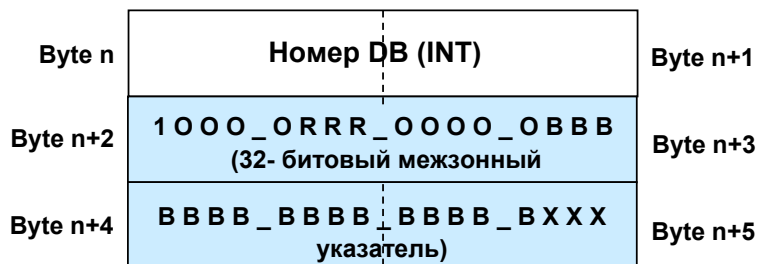
Так как STL/LAD/FBD-редактор немедленно проверяет правильность типа данных и длину при назначении фактического параметра, то внутренне достаточно просто передать полный начальный адрес фактического параметра. В пределах вызываемого блока Вы можете регистрировать тогда при доступе к фактическим параметрам косвенное использование POINTER.

ANY

Тип указателя ANY главным образом используется STEP7 для назначения параметров системным функциям (SFC) и системным функциональным блокам (SFB). Параметры типа данных ANY могут также использоваться пользователем, чтобы создавать мощные блоки.

Структура и назначение типа данных POINTER

Структура типа данных POINTER



Назначение параметров типа POINTER

- Вид указателя**
 P#DBn.DBX x.y где: n= номер DB, x= номер байта, y= номер бита
 P#DIn .DIX x.y (напр.: P#DB5.DBX3.4, P#DI2.DIX10.0, и т.д.)
 P#Zx.y где: Z= область, напр.: P, I, Q, M и L
 (напр.: P#I5.3, P#M10.0, и т.д.)
- Объявление адреса:**
 MD30 (в этом случае, номер DB , идентификатор
 #Motor_on области и битовый адрес автоматически
 "Motor_1".speed вводится в POINTER)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.18Information and Training Center
Knowledge for Automation

Тип данных POINTER Параметр типа данных POINTER содержит, в дополнение к внутризонному указателю, номер блока данных, как беззнаковое число (область значений: 0 ... 65 535). Номер блока данных содержится в первых двух байтах параметра типа POINTER, когда межзонный указатель ссылается на данные в глобальном блоке данных или в экземпляре. Во всех других случаях, когда адрес указывает в другие области, (P, I, Q, M, L), в первые два байта "POINTER" содержат "0".

Назначение параметров Если при вызов блока (FC или FB) должен быть назначен параметр типа данных POINTER, то это может быть сделано используя присвоение указателя или объявляя адрес.

Присвоение указателя В этом случае, указатель (P # ...) должен быть введен как первый бит адреса, например:

- P # DB10. DBX2. 0 // бит 2.0 в DB10, идентификатор области DB
- P # I5. 3 // I 5.3, DB номер = 0, идентификатор области I.

Объявление адреса В этом случае достаточно объявления адреса (без P #...). Адрес может быть введен абсолютным, то есть используя номер DB, идентификатор адреса и связанный байтовый или битовый адрес, типа:

- DB5. DBW10 // Бит 10.0, номер DB = 5, идентификатор области DB или символическим
- # Motor_on, "Motor_1".speed

В обоих случаях, STL/LAD/FBD -редактор устанавливает номер DB , идентификатор области и адрес byte.bit и вводит их в "POINTER".

Структура типа данных ANY

- Указатель ANY для типов данных

Byte n	16#10	Тип данных
Byte n+2	Коэффициент повторения	
Byte n+4	Номер DB	
Byte n+6	1 0 0 0 _ O R R R	0 0 0 0 _ O B B B
Byte n+8	B B B B _ B B B B	B B B B _ B X X X

Тип данных	Идентификатор
VOID	00
BOOL	01
BYTE	02
CHAR	03
WORD	04
INT	05
DWORD	06
DINT	07
REAL	08
DATE	09
TOD	0A
TIME	0B
S5TIME	0C
DT	0E
STRING	13

- Указатель ANY для параметрических типов

Byte n	16#10	Параметрический тип
Byte n+2	16#0001	
Byte n+4	16#0000	
Byte n+6	16#0000	
Byte n+8	Номер таймера, счетчика или блока	

Параметр. тип	Идентификатор
BLOCK_FB	17
BLOCK_FC	18
BLOCK_DB	19
BLOCK_SDB	1A
COUNTER	1C
TIMER	1D

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.19Information and Training Center
Knowledge for Automation

Тип данных ANY

Указатель ANY содержит, в дополнение к межзонному указателю области и номеру DB, идентификатор для типа данных и коэффициент повторения. С помощью него можно идентифицировать не только индивидуальный адрес, но также и полную область данных. Имеются две версии указателя ANY :

- Для переменных типа данных: указатель ANY содержит синтаксический ID для STL 16#10, идентификатор типа данных, коэффициент повторения, номер DB и межзонный указатель.
- Для переменных параметрических типов : указатель ANY состоит из синтаксического ID для STL 16#10, идентификатора для параметрического типа и 16-битного номера (число без знака) в байте n + 8 и байте n + 9, который является номером блока. Байты n + 4, ..., n + 7 заполнены "0".

Объявление указателя ANY

Переменные ANY- типа могут использоваться как IN, OUT и INOUT параметры в FC и FB.

Они могут также использоваться как временные переменные в FB и FC. С помощью этих временных переменных возможно создать указатель ANY , который изменяется во время выполнения и передавать его в вызываемый блок (см.: стр 21 "Косвенное назначение параметра типа ANY").

Идентификатор области (RRR):

000	I/O	001	Входы (PII)
010	Выходы (PIQ)	011	Меркеры
100	Блок данных, DB регистр	101	Блок данных, DI регистр
110	Локальные данные	111	Локальные данные вызывающего блока

Назначение параметров с типом данных ANY

Вид указателя:

- **R#[Data block.]Битовый адрес Числовой тип**

P#DB10.DBX12.0 REAL 20	Указатель на область в DB10, начинающуюся с 12-го байта, содержащую 20 ячеек с типом данных REAL (ARRAY[1..20] OF REAL)
------------------------	---

R# 10.0 BOOL 8	Указатель на область из 8 бит в IB10
-----------------	--------------------------------------

Объявление адреса:

- абсолютное:

DB5.DBD10 Тип данных: DWORD, коэффиц. повтор.(КП): 1
номер DB: 5, указатель: P#DB5.DBX10.0

IW32	Тип: WORD, КП: 1, №DB: 0, указатель: P#I 32.0
T35	Тип : TIMER, Номер.: 35

- **Символическое:**

#Motor_1.speed "Pump".Start	для элементарных типов данных компилятор устанавливает корректный тип данных, коэффициент повторения 1 и указатель
--------------------------------	--

Примечание

При символическом назначении (ARRAY, STRUCT, STRING, UDT) в указателе ANY компилятором устанавливается идентификатор типа данных 02 (BYTE) и длина

~~области в байтах~~

SIMATIC S7

Siemens AG 1999. All rights reserved

Date: 04.11.2005
File: PRO2 04E.20

Information and Training Center
Knowledge for Automation

Назначение

Параметру типа "ANY" может быть назначено значение в виде указателя или присваивая адрес переменной.

Назначение указателя

При назначении, использующем указатель (например:

P #DB5. DBX10. 0 INT 8) STL/LAD/FBD-редактор устанавливает указатель ANY , который соответствует типу с декларациями.

Назначение указателя всегда имеет смысл, когда должна быть адресована область данных, для которой не определена никакая переменная или например, никакая подходящая переменная (например, ARRAY или STRUCT) не может быть определена (например, P, PII, PIQ, M).

Кроме того, указатель должен использоваться, когда в пределах вызываемого блока требуется информация относительно коэффициента повторения и типа данных (напр., `ARRAY [1 .. 8] OF REAL`).

Назначение адреса

Параметр типа "ANY" может быть также назначен непосредственно с помощью адреса, на который указатель ANY должен указать. Эта декларация может быть сделана с помощью абсолютной или символьной адресации.

При назначении абсолютного адреса STL/LAD/FBD-редактор автоматически устанавливает связанный тип данных (BOOL, BYTE, WORD, DWORD,), коэффициент повторения "1", номер DB, а также межзонный указатель на первый бит адреса.

Аналогично, STL/LAD/FBD-редактор устанавливает правильную информацию, используя адрес, когда при назначении используется имя символа и переменная имеет элементарный тип данных.

Обратите внимание

Если переменная имеет сложный тип данных (например, ARRAY [1 .. 8] OF REAL), тогда STL/LAD/FBD-редактор просто вводит информацию относительно области в байтах, занятой переменной (то есть: коэффициент повторения: 32, тип данных: BYTE).

Косвенное назначение параметра типа ANY

Назначение фактического значения типа ANY временной переменной

- объявление временной переменной типа ANY в вызываемом блоке

например: `temp aux_pointer ANY`

- заполнение временной переменной ANY информацией о указателе

например:

```
LAR1 P##aux_pointer // Загрузить адрес на aux_pointer
L B#16#10 // Загрузить идентификатор 10
T LB [AR1,P#0.0] // и перенести его со смещением 0
L ...
...
```

- Назначение параметрам блока значения типа ANY (целевая область) с помощью вспомогательной переменной с указателем

например:

```
CALL FC 111
Targetfield:=#aux_pointer
```

Преимущество

- Динамическое переназначение параметрам указателя ANY во время выполнения

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.21



Information and Training Center
Knowledge for Automation

Косвенное назначение

Вызывающий блок может назначать FC или FB параметры типа ANY с помощью временной переменной типа ANY. Эта временная переменная должна быть сохранена в локальном стеке данных вызывающего блока. В этом случае STL-редактор не позволяет записывать указатели в эту переменную (в локальном стеке данных), но предполагает, что эта временная переменная ANY уже содержит указатель на фактическую переменную. Редактор передает, в этом случае, ANY-указатель, содержащийся во временной переменной, вызываемому FC или FB.

Преимущество

Вы имеете, таким образом, возможность введения указателя ANY в параметр ANY, который Вы можете изменять во время выполнения. Переменная указатель ANY может быть очень полезна, особенно в системных функциях, например, SFC 20 (BLKMOV) или SFC 21 (FILL).

Использование переданного указателя ANY

Address	Declaration	Name	Type	Initial Value	Comment
0.0	in	Par Pointer	ANY		
	out				
	in out				
0.0	temp	Data_type	BYTE		
2.0	temp	WF	WORD		
4.0	temp	DB Nr	WORD		
6.0	temp	Area_Pointer	DWORD		

Network 1: Выделение типа данных, коэффициента повторения, номера DB и указателя

```

L   P##Par_Pointer // Загрузка адреса of #Par_Pointer в ACCU1
LAR1                                // и загрузка его в AR1;
L   B [AR1,P#1.0] // Выделение типа данных из указателя
T   #Data_type // и загрузка во временную переменную;
L   W [AR1,P#2.0] // Выделение коэффициента повторения
T   WF // и загрузка во временную переменную;
L   W [AR1,P#4.0] // Выделение номера DB
T   #DB_Nr // и загрузка во временную переменную;
L   D [AR1,P#6.0] // Выделение указателя
T   #Area_Pointer // и загрузка во временную переменную;

```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.22



Information and Training Center
Knowledge for Automation

Краткий обзор

Внутри вызываемого блока (FB или FC) информация, которая находится в переданном указателе (тип POINTER или ANY), может быть прочитана с помощью косвенной регистровой адресации.

Процедура

Процедура чтения информации из переданного указателя "ANY" состоит из следующих шагов. Эти шаги показаны в примере на слайде, где объявлены входной параметр (тип "ANY") с именем #Par_Pointer и несколько временных переменных для временного хранения информации.

1. Прежде всего межзонный указатель устанавливается на переданный указатель "ANY", и загружается в адресный регистр AR1:

- LAR1 P ## Par_Pointer // в FB или
- L P ## Par_Pointer // в FC, адрес должен сначала быть загруженным в LAR1 // ACCU1 и оттуда в регистр AR1

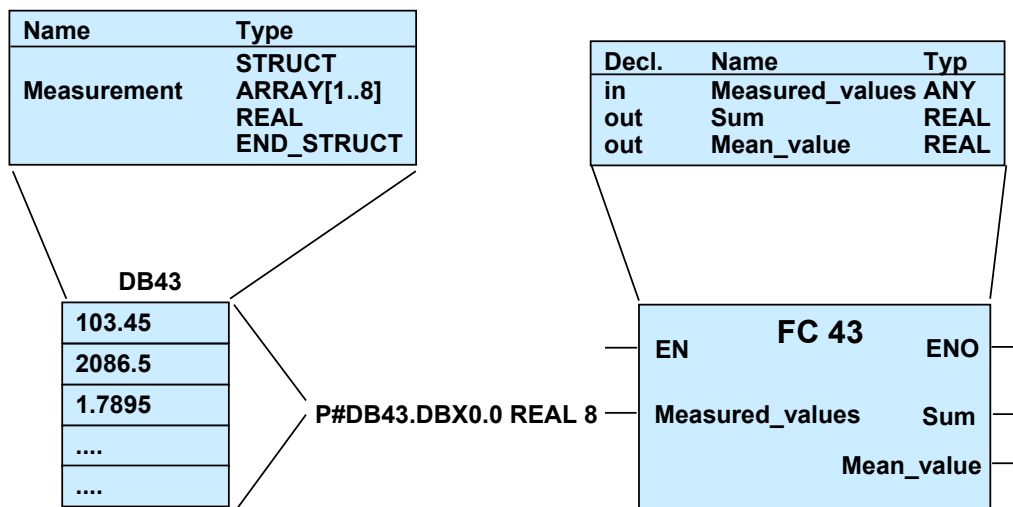
Переданный указатель "ANY", в случае FB, сохраняется в экземпляре DB (он автоматически открыт) или в случае FC в локальном стеке данных вызывающего блока.

2. При использовании регистровой косвенной адресации, информация из переданного указателя "ANY" может теперь быть прочитана и, например, может быть временно сохранена во временных переменных блока для дальнейшей обработки.

- L B [AR1, P # 1.0] // загрузка в ACCU1 идентификатора типа данных // фактического параметра
- L W [AR1, P # 2.0] // загрузка коэффициента повторения в ACCU1
- L W [AR, P #4.0] // загрузка в ACCU1 номера DB, в котором // находится актуальный параметр, или "0", если // актуальный параметр находится в P, PII, PIQ, M, L
- L D[AR1,P#6.0] // загрузка в ACCU1 межзонного указателя на // актуальный параметр

Информация о фактических параметрах из указателя ANY должна быть далее обработана согласно задаче.

Упражнение 4.3: Функция вычисления суммы и среднего значения



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_04E.23Information and Training Center
Knowledge for Automation

Краткий обзор

Родовые FC или FB могут быть созданы с помощью типа данных "ANY". Родовые FC или FB не "привязаны" к определенным типам данных. Они могут "приспосабливаться" во время выполнения к переданным им типам данных или областям различной длины.

Цель

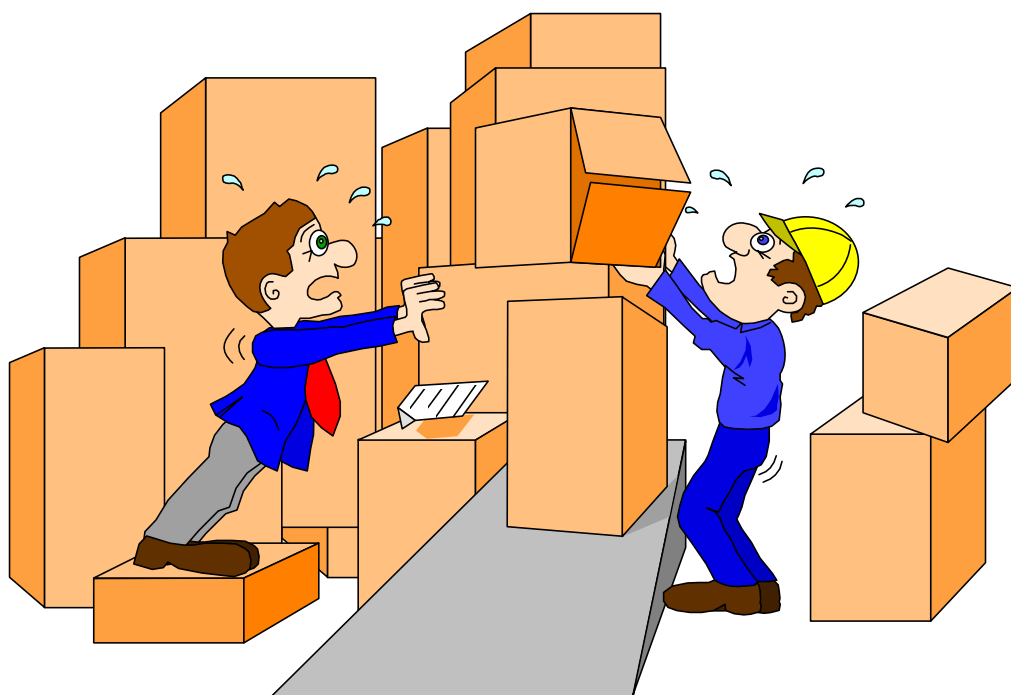
Создайте FC43 со следующими функциональными возможностями:

- функция ожидает область REAL-значений во входном параметре Measured_values (тип "ANY").
- функция выдает сумму значений элементов переданной области в выходной параметр #Sum (тип: REAL) и среднее значение всех элементов области в выходном параметре # Mean_value (тип: REAL).
- если передан другой тип данных, FC 43 завершается с ошибкой (то есть BR-бит = 0, недействительны значения в #Sum и # Mean_value).

Что делать

1. Создайте FC43, и объявите выше упомянутые вход и выходы в списке параметров. Также объявите соответствующие временные переменные для временного хранения информации о коэффициенте повторения, номере DB и указателе на фактическую область параметра.
2. Прочитайте из переданного указателя "ANY" в первую очередь информацию о типе данных и завершите FC43, если тип данных фактического параметра не REAL.
3. В цикле (инструкция LOOP), запрограммируйте вычисление сумм всех элементов области. Вычислите среднее значение и передайте результаты в соответствующие выходные параметры.
4. Создайте DB43. Объявите в DB43 переменную Measurement типа ARRAY[1..8] и введите соответствующие значения.
5. Вызовите FC43 в OB1. Назначьте входной параметр в виде указателя. Назначьте адреса в области меркеров для выходов.
6. Загрузите необходимые блоки в CPU и проверьте результат.

Переменные и типы данных STEP 7



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.1



Information and Training Center
Knowledge for Automation

Содержание

Стр

Значения переменных и типы данных	2
Объявления и свойства переменных	3
Обзор типов данных в STEP 7	4
Элементарные типы данных в STEP 7	5
Важность сложных типов данных	6
Сложные типы данных в STEP 7	7
Параметрические типы в STEP 7	8
Области для хранения переменных	9
Метод функционирования локального стека данных	10
Пример: Использование локального стека LAD-редактором	11
Блоки данных (DB)	12
Тип данных: ARRAY	13
Объявление и инициализация массивов	14
Хранение переменных типа ARRAY в памяти	15
Тип данных: STRUCT	16
Объявление структур	17
Хранение переменных типа STRUCT в памяти	18
Тип данных, определенный пользователем: UDT	19
Использование UDT	20
Тип данных: DATE_AND_TIME	21
Функции для работы с переменными типа DT	22
Тип данных: STRING	23
Хранение строковых переменных в памяти	24
Functions for Processing STRING Variables	25
Упражнение 5.1: Использование сложных типов данных	26
Упражнение 5.2: Доступ к сложным типам данных	27
Дополнительное упражнение 5.3:	
Чтение времени и даты с помощью SFC 1 (READ_CLK)	28

Значения переменных и типы данных

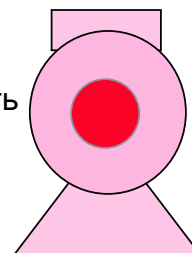
Тип данных характеризует основные свойства данных

- непрерывная область значений: напр., актуальная скорость
- "yes/no" - свойство: напр., вибрации

Тип данных устанавливает:

- доступный диапазон значений (INT: -32 368 ... +32 367, и т.д.)
- допустимые операции (арифметические инструкции: +, -, и т.д.)
- структуру объектов данного типа, т.е. расположение битов в памяти

Переменные разрешают Вам сохранять и позже продолжать обработку величин



Actual_speed: REAL

Set_speed: REAL

Disturbance: BOOL

Enable: BOOL

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.2



Information and Training Center
Knowledge for Automation

Краткий обзор

Современные компьютерные системы были развиты, чтобы упростить и ускорить сложные и требующие больших затрат времени вычисления. Способность обрабатывать большие количества информации, чтобы хранить ее и делать затем вновь доступной, играет важную роль для большинства применений.

Информация, доступная диспетчеру, состоит из определенного количества данных относительно "реального мира". Данные представляют собой абстрактные сущности, потому что непредвиденные и незначительные свойства объектов не приняты во внимание для данной определенной проблемы.

Типы данных

Часто весьма трудно решить, как данные должны быть представлены. Выбор весьма часто также ограничивается доступными возможностями. С одной стороны, свойства объектов, которые описываются данными, должны быть правильно отражены, с другой стороны, инструкции, которые являются необходимыми для управления процессом, должны также быть выполнимы при данном представлении данных.

Тип данных определяет, какие значения принимают данные и какие инструкции могут быть выполнены с этими значениями.

Тип данных уникально определяет:

- Возможный диапазон
- Разрешенные инструкции

Тип данных определяет также представления (формат) значения отдельных битов в представлении величины данного типа.

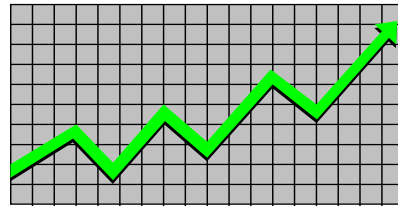
Значения переменных Наряду с командами, переменные - наиболее важные элементы

системы программирования. Их задача состоит в том, чтобы сохранять значения в программе так, чтобы они могли быть обработаны в более позднее время. Значение переменной может быть сохранено "где-нибудь" в памяти PLC.

Объявления и свойства переменных

Следующие свойства определяются при объявлении переменных:

- символическое имя
- тип данных
- видимость переменной



Meas_point: ARRAY[1..10]

Meas_point[1]: Real

Meas_point[2]: Real

Meas_point[3]: Real

Meas_point[10]: Real

Переменные могут быть объявлены:

- в глобальной символьной таблице (элементарные типы данных)
- в таблице описаний глобального блока данных (все типы данных)
- в таблице описаний логического блока (OB, FB и FC)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.3



Information and Training Center
Knowledge for Automation

"Обычные" переменные

При обычном программировании PLC, используется абсолютная адресация памяти PLC при которой определяется область памяти (например: M = память меркеров, I = входы, и т.д.), ширина доступа (например: B = байт, W = слово, и т.д.) байт /(битовый) адрес. Эта адресация областей памяти, может использоваться в пределах программы для различных целей, например, для целых чисел (например, DINT), чисел с плавающей точкой (REAL) или просто как объединение отдельных сигналов (например, WORD).

Вплоть до последнего времени это требовало от программиста, чтобы он помнил формат и прикладную цель отдельных областей памяти. Дефектные программы могли бы давать "результат", потому что неправильные адреса памяти или неправильный формат данных неумышленно использовались в инструкциях.

Объявление переменных

Ранние системы PLC (например, STEP 5) разрешали использовать символы пользователем, что делало программы более легкие читаемыми. STEP 7 пошел на один шаг дальше и использует переменные вместо адресов PLC и символов.

Явно объявляя переменную, Вы устанавливаете следующие свойства :

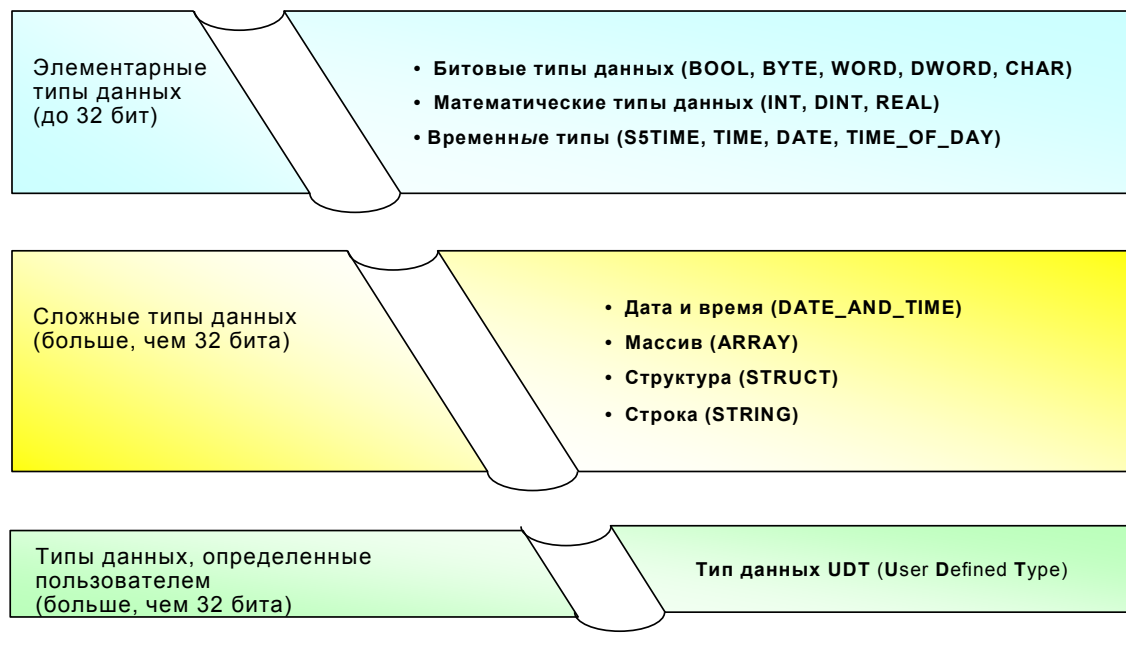
- Символическое имя переменной
- Тип данных переменной
- Видимость переменной

Если переменные объявлены, программный редактор может тогда, например, использовать информацию о типе данных, чтобы проверить допустимость инструкций для параметра, назначенного при вызове блока.

Видимость переменных Переменные, которые объявлены в глобальной таблице символов или в глобальном блоке данных, могут быть адресованы всеми блоками папки программы (папки S7Program). По этой причине такие переменные называются *глобальными переменными* .

Переменные и параметры, которые объявлены в секции объявлений логического блока, называются *локальными*; они могут использоваться только в пределах секции инструкции данного блока.

Обзор типов данных в STEP 7



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.4



Information and Training Center
Knowledge for Automation

Краткий обзор

Решение задач автоматизации, использующих компьютерные системы, основано на алгоритмах, которые обрабатывают данные, собранные датчиками, чтобы выдать на выходы новые значения для управления приводами. Программы - основная формы алгоритмов, которые получают определенные данные и обрабатывают их.

Элементарные типы данных

Элементарные типы данных составляют "атомы" каждой системы программирования. Выбор элементарных типов данных системы программирования много говорит о ее прикладной области.

В STEP 7 элементарные типы данных - предопределены в соответствии с IEC 1131-3. Типы данных выбраны таким образом, что в дополнение к типичным задачам PLC типа двоичной и аналоговой обработки сигнала, возможна простая сигнальная система, а также управление данными времени.

Тип данных определяет размер места в памяти, которое требует переменная. Элементарные типы данных в STEP 7 всегда имеют длину не более, чем 32 бита и могут быть загружены в аккумуляторы полностью и обработаны с помощью инструкций STEP 7.

Сложные типы данных

Основная идея структурирования данных - дифференцирование между элементарными и структурированными.

Первые - атомы, из которых создаются сложные типы данных.

В STEP 7 сложные типы данных могут только использоваться для переменных, объявленных в глобальном DB или в локальном стеке данных. Переменные сложных типов данных не могут быть полностью загружены в аккумулятор и обработаны

Определенный пользователем тип данных

Сложные типы данных (напр., ARRAY[1..3] OF REAL) не имеют собственных идентификаторов для типа данных и как результат не могут многократно использоваться для объявления параметров или переменных.

С помощью определенных пользователем типов данных (UDT), может быть создан уникальный, структурированный тип данных, который может использоваться так часто, как требуется для объявления переменных или параметров.

Элементарные типы данных в STEP 7

Ключевое слово	Длина (в битах)	Пример константы
BOOL	1	1 or 0
BYTE	8	B#16#A9
WORD	16	W#16#12AF
DWORD	32	DW#16#ADAC1EF5
CHAR	8	'w'
S5TIME	16	S5T#5s_200ms
INT	16	123
DINT	32	65539 or L#-1
REAL	32	1.2 or 34.5E-12
TIME	32	T#2D_1H_3M_45S_12MS
DATE	16	D#1999-06-14
TIME-OF-DAY	32	TOD#12:23:45.12

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.5Information and Training Center
Knowledge for Automation

BOOL, BYTE, WORD, DWORD, CHAR Переменные типа данных BOOL состоят из одного бита, переменные типов данных BYTE, WORD, DWORD - соответственно из 8, 16 и 32 битов. Отдельные биты в этих типах данных не оцениваются. Специальные форматы этих типов данных - BCD -формат и формат счетчика, используемый в функциях счетчика. Тип данных CHAR представляет знак в ASCII коде.

S5TIME Переменные типа данных S5TIME, требуются для определения значений таймера в таймерных функциях. Вы определяете время в часах, минутах, секундах и миллисекундах. Вы можете вводить значения таймера с подчеркиванием (1h_4m) или без подчеркивания (1h4m). Функции FC 33 и FC40 из библиотеки "IEC Function Blocks" конвертируют S5TIME-формат в формат TIME и TIME в формат S5TIME.

INT, DINT, REAL Переменные этих типов данных представляют числа, которые могут использоваться в математических действиях.

TIME Переменная типа данных TIME занимает двойное слово. Эта переменная используется, например, для определения значений таймера в функциях IEC-таймера. Содержание переменной интерпретируется как число DINT в миллисекундах и может быть или положительным или отрицательным (например: T # -1s = L # -1 000, T # 24d20h31m23s647ms = L # 214748647).

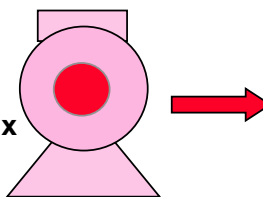
DATE Переменная типа данных DATE, сохраняется в слове в форме целого числа без знака. Содержание переменной представляет число дней от 01.01.1990 (например: D # 2168-12-31 = W #16 # FF62).

TIME_OF_DAY Переменная типа данных TIME_OF_DAY, занимает двойное слово. Она содержит число миллисекунд начиная с начала дня (0:00 часов) в формате целого числа без знака. (Например: TOD # 23:59:59.999 = DW # 16 # 0526_5B77).

Важность сложных типов данных

"Лучшее" структурирование данных:

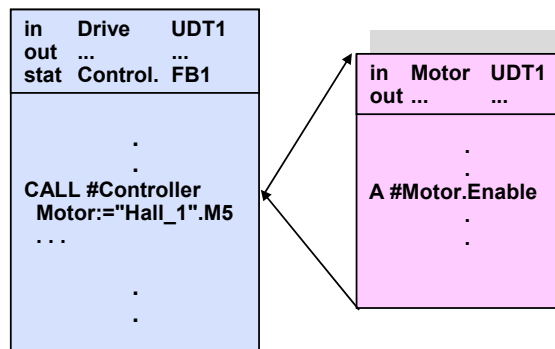
- адаптирует их к задаче
- создает "корректный" тип данных



Motor: STRUCT	
Set_speed:	REAL
Actual_speed:	REAL
Enable:	BOOL
Disturbance:	BOOL
END_STRUCT	

Компактная форма передачи данных при вызове блока:

- "много" элементов данных могут быть переданы в одном параметре
- делает возможным структурированное программирование
- блоки "связываются" только через параметры
- программное обеспечение многократного пользования



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.6Information and Training Center
Knowledge for Automation

Сложные типы данных

Сложные типы данных (массивы и структуры) - группировки элементарных или сложных типов данных.

Сложные типы данных полезны для организации сложных данных. Таким образом, программист может производить типы данных, которые удовлетворяют специфике задачи. Он может объединять элементарные, информационные единицы в новую "единицу" с собственным именем. Типичный пример для структуры - запись данных двигателя. Двигатель описан как список признаков (свойства, состояния), типа *#Set_speed*, *#Actual_speed*, *#Enable* и *#Disturbance*. Некоторые из этих признаков могли бы в свою очередь быть структурами.

#Disturbance могла бы, например, быть составлена из индивидуальных компонентов (битов), которые для каждой части дают Вам более точную информацию относительно причин вибраций.

Структурированное программирование

Сложные данные, в частности можно передавать при вызове блока как единое целое, то есть в одном параметре.

Таким образом, множество элементарных информационных единиц может быть передано при вызове блока изящным и компактным способом.

Программное обеспечение многократного использования

Этот тип передачи данных делает возможным структурированное программирование и гарантирует высокую степень возможности многократного использования программного обеспечения, созданного однажды. Полная задача автоматизации делится на индивидуальные блоки. В разделе инструкций не используются никакие вызов глобальных адресов или переменных в глобальных DB. Обработка информации выполнена исключительно с параметрами, через которые передаются данные процесса. Результаты обработки также возвращаются в параметрах вызывающему блоку.

Сложные типы данных в STEP 7

Ключевое слово	Длина (в битах)	Пример	
DATE_AND_TIME (Дата и время)	64	DT#99-06-14-12:14:55.0	
STRING (Строка символов с max. 254 символами)	8 * (число символов +2)	'This is a string' 'SIEMENS'	
ARRAY (Группа элементов одного и того же типа данных)	Определяется пользователем	Meas_vals: ARRAY[1..20] INT	
STRUCT (Структура, группа элементов разных типов данных)	Определяется пользователем	Motor: STRUCT Speed : INT Current : REAL END_STRUCT	
UDT (User Defined Data Type = "Шаблон", состоящий из элементарных и/или сложных типов данных)	Определяется пользователем	UDT как блок	UDT как элемент массива
		STRUCT Speed : INT Current : REAL END_STRUCT	Drive: ARRAY[1..4] UDT1

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.7Information and Training Center
Knowledge for Automation

Массивы и структуры (ARRAY и STRUCT)

С помощью типа ARRAY (массив), несколько объектов одного и того типа могут быть объединены в один тип данных. ARRAY - тип данных, который состоит из определенного числа элементов заданного типа. Каждому элементу ARRAY назначен индекс. Индекс используется для доступа к элементам.

Пример ARRAY - ряд измерений, который состоит из установленного числа отдельных измеренных величин.

Таким же образом, как ARRAY позволяет объединение элементов одного и того же типа в более "высокую" единицу, данные типа STRUCT (структура), позволяет создать объединение элементов различных типов данных.

Сложные типы данных определяются заранее. Данные типа DATE_AND_TIME имеют длину 64 бита. Длины типов данных ARRAY, STRUCT и STRING определяются пользователем.

Тип данных определенный пользователем

С помощью определенных пользователем типов (UDT), Вы можете определять специальные данные типы (структуры), которые могут после этого использоваться так часто как, Вы пожелаете. Структура данных сохраняется в блоке UDT (UDT1 ... UDT65535) и может использоваться, как "шаблон" в разделе описаний типа данных переменной или параметра в OB, FC, FB и DB.

С помощью UDT Вы можете сэкономить время написания программы, когда одна и та же структура требуется несколько раз.

Пример: Вам требуется одна и та же структура 10 раз в блоке данных.

Сначала Вы определяете структуру и сохраняете ее как UDT 1.

В DB Вы определяете переменную "Drive" как массив с 10 элементами типа UDT1:

```
Drive: array [1 .. 10]
UDT 1
```

Таким образом, Вы создали 10 переменных с данной структурой, которая определена в UDT 1, без дополнительного "печатания".

Параметрические типы в STEP 7

Ключевое слово	Длина (в битах)	Пример
TIMER	16	Contact time: TIMER · SI #Contact_time
COUNTER	16	NoCompParts: COUNTER · LC #No_Comp_Parts
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	16	Recall: BLOCK_FB · UC #Recall
Pointer	48	Measure: POINTER · L P##Measure
ANY	80	Measured Values: ANY · L P##Meas_Values

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.8Information and Training Center
Knowledge for Automation

Параметрические типы

В дополнение к элементарным и сложным типам данных, Вы можете определять параметры FC и FB с помощью параметрических типов. С этими формальными параметрами Вы можете выполнять те же самые инструкции, что и с фактическими адресами. Эти формальные параметры должны быть при вызове блока снабжены связанными фактическими параметрами.

TIMER и COUNTER BLOCK_XX

Эти типы параметра определяют формальные параметры типа TIMER или COUNTER.

С помощью параметрических типов BLOCK_FB или BLOCK_FC программные блоки можно передавать как параметры в вызываемые блоки. Однако можно передавать только те блоки (FB, FC), которые не имеет параметров или статических переменных (BLOCK_FB), . Формальные логические блоки могут только вызываться, используя инструкции UC, или CC (не CALL) в пределах вызываемого блока. Нет ограничений для передачи блоков данных (DB, SDB) и для связанных с ними инструкций (например, OPN ...).

POINTER

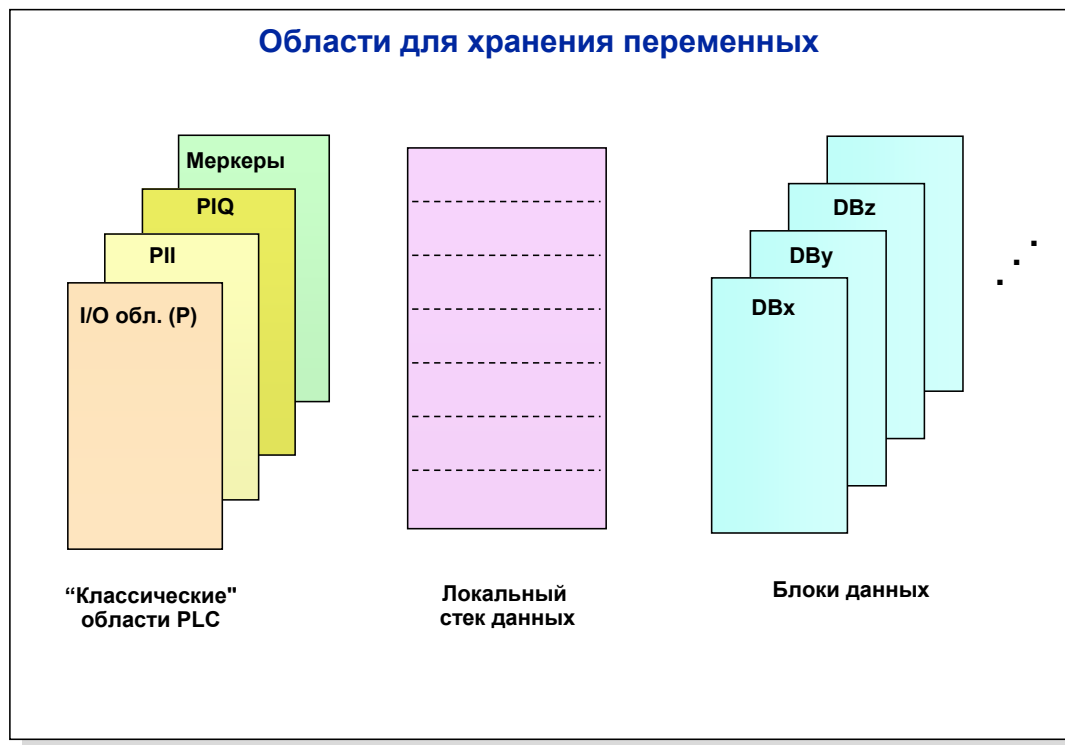
POINTER используется, когда любой тип данных может быть типом данных фактического параметра. POINTER содержит полный начальный адрес (номер DB, область данных, адрес байта и адрес бита) фактического параметра.

Вы можете назначать формальный параметр типа POINTER, давая адрес фактического параметра: например, P # M50.0.

ANY

ANY используется, когда любой тип данных может быть типом данных фактического параметра. В дополнение к полному начальному адресу в указателе ANY передают также информацию относительно типа данных и длины области.

P # M10.0 Byte 10 (Область из 10 компонентов типа данных BYTE, начинающаяся с MB 10).



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.9Information and Training Center
Knowledge for Automation**Краткий обзор**

Кроме программных блоков, программа пользователя также состоит из данных, содержащих информацию относительно состояния процесса, сигналов, и т.д., которые обрабатываются согласно инструкциям программы пользователя.

Переменные могут иметь постоянное местоположение в памяти изображения процесса, области памяти меркеров или в блоках данных. Они могут быть динамически помещены во время выполнения в L-Stack.

PII, PIQ, память меркеров, ввод - вывод

Элементарные переменные могут быть объявлены в глобальной таблице символов, расположенной в папке программы.

В дополнение к символическому имени переменной, Вы должны также указать область памяти, состоящую из идентификатора области и длины, а также тип данных (например, FullCrate MW 10 INT).

В отличие от соответствующей таблицы символов в STEP 5 (Assignment List - Список назначений), редактор программы разрешает не только использование символического имени вместо абсолютного адреса. Он также

контролирует правильное использование переменной, когда параметры назначены при вызове блока.

Переменные, которые были объявлены в глобальной таблице символов, являются общими. Все блоки в папке программы имеют доступ к ним.

Локальный стек данных

Локальный стек данных (L-Stack) - область для хранения:

- Временных переменных логического блока, включая стартовую информацию OB.
- Фактические параметры, которые нужно передать при вызове функций.
- Логические промежуточные результаты в LAD-программе.

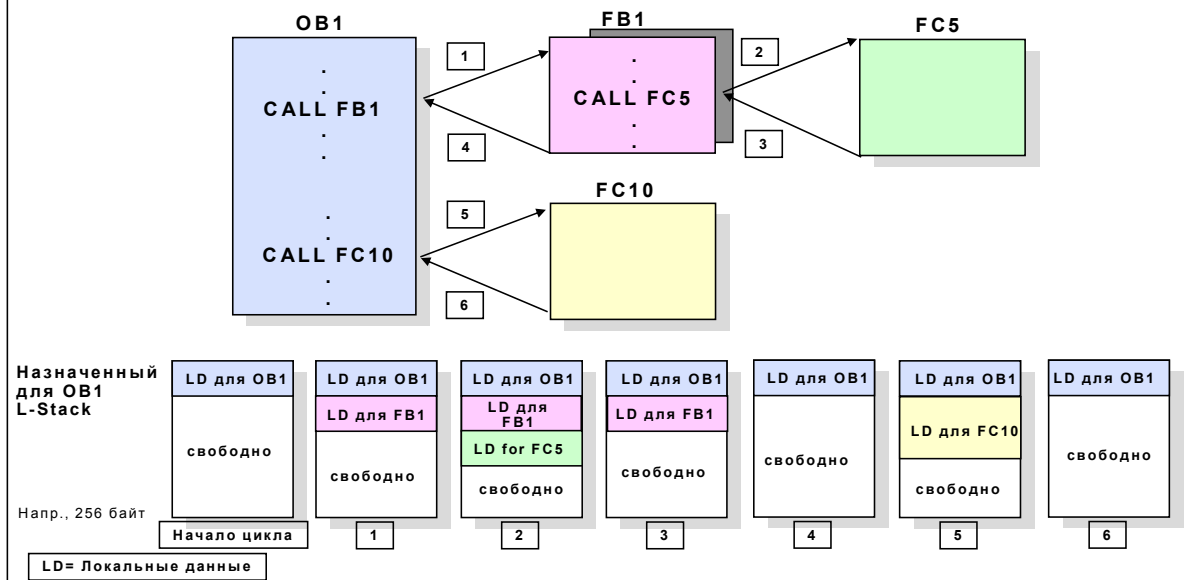
Области для переменных в L-Stack'е создаются динамически во время выполнения блока и освобождаются после выполнения блока.

Блоки данных

Блоки данных - блоки, используемые логическими блоками программы пользователя для хранения данных.

В отличие от временных переменных, содержание переменных в DB - не изменяется, когда выполнение блока закончено.

Метод функционирования локального стека данных



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.10



Information and Training Center
Knowledge for Automation

Локальный стек данных

Каждому классу приоритета, то есть каждому OB, назначен собственный L-стек для временных переменных OB или дополнительно называемых блоков.

Прежде, чем блок (OB, FB или FC) начинает выполняться, система резервирует динамическую память в L-стеке для временных переменных, объявленных в части декларации блока. Память освобождается после BE (Block End - конец блока).

Последовательность Приведенный выше слайд показывает типичную последовательность циклического выполнения OB1. Прежде, чем OB1 начинает выполняться, операционная система резервирует место в памяти (в L-Steck'e) для временных переменных OB1. Таким образом, помимо временных переменных, объявленных пользователем, создается и инициализируется стартовой информацией область 20 в байт.

1. Перед началом выполнения FB1, операционная система резервирует память для временных переменных FB1. Соответствующая область памяти непосредственно следует за памятью для временных переменных OB1.
2. Перед началом выполнения FC5, операционная система резервирует память для временных переменных FB5. Соответствующая область памяти непосредственно следует за памятью для временных переменных FB1.
3. После завершения FC5, связанная память освобождается.
4. После завершения FB1, связанная память освобождается.
5. Теперь операционная система резервирует память для FC10. Эта область непосредственно следует за памятью для временных переменных OB1. Используется то место памяти, которое предварительно использовалось для FB1 и FC5, то есть временные переменные FB1 и FC5 теперь переписываются.

Преимущества

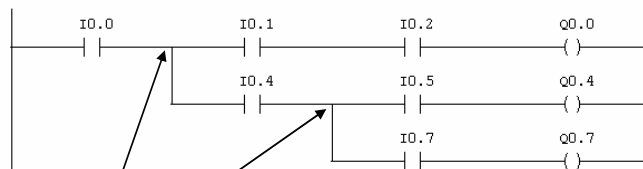
Управление временной памяти выполняется операционной системой и не должно быть организовано пользователем (позволяет избежать ошибок программирования).

Если OB соответствующего класса приоритета прерван OB с другим классом приоритета, местные данные не нужно спасать. Различным OB назначены собственные локальные стеки данных.

Пример: Использование локального стека LAD-редактором

Переходы в LAD

Network 2: Title:



Местоположение переходов

STL представление

Network 2: Title:

```

A      I      0.0
=      L      20.0
A      L      20.0
A      I      0.1
A      I      0.2
=      Q      0.0
A      L      20.0
A      I      0.4
=      L      20.1
A      L      20.1
A      I      0.5
=      Q      0.4
A      L      20.1
A      I      0.7
=      Q      0.7
  
```

Вспомогательные
переменные для
локального стека
данных

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.11



Information and Training Center
Knowledge for Automation

Переходы в редакторе LAD

Приведенный выше пример показывает представление переходов, которые может запрограммировать пользователь с помощью редактора LAD STEP 7, вставляя дополнительные катушки (например. Q 0.7).

Коннекторы

В STEP 5, программирование переходов не было возможно непосредственно. Пользователь должен был вставить вспомогательную переменную, как правило меркерный бит (connector), в месте выполняющегося перехода, как выход сети.

В следующих сетях - отдельном для каждого дополнительного ветвления - эта вспомогательная переменная тогда используется в каждом случае как вход.

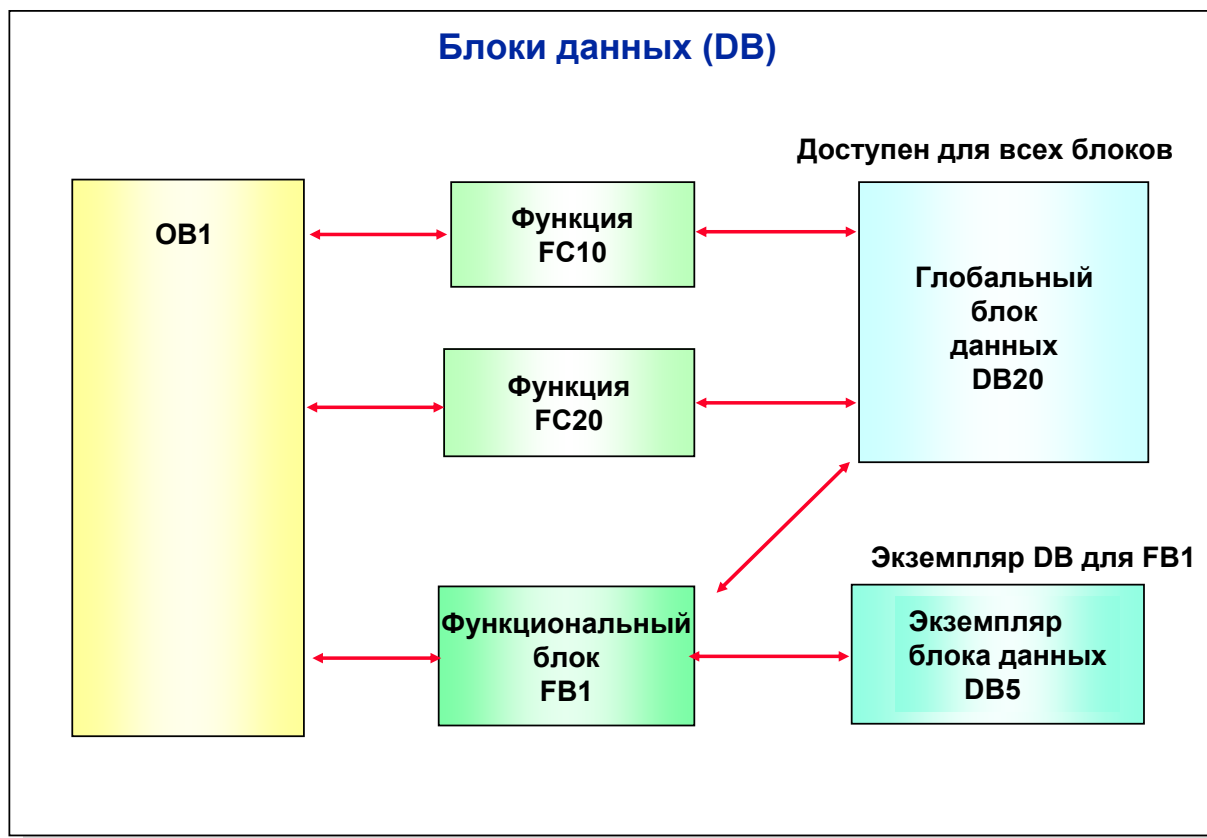
Редактор LAD

С помощью редактора LAD STEP 7 такие переходы в программе возможны непосредственно. Внутренняя вспомогательная переменная - бит локального стека данных - также используется при выполнении перехода редактором LAD.

Использование памяти локальных данных дает гарантии в этом случае, что две вспомогательные переменные (L 20.0 и L 20.1) не могут быть изменены блоками, вызванными в это же время.

Временные переменные

Пользователь может также объявлять временные переменные в части декларации и получать к ним доступ абсолютно или символически, то есть через указанное имя.



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.12Information and Training Center
Knowledge for Automation**Краткий обзор**

Блоки данных используются, чтобы хранить данные пользователя. Точно так же как, логические блоки, блоки данных занимают место в памяти пользователя. Переменные данные (например, числовые значения), с которыми работает программа пользователя, находятся в блоках данных. Программа пользователя может получить доступ к данным блока данных через инструкции для бита, байта, слова или двойного слова. Доступ может быть символическим или абсолютным.

Области приложения Блоки данных могут быть созданы пользователем, в зависимости от их содержания, различными способами. Различаются:

- Глобальные (Shared - разделяемые) блоки данных: они содержат информацию, к которой могут обращаться все логические блоки программы пользователя.
- Экземпляры (Instance) блока данных: они всегда назначаются для одного FB. Данные этого DB должны только быть обработаны связанным FB.

Создание DB

Глобальный DB создается или используя DB-редактор, или в соответствии с предварительно созданным UDT (типом данных, определенным пользователем).

Экземпляр блока данных создается, когда вызывается блок FB.

Регистры

CPU имеет двух регистры блоков данных, регистр DB и регистр DI. Тем самым могут быть открыты в одно и то же время два блока данных.

Тип данных: ARRAY

ARRAY (массив):

- Группа компонентов с одинаковым типом данных

- Объявление:

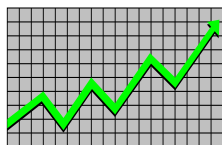
- одномерный:

Имя_массива: ARRAY[*minИндекс*..*maxИндекс*] OF *Тип_данных*;

- многомерный:

Имя_массива :ARRAY[*minИндекс 1*..*maxИндекс 1*, *minИндекс 2*..*maxИндекс 2*,...] OF *Тип_данных*;

Index: тип данных - INT (-32768...32767)



Meas_value: ARRAY[1..10]

Meas_value[1]:	Real
Meas_value[2]:	Real
Meas_value[3]:	Real

...

Meas_value[10]:	Real
-----------------	------

Примеры:

- Объявление переменных:

- одномерная: *Meas_value*: ARRAY[1..10] OF REAL;

- многомерная: *Position*: ARRAY[1..5,2..8,...] OF INT;

- Доступ к переменным:

- L #Meas_value[5] // Загрузить 5-й элемент массива

// Meas_value в ACCU1

- T #Result[10,5]

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.13



Information and Training Center
Knowledge for Automation

Краткий обзор

Тип данных ARRAY представляет область с определенным числом элементов одного и того же самого типа данных. Массив (= ARRAY) может иметь до 6 измерений (число индексов). Следующие наложены на типы данных компонентов массива:

- Элементарный - никакого ограничения
- Сложный: DATE_AND_TIME, STRUCT, STRING, UDT
- Параметрические типы не допустимы
- Тип FB недопустим (в модели мультитемпляров)

Массивы не могут быть вложенными (т.е. нельзя создать "массив массивов"). Пределы диапазона индекса определены диапазоном типа INT, то есть от -32768 до 32767.

Доступ

STL инструкции могут использоваться для доступа к компонентам массива элементарных типов данных. Компонент массива адресуется с помощью имени массива и индекса в квадратных скобках.

Индекс должен быть фиксированной величиной, то есть постоянным.

Индексация переменной во время выполнения в STL не возможна.

Обратите внимание

Индексация переменной индивидуальных элементов массива возможна только на языке S7-SCL. Переменный доступ может быть осуществлен в STL только с помощью косвенной адресации (через память или регистровой).

Объявление и инициализация массивов

DB5 "Declaration View"

Address	Name	Type	Initial Value
0.0		STRUCT	
+0.0	sequence	ARRAY[1..10]	5 (2.730000e+002) , 3 (1.000000e+001)
+4.0		REAL	
+40.0	result	ARRAY[1..5, 3..7]	10 (
+2.0		INT	
=90.0		END STRUCT	

Press F1 for help.

DB5 "Data View"

Address	Name	Type	Initial Value	Actual Value
0.0	sequence[1]	REAL	2.730000e+002	2.730000e+002
4.0	sequence[2]	REAL	2.730000e+002	2.730000e+002
8.0	sequence[3]	REAL	2.730000e+002	2.730000e+002
12.0	sequence[4]	REAL	2.730000e+002	2.730000e+002
16.0	sequence[5]	REAL	2.730000e+002	2.730000e+002
20.0	sequence[6]	REAL	1.000000e+001	1.000000e+001
24.0	sequence[7]	REAL	1.000000e+001	1.000000e+001
28.0	sequence[8]	REAL	1.000000e+001	1.000000e+001
32.0	sequence[9]	REAL	0.000000e+000	1.000000e+001
36.0	sequence[10]	REAL	0.000000e+000	1.000000e+001
40.0	result[1, 3]	INT	5	5
42.0	result[1, 4]	INT	5	5

Press F1 for help. Stat. Data: 90 Dyn. Data: 0 Insert

SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.14



Information and Training Center
Knowledge for Automation

Краткий обзор

В приведенном выше примере в DB5 объявлены две переменные типа ARRAY с помощью редактора блока данных. Введение в DB новой переменной возможно только в представлении "Declaration view" (*View -> Declaration View*):

- sequence: ARRAY[1..10] OF REAL
- result: ARRAY[1..5, 3..7] OF INT

Инициализация массивов

Отдельным компонентам массива могут быть назначены значения при объявлении (не для параметров FC, in_out параметров FB или временных переменных). Когда компоненты инициализируются значением, оно должно быть совместимо с типом данных компонента. Инициализирующие значения вводятся в колонку "Initial Value", отделяются друг от друга запятой. Если несколько последовательных компонентов должны быть инициализированы одним и тем же значением, может использоваться коэффициент повторения. Он помещается перед инициализирующим значением, которое должно быть введена в круглых скобках.

Примеры

5 (1. 23467E + 002) // следующие 5 компонентов - инициализированы
// значением 123.467

5 (7,2,3) // следующие 15 компонентов последовательно
// инициализированы значениями 7, 2, 3

Результат инициализации может быть проверен или изменен в представлении "Data View" (*View -> Data View*). Если число инициализирующих значений меньше, чем число компонентов, то последние компоненты, которым не хватило инициализирующих значений получают значение "0".

Принятие инициализирующих значений

Если введены новые инициализирующие значения в представлении DB "Declaration View", они станут эффективными (имеющими силу как фактические значения) после того, как Вы выберете в представлении DB "Data View" опцию главного меню *Edit -> Initialize Data Block*.

Инициализирующие значения массивов в декларации входных и выходных параметров в FB принимаются как фактические значения в экземпляре DB, когда он создан.

Хранение переменных типа ARRAY в памяти

одномерный массив

- Тип данных BOOL

	7	6	5	4	3	2	1	0
Байт n ¹⁾	8	7	6	5	4	3	2	1
Байт n+1			и т.д.	12	11	10	9	

- Тип данных BYTE, CHAR

Байт n ¹⁾	Байт 1
Байт n+1	Байт 2
Байт n+2	Байт 3
	⋮

- Тип данных WORD, DWORD,...

Байт n ¹⁾	----- Слово 1 -----
Байт n+1	
Байт n+2	----- Слово 2 -----
Байт n+2	
	⋮

¹⁾ n = четное

многомерный массив

- Пример.
ARRAY[1..2,1..3,1..2] OF BYTE

Байт n ¹⁾	Байт 1.1.1
Байт n+1	Байт 1.1.2
Байт n+2	Байт 1.2.1
⋮	Байт 1.2.2
	Байт 1.3.1
	Байт 1.3.2
	Байт 2.1.1
	Байт 2.1.2
	Байт 2.2.1
	Байт 2.2.2
	Байт 2.3.1
	Байт 2.3.2

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.15



Information and Training Center
Knowledge for Automation

Краткий обзор

Точное место хранения в памяти переменных ARRAY необходимо тогда, когда, во время выполнения, к индивидуальным компонентам обращения происходит, используя косвенную адресацию через память или через регистры.

Хранение переменных

Переменная типа ARRAY всегда начинается с границы слова, то есть в байте четным адресом. Переменная ARRAY занимает память до следующей границы слова.

Компоненты с типом данных BOOL начинаются в бите с наименьшим адресом, компоненты с типом данных BYTE и CHAR в байте с четным адресом. Отдельные компоненты внесены в список последовательно. В многомерных массивах, компоненты хранятся по строкам, начиная с первого измерения. Новое измерение всегда начинается в следующем байте, если компонент байт или бит и в следующем слове, если компонент имеет другой тип данных.

Обратите внимание Адреса индивидуальных компонентов массива в DB показаны в представлении "Data View" в колонке "Address".

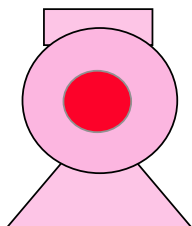
Тип данных: STRUCT

STRUCT (Структура):

- Группа компонентов различных типов данных

- Объявление:

```
StructName:  STRUCT
Имя_комп1: Тип_данных;
Имя_комп2: Тип_данных;
...
END_STRUCT
```



Motor: STRUCT	
Set_Speed:	REAL
Actual_Speed:	REAL
Enable:	BOOL
Disturbance:	BOOL
END_STRUCT	

Пример:

- Объявление переменных:

```
• MotorControl : STRUCT
  ON           : BOOL;
  OFF          : BOOL;
  SetSpeed     : INT;
  ActualSpeed  : INT;
END_STRUCT;
```

Доступ к переменным

```
S #MotorControl.ON
L #MotorControl.ActualSpeed
T #MotorControl.SetSpeed
...
```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.16



Information and Training Center
Knowledge for Automation

Краткий обзор

Тип данных STRUCT (Структура) представляет собой определенное число компонентов, каждый из которых может иметь различные типы. Структура может иметь до 8 уровней вложений.

Структура может быть объявлена в части объявлений логического блока, в глобальном DB или в определенном пользователем типе данных (UDT).

На типы данных компонентов структуры наложены следующие ограничения:

- Элементарный тип данных - никаких ограничений
- Сложный - DATE_AND_TIME, ARRAY, STRUCT, STRING, UDT
- Параметрические типы данных - невозможны
- Невозможен тип FB (в модели мультиэкземпляров)

Доступ к компонентам

STL-инструкции могут использоваться для доступа к компонентам структуры элементарных типов данных.

Компонент структуры адресуется, используя:

- Имя_Структуры. Имя_Компонента*

Точка должна быть вставлена между *Имя_Структуры* и *Имя_Компонента* как разделитель.

Если глубина вложения структуры большая, то есть компоненты структуры - в свою очередь являются структурами, то доступ к самым низким компонентам структуры возможен по следующей схеме:

- Имя_Структуры. Имя_Компонента. Имя_Подкомпонента. ...*

Точка должна быть вставлена между именами компонентов и подкомпонентов в каждом случае.

Объявление структур

Пример: Объявление массива структур с полями типа ARRAY

DB6 "Declaration View"

Address	Name	Type	Initial Value	Comment
0.0		STRUCT		
+0.0	Axis	ARRAY[1..4]		
*0.0		STRUCT		
+0.0	Start	BOOL	FALSE	
+0.1	Stop	BOOL	TRUE	
+2.0	Position	ARRAY[1..10]		
*0.0		STRUCT		
+0.0	Cutoffpoint_front	REAL	0.00	
+4.0	Cutoffpoint_back	REAL	0.00	
+8.0	Stoppingpoint	REAL	0.00	
=12.0		END_STRUCT		
=122.0		END_STRUCT		
=488.0		END_STRUCT		

DB6 "Data View"

Address	Name	Type	Initial Value	Actual Value
0.0	Axis[1].Start	BOOL	FALSE	FALSE
0.1	Axis[1].Stop	BOOL	TRUE	TRUE
2.0	Axis[1].Position[1].Cutoffpoint_front	REAL	0.000000e+000	0.000000e+000
6.0	Axis[1].Position[1].Cutoffpoint_back	REAL	0.000000e+000	0.000000e+000
10.0	Axis[1].Position[1].Stoppingpoint	REAL	0.000000e+000	0.000000e+000
14.0	Axis[1].Position[2].Cutoffpoint_front	REAL	0.000000e+000	0.000000e+000
18.0	Axis[1].Position[2].Cutoffpoint_back	REAL	0.000000e+000	0.000000e+000
22.0	Axis[1].Position[2].Stoppingpoint	REAL	0.000000e+000	0.000000e+000
26.0	Axis[1].Position[3].Cutoffpoint_front	REAL	0.000000e+000	0.000000e+000
30.0	Axis[1].Position[3].Cutoffpoint_back	REAL	0.000000e+000	0.000000e+000
34.0	Axis[1].Position[3].Stoppingpoint	REAL	0.000000e+000	0.000000e+000
38.0	Axis[1].Position[4].Cutoffpoint_front	REAL	0.000000e+000	0.000000e+000
42.0	Axis[1].Position[4].Cutoffpoint_back	REAL	0.000000e+000	0.000000e+000
46.0	Axis[1].Position[4].Stoppingpoint	REAL	0.000000e+000	0.000000e+000
50.0	Axis[1].Position[5].Cutoffpoint_front	REAL	0.000000e+000	0.000000e+000
54.0	Axis[1].Position[5].Cutoffpoint_back	REAL	0.000000e+000	0.000000e+000
58.0	Axis[1].Position[5].Stoppingpoint	REAL	0.000000e+000	0.000000e+000

SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.17



Information and Training Center
Knowledge for Automation

Краткий обзор

В приведенном выше примере в пределах DB6 объявлен, отдельный массив ARRAY [1 .. 4] с компонентами типа STRUCT ("Hall_1") с помощью инкрементного редактора блоков данных.

Структура в свою очередь состоит из трех компонентов, первые два из которых, то есть "START" и "STOP" имеют тип данных BOOL. Третий компонент имеет типов данных ARRAY [1 .. 10].

Компоненты этого типа данных ARRAY [1 .. 10] имеет в свою очередь тип STRUCT с компонентами типа REAL "Cutoffpoint_front", "Cutoffpoint_back" и "Stoppingpoint".

Доступ

Отдельные компоненты могут быть адресованы следующим образом, например:

- L "Hall_1".Axis [3].Position [7].Cutoffpoint_back
- S "Hall_1".Axis [2].START, и т.д.

Инициализация структуры

Отдельные компоненты структуры могут быть инициализированы начальным значением (столбец "Initial Value"). Не могут быть инициализированы следующие параметры или переменные :

- Любые параметры FC
- in_out параметры в FB
- Локальные данные в OB, FB и FC

Тип данных инициализирующего значения должен быть совместим с типом данных компонента.

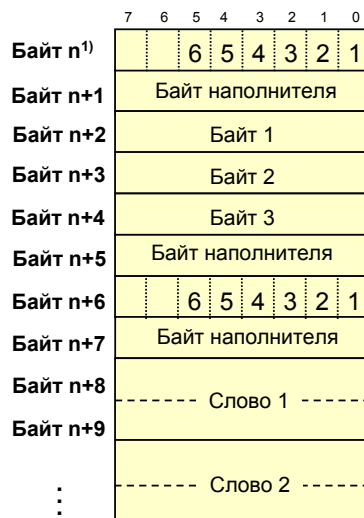
Принятие инициализирующих значений

Если введены новые инициализирующие значения в представлении DB "Declaration View", они станут эффективными (имеющими силу как фактические значения) после того, как Вы выберете в представлении DB "Data View" опцию главного меню *Edit -> Initialize Data Block*.

Инициализирующие значения структур в декларации входных и выходных параметров в FB принимаются как фактические значения в экземпляре DB, когда он создан.

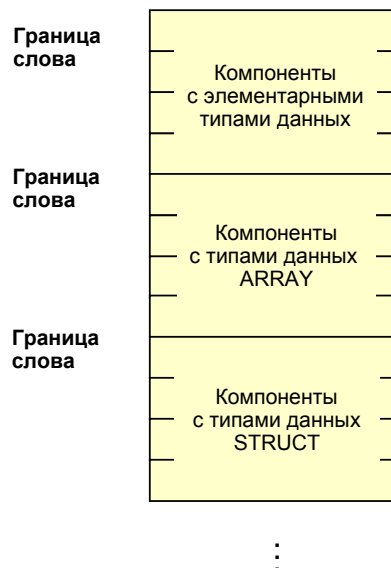
Хранение переменных типа STRUCT в памяти

Структуры с элементарными типами данных



¹⁾ n = четное

Структуры со сложными типами данных



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.18



Information and Training Center
Knowledge for Automation

Краткий обзор

Знание точного места хранения в памяти переменных STRUCT необходимо тогда, когда, во время выполнения обращение к компонентам происходит используя косвенную адресацию через память или через регистры.

Хранение переменных

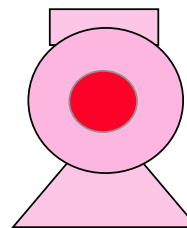
Переменная STRUCT всегда начинается с границы слова, то есть с байта с четным адресом. Далее отдельные компоненты расположены в памяти в последовательности их объявления. Переменная STRUCT занимает память до следующей границы слова.

Компоненты с типом данных BOOL начинаются на границе байта с младшего бита, компоненты с типом данных BYTE и CHAR на границе байта. Компоненты с другими типами данных всегда начинаются на границе слова.

Тип данных, определенный пользователем: UDT

UDT- тип данных, определенный пользователем:

- создается шаблон для дальнейшего использования в объявлениях
- доступен для всех блоков из программной папки



Пример:

- Определение нового типа данных (структуры):

```
UDT1 STRUCT
```

```
SetSpeed : REAL;
```

```
ActualSpeed : REAL;
```

```
Enable : BOOL;
```

```
Disturbance : BOOL;
```

```
END_STRUCT;
```

- Объявление переменных:

```
Motor_1: UDT1;
```

```
Motor_2: UDT1;
```

- Доступ к переменной:

```
L #Motor_1.ActualSpeed
```

```
UDT1: STRUCT
```

Set_Speed:	REAL
------------	------

Actual_Speed:	REAL
---------------	------

Enable:	BOOL
---------	------

Disturbance:	BOOL
--------------	------

```
END_STRUCT
```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.19



Information and Training Center
Knowledge for Automation

Краткий обзор

Когда одна и та же структура данных часто повторяется в программе пользователя или когда структуре данных должно быть дано собственное имя, тогда в STEP7 можно определить собственный, пользовательский тип данных (UDT = **U**ser **D**efined **D**ata **T**ype) (подобно *typedef* на языке высокого уровня "C" или *type* в "PASCAL").

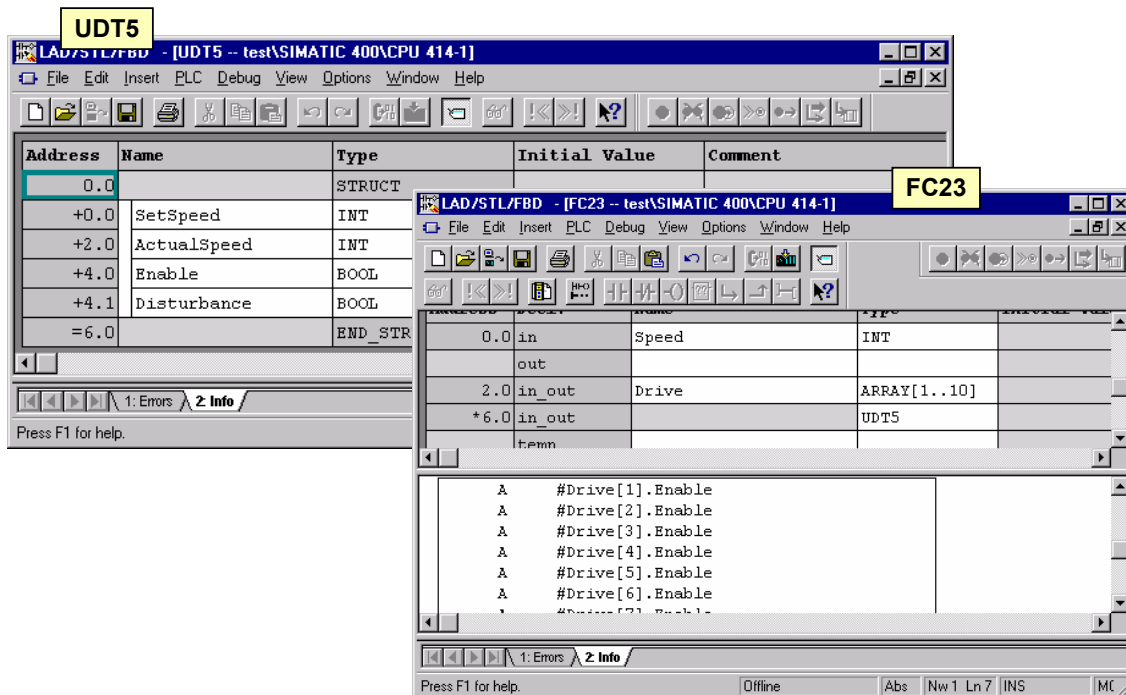
С помощью созданных типов данных решаемая задача может быть запрограммирована более эффективно. Пользователи могут тогда в проекте использовать типы данных, приспособленные к их проблеме.

Создание UDT

UDT создается DB-редактором или текстовым редактором и затем сохраняется в папке блоков, как блок (UDT1 ... UDT65535).

Символическое имя может быть назначено для UDT в таблице символов. Через UDT создается глобальный "шаблон", который можно использовать, так часто, как это необходимо при объявлении новых переменных или для создания глобального DB.

Использование UDT



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.20Information and Training Center
Knowledge for Automation

Краткий обзор

В приведенном выше примере создан UDT5 с 4-я компонентами (SetSpeed, ActualSpeed, Enable, Disturbance) для структуры двигателя и затем в разделе объявлений в FC23 создан in_out-параметр типа ARRAY с 10 компонентами типа UDT5.

Начальные значения для UDT

Пользователь определяет типы данных, дает им начальные значения и затем использует их в своей программе точно так же, как структуры. Структура UDT такая же, как и структура STRUCT. При создании UDT не создается переменных, которые могут обрабатываться в программе пользователя. UDT - "шаблон", который Вы можете использовать так часто, как Вам это необходимо для создания новых переменных. Также как в структуре, Вы имеете возможность установить начальные значения в UDT. Если UDT после этого используется для объявления переменных, содержание этих переменных инициализируется начальными значениями UDT (это не действительно для параметров в FC, in_out-параметров FB и временных переменных).

Создание DB

UDT может также использоваться как образец для создания (Диалог: *New Data Block*) глобального блока данных. В этом случае DB создается с той же самой структурой и с теми же начальными значениями, что и у соответствующего UDT.

Тип данных: DATE_AND_TIME

Структура:

Байт n ¹⁾	Год (90 ... 89)	Месяц (01 ... 12)	Байт n+1
Байт n+2	День (01 ... 31)	Часы (00 ... 23)	Байт n+3
Байт n+4	Минуты (00 ... 59)	Секунды (00 ... 59)	Байт n+5
Байт n+6	Миллисекунды (000 ... 999)	День недели (1..7)	Байт n+7

1=Воскресенье
 2=Понедельник
 3=Вторник
 4=Среда
 5=Четверг
 6=Пятница
 7=Суббота

- Все значения хранятся в BCD формате
- Предусстановки переменных:
 DT#Год-Месяц-День-Часы:Минуты:Секунды.[Миллисекунды]
 Пример: DT#1998-03-21-17:23:00:00
- Работа через функции IEC-библиотеки

¹⁾ n = четное

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.21



Information and Training Center
Knowledge for Automation

Краткий обзор

Тип данных DATE_AND_TIME представляет момент, состоящий из даты и "времени дня". Сокращение DT может также использоваться вместо DATE_AND_TIME.
 DATE_AND_TIME или DT - ключевые слова и поэтому могут также быть написаны строчными буквами.

Предусстановки

Переменная может быть предустановлена на начальное значение при объявлении (это не действительно для параметров в FC, in_out-параметров FB и временных переменных). Значение для предустановки должно иметь тип:

- DT #Год-Месяц-День-Часы:Минуты:Секунды. Миллисекунды
- Определение миллисекунд может быть опущено.

Использование

Переменные типа DATE_AND_TIME могут быть обработаны или с помощью абсолютного доступа к индивидуальным компонентам или с помощью функций IEC-библиотеки.

Обратите внимание

Текущее время дня из часов CPU в реальном масштабе времени может быть прочитано с помощью SFC1 (READ_CLK). Время возвращается SFC1 в выходном параметре типа DATE_AND_TIME.

Функции для работы с переменными типа DT

IEC-библиотека в Standard Library V3.x

- **FC1 (AD_DT_TM):** Функция FC 1 добавляет продолжительность времени (в формате TIME) к моменту (в формате DT) и возвращает новый момент (в формате DT) как результат.
- **FC34 (SB_DT_DT):** Функция FC 34 вычитает два момента (в формате DT) и возвращает продолжительность времени (в формате TIME) как результат.
- **FC35 (SB_DT_TM):** Функция FC 35 вычитает продолжительность времени (в формате TIME) из момента (в формате DT) и возвращает новый момент (в формате DT) как результат.
- **FC3 (D_TOD_DT):** Функция FC 3 объединяет данные в формате DATE и TIME_OF_DAY (TOD) и возвращает результат в формате DATE_AND_TIME (DT).
- **FC6 (DT_DATE):** Функция FC 6 извлекает данные формата DATE из формата DATE_AND_TIME.
- **FC7 (DT_DAY):** Функция FC 7 извлекает день недели из формата DATE_AND_TIME.
- **FC8 (DT_TOD):** Функция FC 8 извлекает данные формата TIME_OF_DAY из формата DATE_AND_TIME.
- **Функции сравнения для DT #Переменных:** FC9 (EQ_DT), FC12 (GE_DT), FC14 (GT_DT), FC18 (LE_DT), FC23 (LT_DT), FC28 (NE_DT)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.22Information and Training Center
Knowledge for Automation

Краткий обзор

При установке STEP7 также устанавливается Standard Library V3.x с под-библиотекой IEC Converting Blocks, которая содержит функции для обработки IEC -типов данных.

Функции для обработки переменных типа DATE_AND_TIME - также находятся в этой библиотеке.

FC1, FC35

При использовании FC1, FC34 и FC35 должны быть соблюдены следующие моменты:
момент времени (параметр T) должен находиться в диапазоне между DT #1990-01-01-00:00:00.000 и DT # 2089-12-31-23:59:59.999. Функция не выполняет проверку входных параметров .
Если параметры типа DT не находится в диапазоне, указанном выше, результат ограничен соответствующим значением, и BR-бит установлен "0".

FC34

Моменты должны находиться в диапазоне между DT # 1990-01-01-00:00:00.000 и DT # 2089-12-31-23:59:59.999. Функция не выполняет проверку входа. Если первый момент (параметр T1) больший, чем второй (параметр T2), результат положителен. Если первый момент менее, чем второй, результат отрицательный.
Если результат вычитания вне диапазона типа TIME, то он ограничен соответствующим значением и BR-бит установлен в "0".

FC3, FC6, FC7, FC8

Эти функции не сообщают о своих ошибках. Пользователь сам ответствен за назначение входным параметрам корректных значений .

Функции сравнения

Функции сравнения также не возвращают кода ошибки.
Функции сравнения возвращают результат сравнения в параметре RET_VAL: если сравнение выполнено, то RET_VAL = TRUE , а если нет - RET_VAL = FALSE.

Тип данных: STRING

Переменная типа STRING (строка) :

- Тип данных STRING - строка символов до 254 символов длиной
- Применение: обработка текстовых сообщений
- Объявление:
 - *Имя_Строки: STRING[maxNo]: 'Текст_инициализации'*
(Строка максимум из maxNo символов, maxNo: 0... 254)
 - *Имя_Строки: STRING: 'Текст_инициализации'*
(Строка максимум из 254 символов)

Примеры:

- Объявление переменной:
 - `Fault_signal : STRING 'Motor_failure_4'`
(Переменная *Fault signal* инициализируется указанным текстом)
 - `Warning : STRING[50] ''`
(“пустая” переменная *Warning*, может содержать до 50 символов)
- Обработка:
 - элементарный доступ:
`L #Fault_signal[5]` (загрузить 5-й символ из *Fault_signal*)
 - Обработка посредством FC из IEC- библиотеки

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.23Information and Training Center
Knowledge for Automation

Краткий обзор

Тип данных STRING (строка) используется для хранения строк символов (например, текстовых сообщений). Таким образом, простая "система обработки текстов (сообщения)" может быть осуществлена в S7-CPU. Тип данных STRING представляет строку символов (максимум 254 символа). Число, указанное в квадратных скобках в объявлении (1 .. 254) дает максимальное число символов, которые могут быть сохранены в переменной STRING. Если эта информация не указана, STL/LAD/FBD-редактор принимает длину, равную 254 символа.

К доступу к строковым переменным

К отдельным символам переменной STRING можно обращаться с помощью STL- инструкций для элементарных типов:

- `L StringName [5]` //Загружает 5-ый символ переменной

Фактическая обработка переменных STRING (текстов сообщений) возможна при использовании FC IEC-библиотеки.

Инициализация

При объявлении, переменные типа данных STRING могут быть "предустановлены" начальным текстом (это не относится к параметрам FC, и к in_out - параметрам FB, а также к временным переменным). Инициализация происходит в ASCII-кодах символов. Если должны быть включены специальные символы для управления, то перед ними должен быть помещен знак доллара (\$) .

Специальные символы:

- \$\$ Простой символ доллара
- \$L, \$I Выравнивание подачи (LF)
- \$P, \$p Подача страницы
- \$R, \$r Перевод каретки
- \$T, \$t Табулятор

Хранение строковых переменных в памяти

Пример:

- **Объявление с инициализацией**
 - Given_name: STRING[8]: 'OTTO'
- **Хранение строковой переменной "Given_name"**

Байт n ¹⁾	max длина= 8
Байт n+1	Текущая длина= 4
Байт n+2	1-й символ= 'O'
Байт n+3	2-й символ = 'T'
Байт n+4	3-й символ = 'T'
Байт n+5	4-й символ = 'O'
Байт n+6	B#16#00
Байт n+7	B#16#00
Байт n+8	B#16#00
Байт n+9	B#16#00
	⋮

→ Определяет максимальное число сохраняемых знаков, то есть длина, указанная в декларации

→ Определяет число знаков в настоящее время сохраненных в переменной типа STRING

- Информация относительно максимального числа сохраняемых знаков или относительно текущей длины может быть оценена функциями IEC-библиотеки.

¹⁾ n = четное

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.24



Information and Training Center
Knowledge for Automation

Краткий обзор

Переменная типа STRING занимает в памяти максимум 256 байтов, из которых максимум 254 байта занимают "чистые данные", то есть символы, которые сохраняет пользователь.

Хранение переменных

Переменные STRING всегда начинаются на границе слова, то есть в байте с четным адресом.

При создании переменной максимальная длина вводится в первый байт переменной согласно объявлению. Аналогично, при инициализации или в процессе обработки используемая в настоящее время длина, т.е. длина сохраненной строки символов вводится во второй байт.

Оба числа требуются функциями IEC-библиотеки при обработке переменных STRING.

Символы сохраняются в ASCII-коде. Незанятые байты в переменной STRING заполнены при инициализации B # 16 # 00.

Передача параметра Переменные типа STRING можно передавать в блоки точно так же, как переменные типа ARRAY или STRUCT. Они должны иметь ту же длину, что и в объявлении формального параметра.

Передача фактических значений типа STRING параметрам типа POINTER или ANY FC или FB также возможна.

Functions for Processing STRING Variables

IEC-библиотека в Standard Library V3.x

- **FC2 (CONCAT):** Функция FC2 объединяет две переменные типа STRING в одну строку.
- **FC4 (DELETE):** Функция FC 4 удаляет L символов от P-го символа в строке
- **FC11 (FINF):** Функция FC 11 находит положение второй строки в пределах первой строки.
- **FC17 (INSERT):** Функция FC 17 вставляет строку из параметра IN2 в строку из параметра IN1 после P-го символа.
- **FC20 (LEFT):** Функция FC 20 поставляет первые L символов строки (левая часть строки).
- **FC21 (LEN):** Функция FC 21 возвращает текущую длину строки (в байтах).
- **FC26 (MID):** Функция FC 26 поставляет среднюю секцию строки
- **FC31 (REPLACE):** Функция FC 31 заменяет L символов первой строки (IN1) от P-го символа (включительно) символами второй строки (IN2).
- **FC32 (RIGHT):** Функция FC 32 поставляет правые L символов строки (правая часть строки).
- **Функции сравнения для переменных типа STRING:** FC10 (EQ_STRING), FC13 (GE_STRING), FC15 (GT_STRING), FC19 (LE_STRING), FC24 (LT_STRING), FC29 (NE_STRING)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.25Information and Training Center
Knowledge for Automation

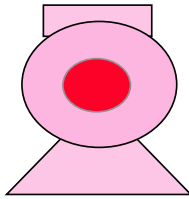
Краткий обзор

При установке STEP 7 также устанавливается Standard Library с под-библиотекой IEC Converting Blocks, которая содержит функции для обработки IEC -типов данных.

Обратите внимание Функции, в общем, выполняют оценку ошибки с помощью максимальной длины или фактической используемой длины строки. Если функция обнаруживает ошибку, то, вообще, BR-бит устанавливается в "0". Детальное описание индивидуальных функций может быть найдено в диалоговой помощи для IEC-библиотеки.

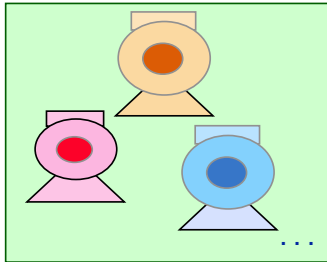
Функции сравнения Функции сравнения исполняет лексикографическое сравнение символов строки. Сравнение символов происходит, начиная слева, по их ASCII-кодам (например 'a' больше, чем 'A' и 'A' меньше, чем 'B'). Первый отличающийся символ определяет результат сравнения. Если левая часть более длинной строки идентична более короткой строке, то более длинная строка является большей. Функции не возвращают кода ошибки. Результат функции сравнения возвращают в параметре RET_VAL : если сравнение выполнено - RET_VAL = TRUE, если нет - RET_VAL = FALSE.

Упражнение 5.1: Использование сложных типов данных



UDT99 "Motor"

Address	Name	Type	Initial Value
0.0		STRUCT	
+0.0	SetSpeed	REAL	0.000000e+000
+4.0	ActualSpeed	REAL	0.000000e+000
+8.0	SetActDiffMax	REAL	5.000000e-002
+12.0	Enable	BOOL	FALSE
+12.1	Disturbance	BOOL	TRUE
=14.0		END_STRUCT	



Hall_1

DB51 "Conv_area_Motors"

Address	Name	Type	Initial Value
0.0		STRUCT	
+0.0	ConvArea_1_Motor	ARRAY[1..20]	
+14.0		UDT99	
+280.0	ConvArea_2_Motor	ARRAY[1..20]	
+14.0		UDT99	
=560.0		END_STRUCT	

SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.26



Information and Training Center
Knowledge for Automation

Цель упражнения: Ознакомление с данными сложного типа STRUCT, а также с обработкой с помощью UDT.

Задача В конвейере мукомольного предприятия есть 20 двигателей одинакового типа. По этой причине для управления конвейером нужно создать переменную типа ARRAY.

С другой стороны, структура данных для каждого отдельного двигателя идентична, так что можно однажды создать UDT.

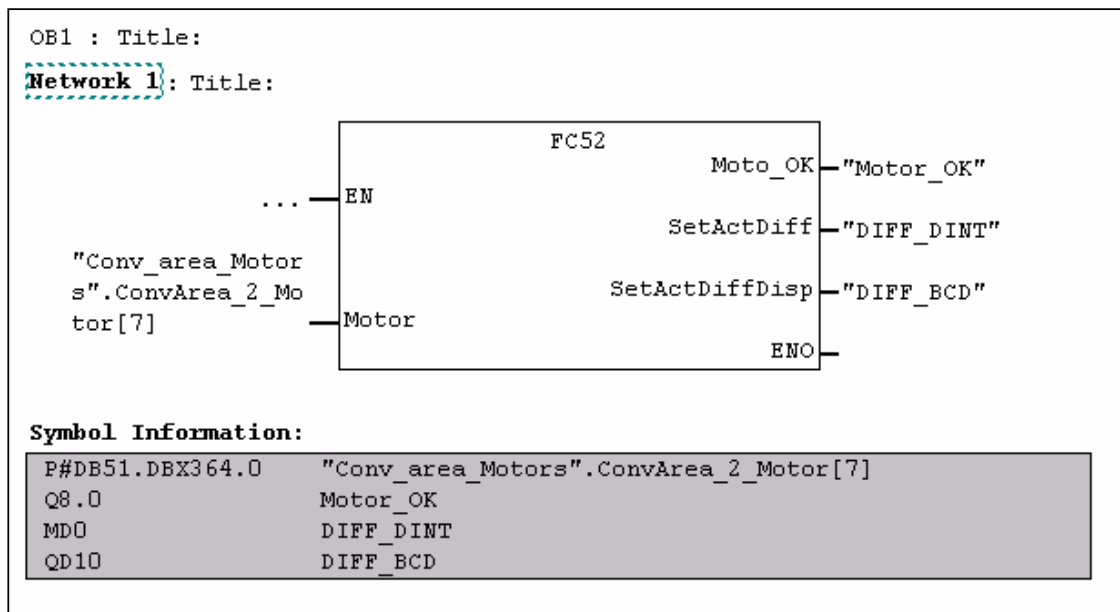
Запись данных двигателя состоит из следующей информации:

- *SetSpeed* (REAL): Скорость, указанная диспетчером
- *ActualSpeed* (REAL): Измеренная фактическая скорость
- *SetActDiffMax* (REAL): Максимальный процент отклонения между заданной и фактической скоростью, указанной в соответствии с гарантией качества
- *Enable* (BOOL): Сигнал разрешения, подаваемый диспетчером
- *Disturbance* (BOOL): ОК сигнал, возвращаемый диспетчеру.

Что делать

1. Прежде всего создайте UDT99 "Motor" с необходимой структурой.
2. Инициализируйте UDT99 следующим образом:
 - *SetSpeed*: 0.0
 - *ActualSpeed*: 0.0
 - *SetActDiffMax*: 0.05 (соответствует максимальному отклонению 5 %)
 - *Enable*: FALSE
 - *Disturbance*: TRUE
3. Создайте DB51 "Conv_Area_Motors". В пределах DB51 объявите две, переменные ConvArea_1_Motor и ConvArea_2_Motor типа ARRAY[1..20] с компонентами типа UDT99.
4. Проверьте содержимое DB51 в представлении *data view*.

Упражнение 5.2: Доступ к сложным типам данных



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.27Information and Training Center
Knowledge for Automation

Цель упражнения: Ознакомление с доступом к параметрам и переменным сложного типа данных, также объявленных как UDT.

Задача Операционные режимы отдельных двигателей должна быть проверена в функции FC52.

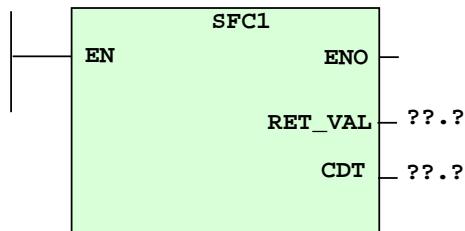
FC52 имеет следующие свойства:

- FC52 ожидает запись данных любого двигателя в входном параметре **#Motor** (UDT99).
- FC52 предоставляет, в выходном параметре **#Motor_OK** (BOOL), TRUE, если бит Disturbance не установлен, и процент отклонения между *SetSpeed* и *ActualSpeed* - меньше, чем *SetActDiffMax* в записи данных для двигателя, которая поступает на вход.
- Кроме того, FC52 возвращает разность между *SetSpeed* и *ActualSpeed* как DINT-число или как BCD-число в выходных параметрах **#SetActDiff** (DINT) и **#SetActDiffDisp** (DWORD).
- В случае переполнения при преобразовании к DINT или DWORD REAL чисел, FC52 устанавливает BR-бит в "0".

Что делать

1. Создать FC52 с необходимыми свойствами
2. Вызвать FC52 в OB1. Снабдите параметр входа **#Motor** записью данных для 7-го двигателя из *ConvArea_2*. Вывести выходной сигнал **#Motor_OK** на выход Q8. 0. Также снабдите выходные параметры **#SetActDiff** и **#SetActDiffDisp** с фактическими параметрами MD0 и QD10 (цифровой показ)
3. Загрузить блоки к CPU.
4. Проверить FC52 с помощью функции "Monitor/Modify Variable", определяя различные значения в записи.

Дополнительное упражнение 5.3: Чтение времени и даты с помощью SFC 1 (READ_CLK)



Параметры

Параметры	Описание	Тип данных	Обл. памяти	Комментарий
CDT	OUTPUT	DATE_AND_TIME (DT)	D, L	Выход для текущего времени дня и даты
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Код ошибки

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_05E.28



Information and Training Center
Knowledge for Automation

Цель упражнения: Ознакомиться со сложным типом данных DATE_AND_TIME.

Задача

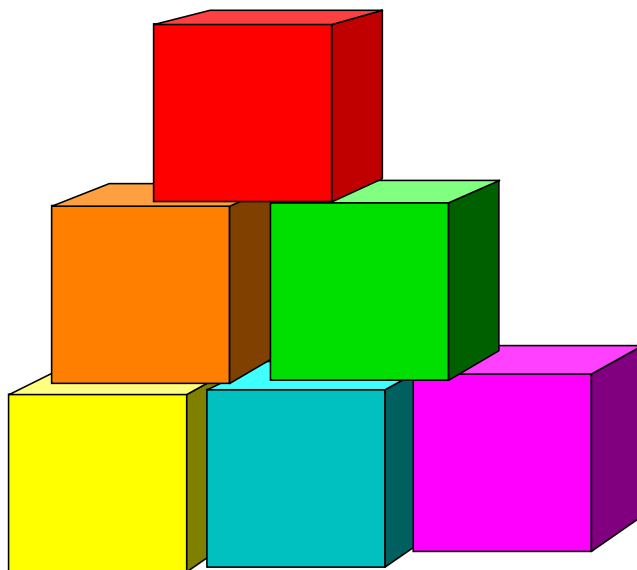
Создайте FC53 со следующими функциональными возможностями:

- FC53 возвращает текущее " время дня " с помощью системной функции SFC51.
- Часы и минуты выведите на цифровой дисплей.

Что делать

1. Создайте блок FC53 с вышеупомянутыми функциональными возможностями.
2. Вызовите FC53 в OB1.
3. В SIMATIC Manager проверьте с помощью выбора меню *PLC -> Set Time and Date*, правильно ли установлены часы CPU.
4. Загрузите FC53 и OB1 в CPU.
5. Проверьте программу.

Вызов блоков и модель мультиэкземпляров



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.1Information and Training Center
Knowledge for Automation

Содержание

Стр

Блоки для структурированного программирования	2
Обзор блоков STEP 7	3
Свойства функции	4
Механизм передачи элементарных типов данных	5
Вызов функций с параметрами сложных типов	6
Характеристики вызова функций	7
Свойства функциональных блоков	8
Формирование экземпляров функционального блока	9
Передача параметров при вызове FB	10
Вызов FB со сложными типами данных	11
Характеристики вызова функционального блока	12
Упражнение 6: Модель конвейера как установка для розлива	13
Упражнение 6.1a: Установка розлива - способ секций	14
Упражнение 6.1b: Установка розлива - конвейер	15
Структура модели мультиэкземпляров	16
Объектно-ориентированное программирование с использованием мультиэкземпляров	17
Осуществление "Линии прессов" в STEP 7	18
Свойства модели мультиэкземпляров	19
Упражнение 6.2: Модель конвейера как сборочная линия	20
Упражнение 6.2a: Структура программы для рабочего места	21
Метод функционирования FB1 "Station"	22
Метод функционирования FB2 "Transport"	23
Упражнение 6.2b: Расширение до 3 рабочих мест	24
Соединение параметров блоков	25

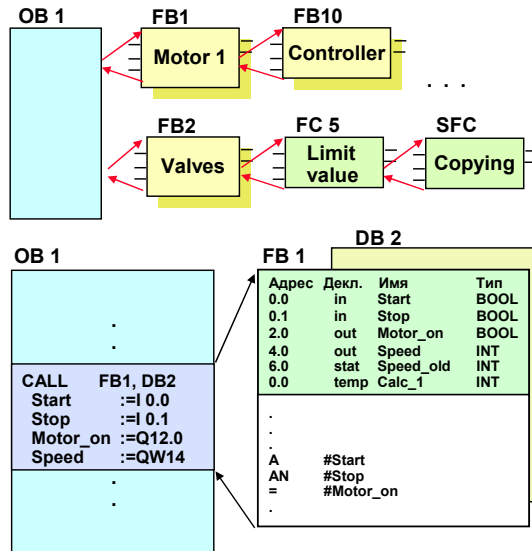
Блоки для структурированного программирования

Модуляризация полной задачи:

- Частные задачи решаются в их собственных блоках
- Назначение параметров дает гибкость в использовании
 - Пример: Цикл бурения с параметром для глубины

Повторное использование блоков:

- Блоки могут вызываться так часто, как это требуется
- Ограничения:
 - Не должно быть доступа к глобальным адресам
 - Связь только через параметры



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.2Information and Training Center
Knowledge for Automation

Что такое структурированная программа?

Когда в управляемом процессе встречаются подобные или идентичные функции, Вы ищете общее, решение и описываете его в параметрируемом блоке.

Этот блок неоднократно вызывается в программном цикле. При этом ему передаются фактические параметры, необходимые при данном вызове.

Преимущества

Разделение программы пользователя на индивидуальные блоки имеет следующие преимущества:

- общая сложная задача может быть разделена в меньшие, более ясные частные задачи. Блоки для частных задач могут быть созданы и проверены независимо друг от друга.
- с помощью параметров блоки могут быть разработаны гибкими в использовании.

Например, блок для бурения может быть создан с параметрами для координат и глубины отверстия.

- Блоки могут вызываться так часто, как это требуется, с различными фактическими параметрами.

Неограниченный вызов блока возможен только тогда, если внутри блока нет никакого вызова адресов, типа входов, выходов, меркеров или переменных из DB.

Блоки должны общаться с "внешним миром" исключительно через свои параметры.

- Блоки в STEP 7 могут быть изменены и згружены в CPU во время выполнения независимо друг от друга. Тем образом, Вы можете, например, модернизировать программное обеспечение системы, в то время как оно находится в действии.
- Для многих специальных задач могут быть поставлены во взаранее созданных стандартных библиотеках.
- Блоки для часто используемых стандартных задач могут быть объединены изготовителем в операционной системе CPU в форме системных функций (системные блоки (SFB) или функции (SFC)).

Обзор блоков STEP 7

Тип блока	Свойства
Организационный блок (OB)	- пользовательский интерфейс - система приоритетов (0..28) - специальная стартовая информация в локальном стеке данных
Функциональный блок (FB)	- параметризуемый - с памятью
Функция (FC)	- параметризуемая (параметры назначаются при вызове) - может иметь возвращаемое значение - без памяти
Блок данных (DB)	- хранение структурированных локальных данных (экземпляр DB) - хранение глобальных данных (доступен из любого места программы)
Системный функциональный блок (SFB)	- FB (с памятью) хранящийся в операционной системе CPU и вызываемый пользователем
Системная функция (SFC)	- FC (без памяти) хранящийся в операционной системе CPU и вызываемый пользователем
Системный блок данных (SDB)	- блок данных для данных конфигурации и параметров

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.3Information and Training Center
Knowledge for Automation

Краткий обзор

Для разделения сложной задачи автоматизации в меньшие частные задачи, STEP 7 предлагает ряд различных блоков. Эти частные задачи отражают технологические функции системы или процесса.

Классы блоков в STEP 7

Блоки, их функции, их структура и применение обусловлены программой пользователя.

Блоки в STEP 7 могут - в соответствии с их содержанием - быть разделены на два класса:

- Логические блоки:

Логические блоки - организационные блоки (OB), функциональные блоки (FB), функции (FC), а также системные функциональные блоки (SFB) и системные функции (SFC).

В этих блоках содержатся инструкции программы пользователя.

- Блоки данных:

Блоки данных - пользовательские блоки данных (DB) и системные блоки данных (SDB).

Пользователь может хранить данные в время выполнения программы и получать доступ к этим данным в более позднее время.

Содержание системных блоков данных (SDB) использует исключительно CPU (назначения параметров). SDB создаются не программой пользователя, а инструментами типа HW-CONFIG или NETPRO.

Свойства функции

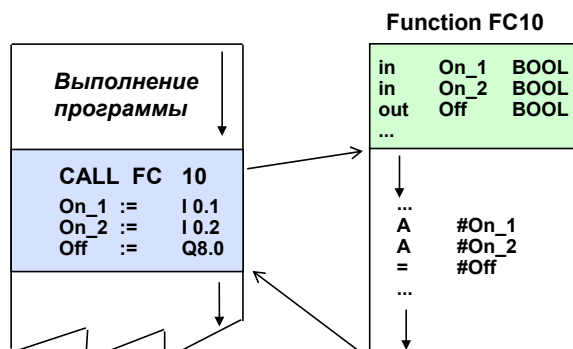
Параметрируемые блоки:

- Параметров типа in, out, и in_out может быть сколько требуется
- Без памяти, то есть только временные переменные

Соответствующие требования IEC 1131-3:

- Так много входных параметров, как это требуется
- Только один выходной параметр RET_VAL
- Нет доступа к глобальным переменным и абсолютным адресам

Расширяют набор инструкций процессора



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.4Information and Training Center
Knowledge for Automation

Краткий обзор

Функции представляют параметрируемые блоки без памяти. В STEP 7 они могут иметь так много входных параметров, выходных параметров и входных/выходных параметров, как это требуется. Функции не имеют никакой памяти, т.е. не существует никакой отдельной, постоянной области данных для хранения результатов. Временные результаты, которые доступны в течение выполнения функции, могут быть сохранены только во временных переменных соответствующего локального стека данных.

Функции расширяют набор инструкций процессора.

Применение

Функции прежде всего используются, когда значения функции должны быть возвращены вызывающим блоком. (Например, математические функции).

Соответствие функциям IEC-1131

Если функции должны соответствовать IEC 1131-3, то необходимо соблюдать следующие правила:

- Функции могут иметь так много входных параметров, как это требуется. Однако, они должны возвращать результат только в одном выходном параметре RET_VAL.
- Глобальные переменные не могут ни читаться, ни быть записаны в пределах функции.
- Абсолютные адреса не могут ни читаться, ни быть записаны в пределах функции.
- Никакие экземпляры функциональных блоков не могут вызываться в пределах функции.

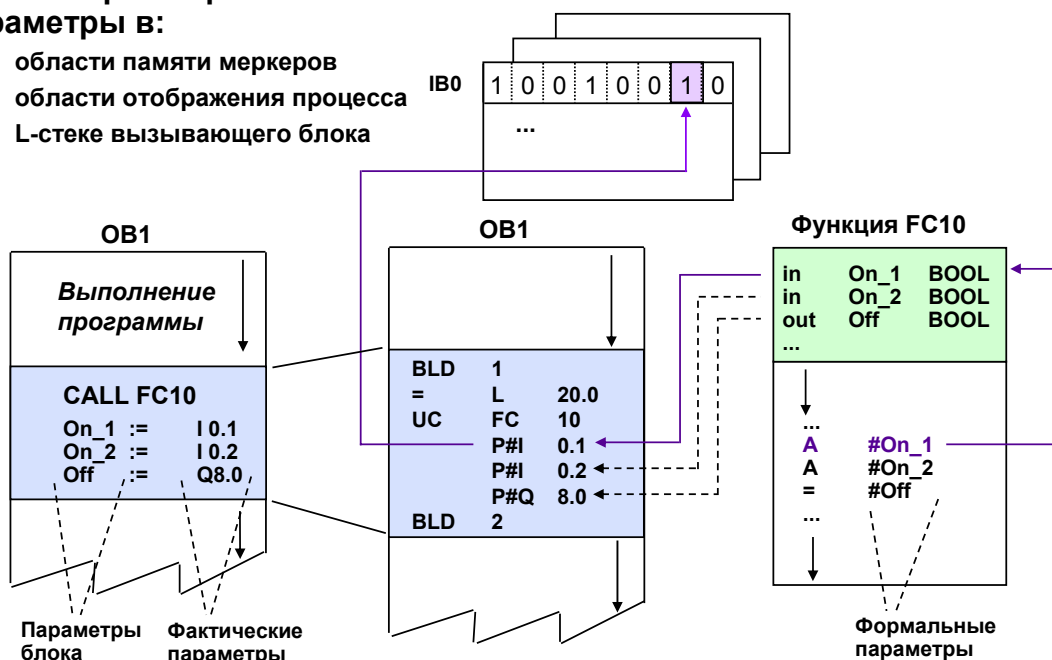
Из-за отсутствующей "памяти", возвращаемый результат соответствующей этим нормам функции, исключительно зависит от значений входных параметров. При одинаковых входных параметрах, функция также возвращает одинаковый результат.

It is therefore up to the programmer to create norm-conforming functions or to do the block programming and structuring in STEP 7 as it is in STEP 5.

Механизм передачи элементарных типов данных

Элементарные фактические параметры в:

- области памяти меркеров
- области отображения процесса
- L-стеке вызывающего блока



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.5Information and Training Center
Knowledge for Automation

Параметры FC

Данные для обработки могут быть переданы вызываемой функции. Эта передача данных имеет место исключительно через список параметров, который появляется после CALL. Имена и типы данных параметров блока, которые появляются, объявлены в части декларации FC.

Могут быть объявлены входы (только для чтения), выходы (только для записи) и параметры входа-выхода (чтение и запись).

Номер параметров неограничен (хватило бы места в памяти!), имена могут содержать максимум 24 символа. Кроме того, параметры можно снабдить детальным комментарием. Если блок не имеет параметров, то их список отсутствует при вызове FC.

Механизм передачи

Начиная выполнять команду CALL, STL/LAD/FBD-редактор сначала вычисляет указатели с указанием области для всех фактических параметров из списка формальных параметров и сохраняет их сразу после вызова инструкции FC.

Если теперь имеет место в пределах FC доступ к формальному параметру (например: `A #On_1`), CPU определяет инструкцию вызова FC от адреса возврата, сохраненного в В-стеке. Из списка параметров FC определяет указатель с указанием области на фактические параметры, которые поставлены в соответствие формальным параметрам. Доступ к фактическим параметрам тогда осуществляется через этот указатель.

Этот механизм передачи называется "передача по ссылке".

Этот механизм передачи через указатель влечет за собой следующее:

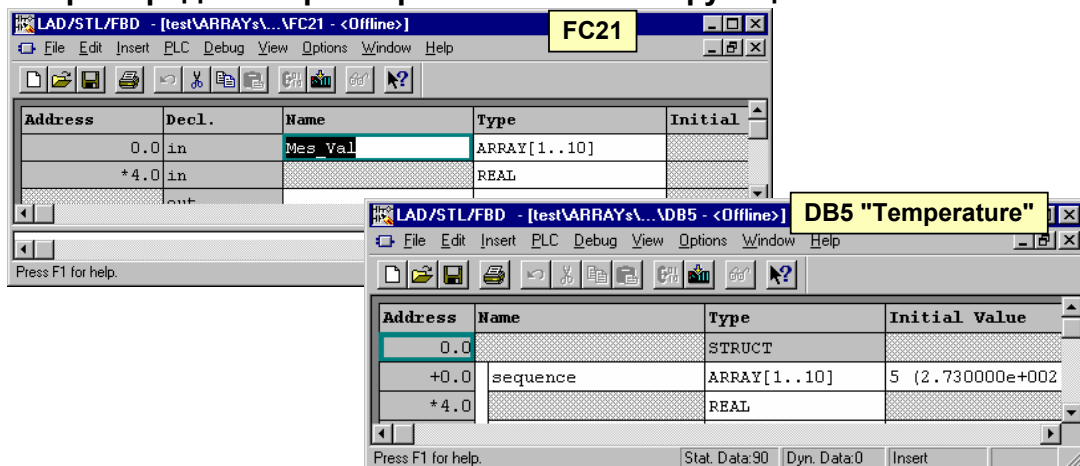
- Все параметры блока должны быть назначены при вызове FC.
- Параметры блока не могут быть инициализированы в декларации.

Замечание

Если блок параметр поставлен в соответствие фактическому параметру из DB или если используются сложные типы данных, то передача параметра становится более сложной. (см. Приложение).

Вызов функций с параметрами сложных типов

Пример: Передача параметра типа ARRAY в функцию



Назначение параметров сложных типов возможно только символически

Network 1: Mes_Val объявлен как array в FC21

```
CALL FC 21
  Mes_Val := "Temperature".sequence
```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.6



Information and Training Center
Knowledge for Automation

Краткий обзор

Параметры сложного типа данных (ARRAY и STRUCT) предлагают ясный и эффективный путь для передачи больших количеств связанных данных между вызывающим и называемым блоками. Массив или структуру можно передать в вызываемую функцию как полную переменную.

Назначение параметров

Передаваемый параметр должен иметь тот же самый тип данных, что и фактический параметр. Параметры сложных типов (ARRAY, STRUCT, DATE_AND_TIME и STRING) могут передаваться только символически. Так как переменные сложного типа данных могут находиться только в блоках данных или в локальном стеке данных, фактический параметр должен или быть расположен в блоке данных (глобальном или экземпляре DB) или в локальном стеке данных вызывающего блока. После того, как редактор STL/LAD/FBD проверил совместимость типов данных фактического параметра и параметра, объявленного в блоке, только тогда создается POINTER с номером DB и межзонным указателем на фактический параметр, передаваемый в FC. Этот POINTER помещается в L-стек вызывающего блока (V-область) с помощью макровывоза. Этот POINTER важен для программиста, в частности, когда к переданному параметру нужно обращаться косвенно (см. Приложение).

Обратите внимание

- Число занятых локальных данных может быть определено через меню *View -> Block Properties*.
- Компоненты ARRAY или STRUCT можно также передавать в блоки, если соответствующий параметр блока и компоненты ARRAY или STRUCT имеют тот же самый тип данных.

Характеристики вызова функций

Инструкция CALL

- **Инструкция является макрокомандой**
 - Содержание регистров может быть изменено, даже DB-регистра
 - Сохранение в В-стеке
 - После вызова открывается другой DB
 - Время выполнения CALL зависит от числа и расположения в памяти фактических параметров
- **Инструкция CALL проверяет правильность назначения блоку формальных параметров**
- **Пример:**
 - CALL FC10
 - On_1 := I 0.1
 - On_2 := I 0.2
 - Off := Q8.0

Инструкции вызова UC и CC

- **Независимый от RLO (UC) или зависимый от RLO (CC) вызов блока**
 - Пример: UC FC20 or CC FC20
- **Только для FC без параметров**

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.7Information and Training Center
Knowledge for Automation

Инструкция CALL

Инструкция (макро) CALL должна использоваться для вызова блоков (FC, SFC, FB и SFB).

В вызове FC, прямой информационный обмен между вызывающим блоком и вызываемой функцией возможен только через CALL. CALL проверяет, что формальные параметры блока правильно снабжены фактическими.

CALL реализована как макрокоманда, т.е. состоит из несколько STL инструкций.

Если формальный параметр назначен с адресом, который находится в DB, то передача параметра имеет место с помощью регистра DB (см.

Приложение). Из сказанного следует:

- В пределах называемого FC, возможно, что, DB, который открыт - не тот DB, который был открыт перед CALL.
- Если CPU переходит в STOP в течение обработки вызываемого FC, то значение DB-регистра, показанное в В-стеке, то, которое заменено при назначении параметра.
- Если после обработки сделан переход назад в вызывающий блок, возможно, что DB который был открыт перед CALL, больше не открыт.

Инструкции UC, CC

Блоки могут также вызываться инструкциями UC или CC. Инструкция вызова UC - абсолютная инструкция, то есть UC всегда вызывает блок независимо от условий (например: UC FC20).

Инструкция вызова CC - условная инструкция, то есть CC вызывает блок только когда RLO равен "1". Если RLO равен "0", то CC не вызывает блок и устанавливает RLO в "1". Обрабатывается следующая инструкция после команды CC.

Важно

UC и CC могут использоваться только для вызова FC без параметров.

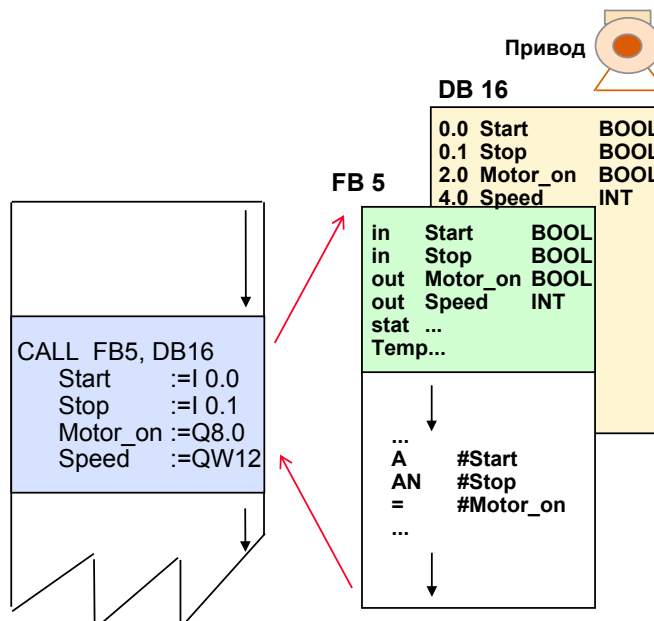
Свойства функциональных блоков

Параметрируемые блоки:

- Соответствуют IEC 1131-3
- Параметров типа in, out и in_out столько, сколько требуется
- С памятью, то есть не только временные, но и статические переменные
- Вызывается с собственной областью данных
- "Инкапсуляция данных"

Приложения:

- Функции таймеров и счетчиков
- Управление процесса со внутренними состояниями
 - Котлы
 - Приводы, вентили, и т.д.



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.8Information and Training Center
Knowledge for Automation

Краткий обзор

Функциональные блоки (FB) - программные пользовательские блоки - представляют, согласно стандарту IEC 1131-3, логические блоки с памятью. Они могут вызываться из OB, FB и FC.

Функциональные блоки могут иметь любое количество входов, выходов и параметров типа вход / выход, а также статические и временные переменные.

В отличие от FC, FB - блоки с назначенной собственной частной областью данных, в которой FB, например, может "помнить" состояние процесса от вызова до вызова. В самой простой форме, эта частная область данных - собственный DB, так называемый экземпляр DB.

"Память"

Программист имеет возможность объявить статические переменные в разделе объявлений функционального блока. Функциональный блок может "помнить" информацию между вызовами в своих переменных.

Возможность функционального блока "помнить" информацию между вызовами - существенное отличие от функций.

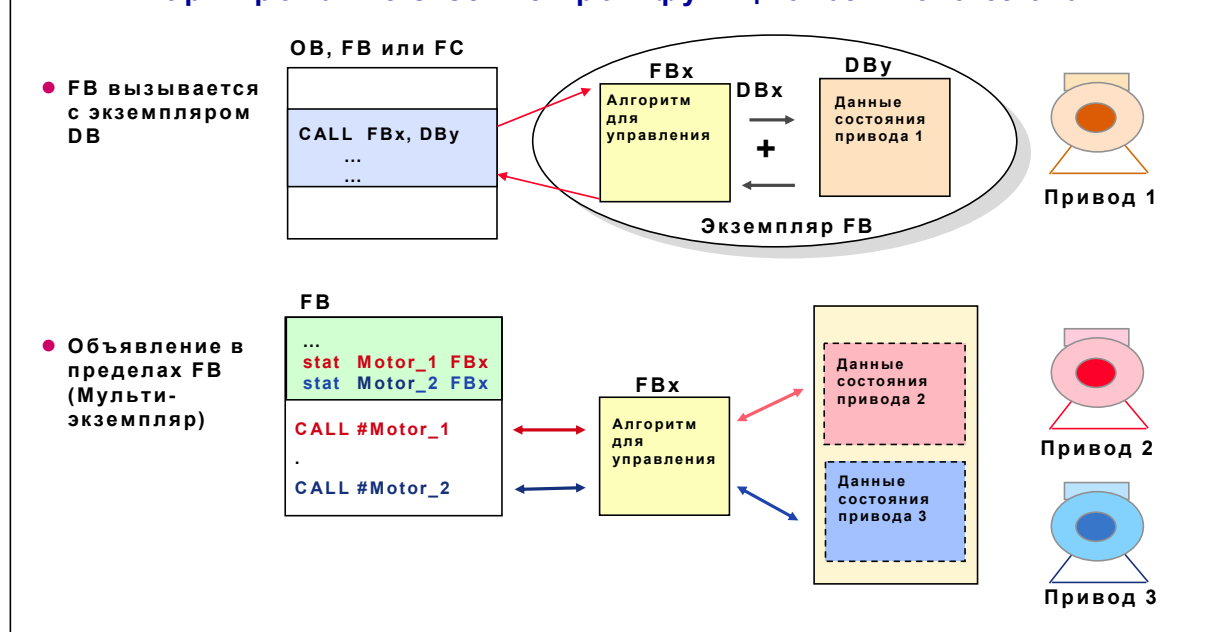
Применение

С помощью этой "памяти", функциональный блок может, например, осуществлять функции счетчика и таймера или управлять объектами, типа двигателей, котлов и т.д.

В частности, функциональные блоки хорошо приспособлены для управления всеми теми процессами, чья работа зависит не только от внешних влияний, но также и от внутреннего состояния, типа скорости, температура и т.д.

При управлении такими процессами, внутренние данные о его состоянии тогда копируются в статические переменные функционального блока.

Формирование экземпляров функционального блока



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.9



Information and Training Center
Knowledge for Automation

Что такое экземпляр?

Концепция экземпляров функциональных блоков имеет большое значение и составляет существенный отличительный критерий от FC. Введение переменных в языке высокого уровня типа "C", согласно объявленным имени и типу данных в декларации называется "формированием экземпляра".

Точно так же, как переменные, функциональные блоки - также "снабжаются экземпляром". Только через эту 'собственную' область данных, в которой сохраняются значения параметров блока и статические переменные, FB становятся выполнимой единицей (FB-экземпляр). Управление физическим объектом, типа двигателя или котла, имеет место с помощью экземпляра FB, то есть функционального блока с назначенной областью данных. Необходимые данные для процесса управления сохранены в этой области данных.

Создание экземпляра FB

Создание экземпляра FB, то есть назначение собственной области памяти в вызове FB, может быть сделано в STEP 7 двумя способами :

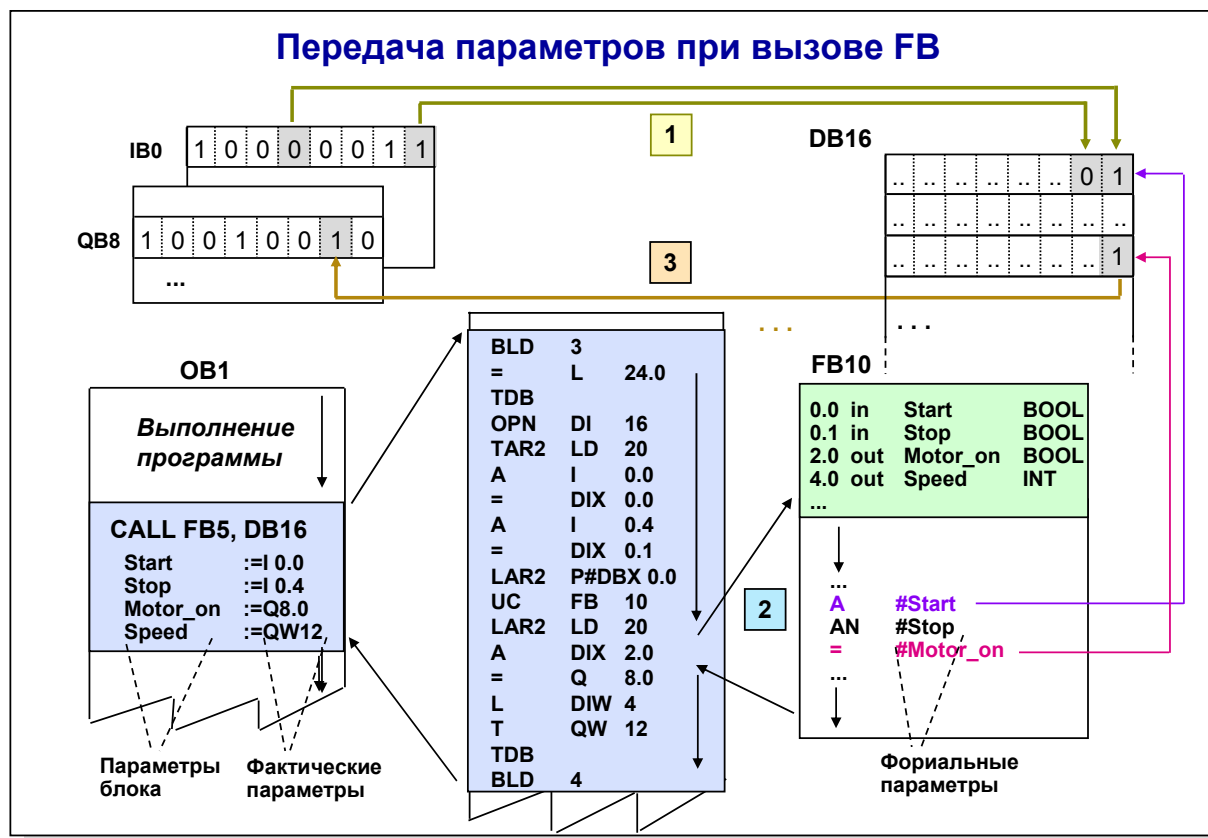
- Через явное объявление так называемых экземпляров блоков данных, когда вызывается функциональный блок.
- Через явное объявление экземпляра функционального блока в функциональном блоке более высокого уровня (модель мульти-экземпляров).

STEP 7 в этом случае проверяет, что область данных, требуемая для экземпляра создана в пределах области данных FB более высокого уровня.

Преимущества

Концепция экземпляров в STEP 7 имеют следующие преимущества:

- При вызове FB, не нужно принимать никаких мер для сохранения и администрирования локальных данных.
- Функциональный блок может по концепции экземпляров использоваться несколько раз . Если, например, несколько двигателей того же самого типа должны управляться, то это можно сделать, используя несколько экземпляров FB. Данные состояния отдельных двигателей сохранены в статических переменных FB.



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.10Information and Training Center
Knowledge for Automation

Параметры в вызове FB

Данные для обработки могут быть переданы называемому экземпляру FB. Передача этих данных происходит через список параметров, который появляется после CALL. Тип (in, out или in / out), имя и тип данных параметра установлен в разделе описаний FB.

В отличие от вызова FC, все параметры элементарных типов данных могут не назначаться в вызове FB. Причина этого находится в механизме передачи фактических параметров вызываемому FB.

Механизм передачи

Если экземпляр DB создан для FB, редактор блоков автоматически резервирует память для параметров блока (входы, выходы и параметры типа вход-выход) и для статических переменных, объявленных в разделе описаний FB. Адреса параметров и статических переменных в экземпляре DB - точно те, которые найдены в первой колонке раздела описаний FB.

При вызове FB-экземпляра, использующем макрокоманду CALL, экземпляр-DB открывается, используя регистр DI, и значения текущих параметров типа in и in_out копируются в экземпляр-DB перед фактической обработкой FB.

Если теперь имеет место доступ к формальным параметрам во время обработки FB, то происходит доступ к адресам, принадлежащим экземпляру-DB. Этот доступ имеет место, используя внутризонную регистровую косвенную адресацию, используя регистры DI и AR2.

После обработки FB, значения формальных параметров типа out и in_out копируются в фактические параметры, указанные в CALL, только после того, как процесс выполнения программы продолжается со следующей инструкцией после CALL.

Вызов FB со сложными типами данных

Пример: Передача ARRAY в функциональный блок

Передача параметров сложных типов может быть только символической

Network 1:

```

CALL FB 17, DB 30
Meas_1      := "Temperature".Cylinder
Sum_1       := MD20
Sum_2       := MD30
Meas_2      := "Temperature".Shaft
  
```

Address	Decl.	Name	Type	Initial Value	Comme
0.0	in	Meas_1	ARRAY[1..10]		
+4.0	in		REAL		
40.0	out	Sum_1	REAL		
44.0	out	Sum_2	REAL		
48.0	in_out	Meas_2	ARRAY		
+4.0	in_out		REAL		
54.0	stat	DB_Num	INT		

Address	Name	Type	Initial Value
0.0		STRUCT	
+0.0	Cylinder	ARRAY[1..10]	
+4.0		REAL	
+40.0	Shaft	ARRAY[1..15]	
+4.0		REAL	
=100.0		END STRUCT	

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.11Information and Training Center
Knowledge for Automation

Сложные типы данных

Также, как и у функций, адреса сложных типов данных (ARRAY, STRUCT, DATE_AND_TIME и STRING) можно полностью передавать вызываемому функциональному блоку.

Для передачи параметр того же самого типа данных, что и фактический параметр, который нужно передать, должен быть объявлен в вызываемом функциональном блоке.

Входные и выходные параметры

Назначение такого параметра возможно только символически.

Для входных и выходных параметров со сложным типом данных, области передачи для значений фактических параметров находятся в экземпляре DB. При вызове FB, фактические параметры для входных параметров копируются в экземпляр DB с использованием SFC 20 (BLKMOV) ("Передача по значению"), перед фактическим переходом к секции инструкций FB.

Тем же самым способом, значения выходных параметров копируются назад в экземпляр DB после того, как FB был обработан.

В результате на передачу параметров типа in и in_out затрачивается определенное время (время обработки). Это копирования не нужны параметрам типа in_out.

In_out-параметры

Никакой "передачи по значению" не происходит при передаче in_out-параметров сложного типа данных. Просто в экземпляре блока данных сохраняются 6 байтов для каждого in_out-параметра. В этих байтах сохраняется POINTER на фактические параметры ("Передача по ссылке").

Обратите внимание

- Входные и выходные параметры сложного типа данных могут быть инициализированы в разделе объявлений FB, однако in_out-параметры - нет.
- Входные и выходные параметры сложного типа данных могут не назначаться при вызове FB, in_out-параметры должны быть назначены.
- Доступ через память или с помощью косвенной регистровой адресации к параметрам типа in и out или in_out-параметрам сложного типа данных отличен от такового для элементарных параметров.

Характеристики вызова функционального блока

Передача параметров "по значению" (копирование значений):

- **Назначение параметров FB в CALL:**
 - Параметры FB могут быть не назначены
 - Назначение и отмена могут происходить "из вне"
Например: прямо с панели оператора
 - Исключение: in_out-параметры сложных типов данных (STRUCT, ARRAY, STRING and DATE_AND_TIME)
- **Инициализация:**
 - Параметры FB могут быть инициализированы при объявлении
 - Исключение: in_out-параметры сложных типов данных (STRUCT, ARRAY, STRING and DATE_AND_TIME)
- **Доступ к формальным параметрам имеет место с неявным использованием регистров DI и AR2**
 - Если регистры DI или AR2 изменены, доступ к данным экземпляра больше не возможен
- **Дополнительные инструкции вызова UC и CC**
 - Примеры: UC FB20 или CC FB20
 - Используются, если FB не имеет никаких данных экземпляра (параметров + статических переменных)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.12Information and Training Center
Knowledge for Automation

Назначение параметров блока

Параметры блока могут быть не назначены в вызове FB. В этом случае значения не копируются в или из экземпляра - DB. Параметры в экземпляре - DB содержат значения, которые были туда записаны в предыдущем вызове.

Исключение: in_out-параметры сложного типа данных должны быть назначены в списке параметров.

Доступ к параметрам из "вне"

Доступ к параметрам в экземпляре DB может быть осуществлен тем же самым способом как к данным глобального DB. Параметры блока могут таким образом также быть назначены или изменены из "вне". Это особенно полезно, когда, например, только индивидуальные компоненты сложных типов данных должны быть назначены или изменены, или параметры непосредственно связанные с областями ввода / вывода на OP.

Исключение: in_out-параметры сложного типа данных не могут быть назначены или изменены из "вне".

Инициализация

Параметры блока и статические переменные могут быть инициализированы в разделе объявлений FB. Если экземпляр DB создан, то начальные значения, указанные в разделе описаний вводятся в экземпляр DB.

Исключение: in_out-параметры сложного типа данных не могут быть инициализированы.

Обратите внимание

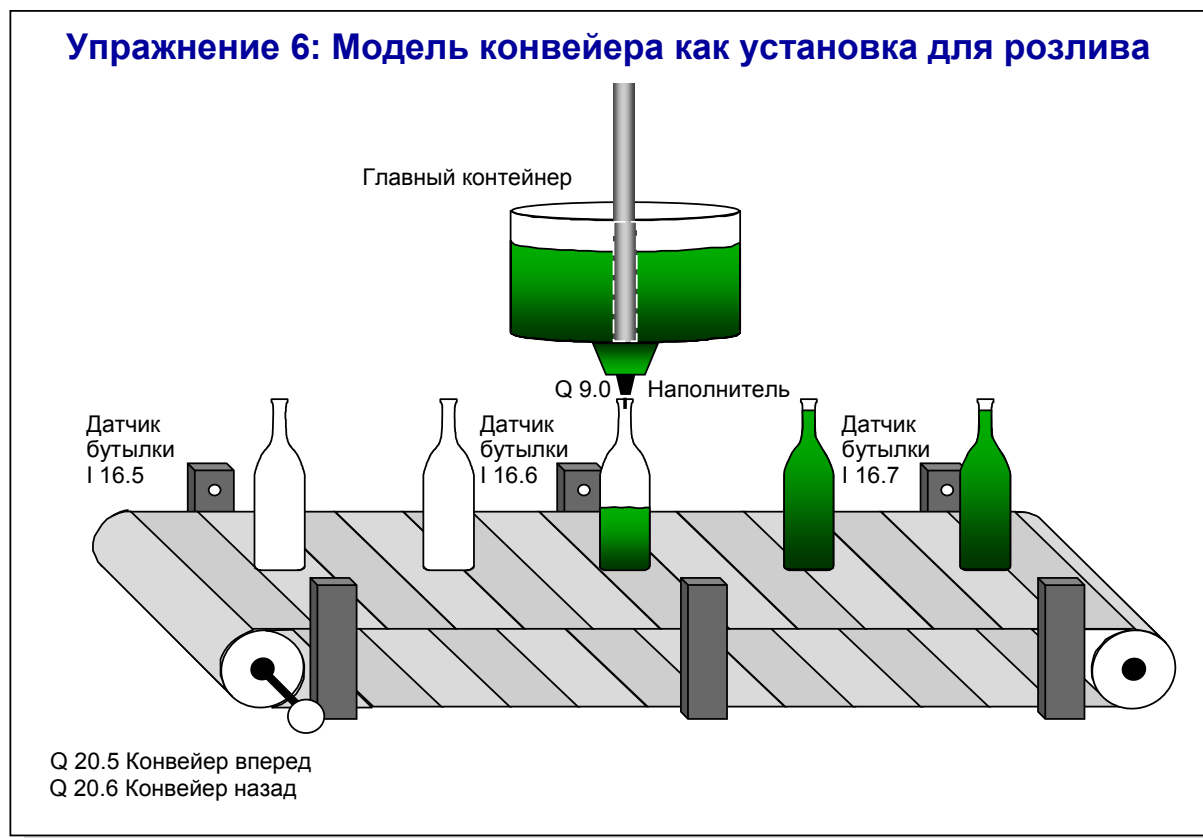
Если регистры DI или AR2 изменены во время обработки FB, то доступ к данным экземпляра (входам, выходам, in_out-параметрам и статическим переменным) в пределах FB больше не возможен.

Инструкции UC, CC

Блоки могут также вызываться независимой от RLO инструкцией UC или зависимой от RLO инструкцией CC.

UC и CC может использоваться только в случае, когда вызываемый FB не имеет никаких данных экземпляра, то есть ни параметров блока, ни статических переменных не было объявлено в разделе объявлений.

Упражнение 6: Модель конвейера как установка для розлива



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.13Information and Training Center
Knowledge for Automation

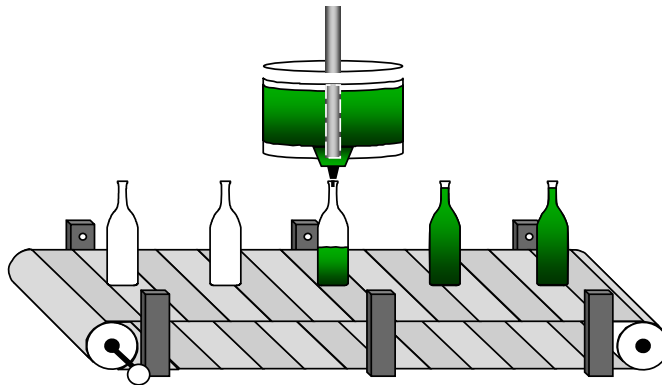
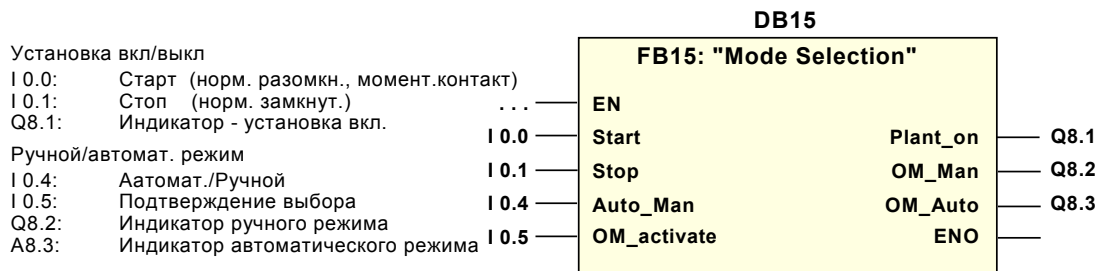
Задача

В следующей задаче должна быть автоматизирована установка для розлива бутылок:

Установка может работать в двух режимах "Ручной" и "Автоматический". Кроме того, существует соответствующая логика для включения и выключения системы.

- **Включение и выключение:** установка может быть включена, используя вход I 0.0 (Выключатель - мгновенный нормально разомкнутый контакт). Установка выключается входом I 0.1 (нормально замкнутый контакт). Когда установка включена, светодиод на выходе Q 8.1 зажег.
- **Выбор режима работы:** Когда установка включена, может быть выбран режим работы.
Ручной режим - I 0.4 = 0, автоматический - I 0.4 = 1. Выбранный способ должен быть подтвержден импульсом на входе I 0.5. Индикаторы для выбранного способа: ручной - Q 8.2, автоматический - Q 8.3.
Когда режим изменяется или установка выключается, выбранный режим отменяется.
- **Управление конвейером в ручном режиме:** В ручном режиме у конвейера может быть включено движение вперед (мгновенный контакт I 0.2 (Q 20.5)) и назад (I 0.3 (Q 20.6)).
- **Управление конвейером в автоматическом режиме:** В автоматическом режиме двигатель (Q 20.5) включается и остается включенным до того момента, как он будет выключен (I 0.1) или датчик в позиции заполнения (I 16.6) обнаруживает бутылку.
- **Заполнение бутылки:** Когда бутылка обнаруживается под заполнителем (I 16.6 = 1), начинается заполнение. Процедура заполнения моделируется интервалом в 3 секунды и индицируется на выходе Q 9.0.
- **Подсчет бутылок:** Другой датчик используется, чтобы считать заполненные бутылки. Датчик бутылок I 16.7 реагирует на полные бутылки. Полные бутылки подсчитываются со времени, включения установки и показан на цифровом индикаторе QW 12.

Упражнение 6.1а: Установка розлива - способ секций



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.14Information and Training Center
Knowledge for Automation

Обратите внимание При управлении объектом автоматизации (двигатель, лента конвейера и т.д.) используйте функциональные блоки .

Чтобы не нарушать принципы структурного программирования, Вы должны исключить в пределах функционального блока непосредственный доступ к глобальным адресам типа входов, выходов, меркеров и т.д. Каждая информационная передача сигналов процесса другим блокам программы пользователя должна происходить, используя параметры блока.

Только в вызове блока на самом высоком уровне, то есть в организационном блоке, Вы может назначать глобальные адреса непосредственно параметрам блока.

Способ управления: Вы найдете таблицу символов для упражнения 6.1 в программной папке Chap_06_1 проекта "Pro2_e_x51".

FB15

Прежде всего создайте FB15 " Mode_Selection", в котором запрограммирована полная логика для вкл\выкл и выбор режима работы установки .

FB15 "Mode_Selection" имеет следующие входные и выходные параметры:

#Start (in, BOOL): установка включается, используя #Start (1-активно).

#Stop (in, BOOL): установка выключается, используя #Stop (0-активно).

Входной параметр #Stop имеет приоритет по отношению к #Start.

#Plant_on (out, BOOL): Когда установка включена # Plant_on, установлен в "1".

Вы можете выбирать способ, когда установка включена.

#Auto_Man (in, BOOL): #Auto_Man = 0 - ручной режим работы

#Auto_Man=1- автоматический режим работы

#OM_activate (in, BOOL): Выбранный способ принят, когда поступает импульс в #OM_activate.

Активный способ обозначен FB15 в следующих выходных параметрах:

#OM_Man (out, BOOL): Ручной способ активен

#OM_Auto (out, BOOL): Автоматический способ активен

Упражнение 6.1b: Установка розлива - конвейер

Ручной режим

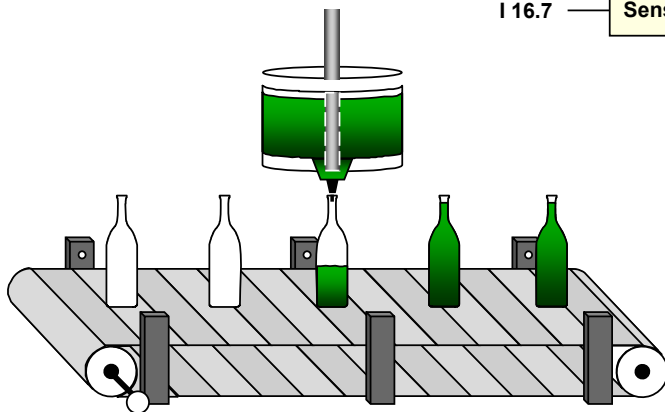
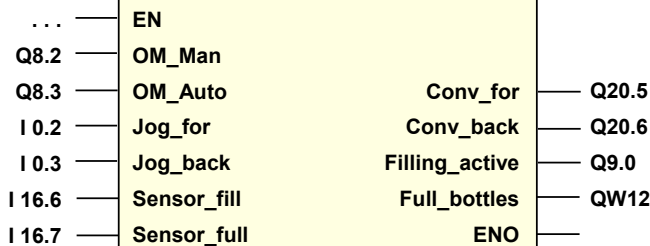
I 0.5: Выключатель вперед
I 0.6: Выключатель назад
Q20.5: Конвейер вперед
Q20.6: Конвейер назад

Автоматический режим

I 16.6: Датчик: место наполнения
I 16.7: Датчик: счетчик бутылок
Q9.0: Наполнение активно
QW12: Показ числа наполненных бутылок

DB16

FB16: "Conveyor_Control"



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.15



Information and Training Center
Knowledge for Automation

Управление конвейером: FB16

Создайте FB16 "Conveyor_Control", содержащий полную программу для управления конвейером в ручном и автоматическом режимах. FB16 имеет следующие входные и выходные параметры:

#OM_Man (in, BOOL): Конвейер используется в ручном режиме.

#OM_Auto (in, BOOL): Конвейер используется в автоматическом режиме.

#Jog_for (in, BOOL): Используя этот вход, конвейер может перемещаться вперед в ручном режиме. Этот вход не играет никакой роли в автоматическом режиме.

#Jog_back (in, BOOL): Используя этот вход, конвейер может перемещаться назад в ручном режиме. Этот вход не играет никакой роли в автоматическом режиме.

#Sensor_fill (in, BOOL): Указывает, что пустая бутылка достигла заполнителя.

#Sensor_full (in, BOOL): Указывает, что полная бутылка прошла барьер для подсчета полных бутылок.

#Conv_for (out, BOOL): Выдает сигнал управления для движения конвейера вперед.

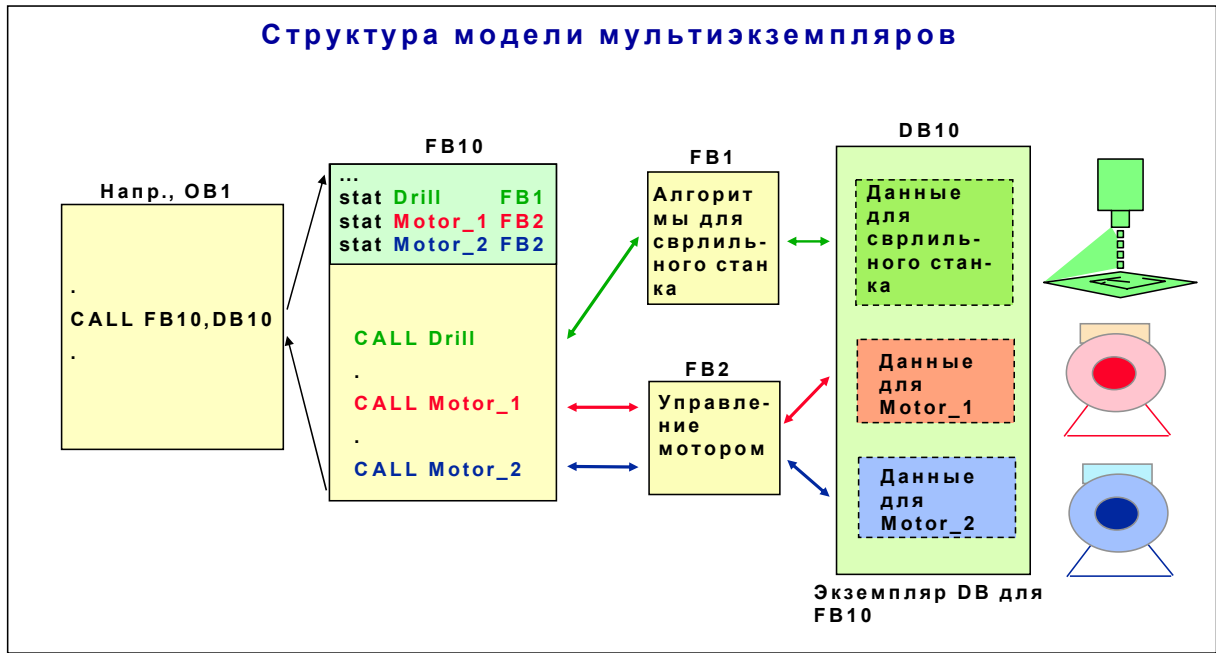
#Conv_back (out, BOOL): Выдает сигнал управления для движения конвейера назад.

#Filling_active (out, BOOL): Указывает, что заполнение в настоящее время активно.

#Full_bottles (out, WORD): Дает число заполненных бутылок в BCD коде.

Вызовите оба эти блока вместе со связанными экземплярами блоков данных DB15 и DB16 в OB1 и назначьте параметры обоим FB, как это указано выше на двух слайдах.

Структура модели мультиэкземпляров



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.16



Information and Training Center
Knowledge for Automation

Модель мультиэкземпляров

В дополнение к экземплярам функциональных блоков, когда экземпляр DB определяется в запросе FB, STEP 7 также поддерживает явную декларацию экземпляров FB в пределах функционального блока более высокого уровня.

Для этого экземпляры вызываемых функциональных блоков объявляются в разделе описаний вызывающего функционального блока FB 10, как статические переменные с типами данных FB1 или FB2, используя символические идентификаторы (Drill, Motor_1 and Motor_2). В пределах функционального блока более высокого уровня отдельные экземпляры тогда вызываются, используя их символические идентификаторы. Функциональный блок FB10 более высокого уровня должен, однако, вызываться с собственным экземпляром DB (DB10).

STEP 7 проверяет, что в созданном экземпляре DB более высокого уровня созданы области данных, требуемые для отдельных экземпляров. В вызове отдельных экземпляров используются символические имена, макрокоманда CALL устанавливает в регистр AR2 указатель на начало области данных, назначенной экземпляру, чтобы во время работы блока можно было обращаться к параметрам и локальным переменным экземпляра.

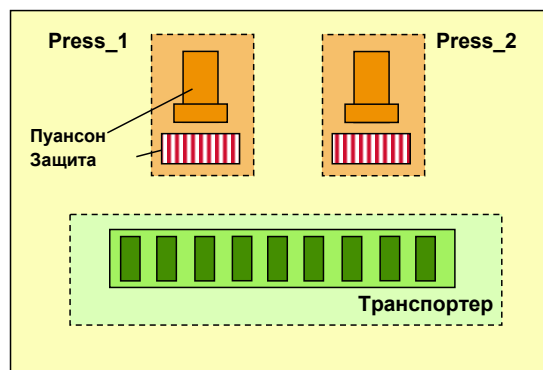
Преимущества

Использование модели мультиэкземпляров имеют следующие преимущества:

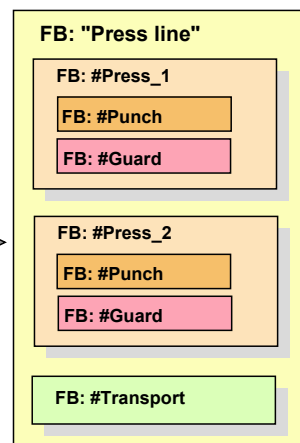
- отдельные экземпляры не требуют каждый раз собственного блока данных. В пределах иерархии вызовов функциональных блоков только один экземпляр DB "потрачен впустую" при вызове функционального блока, вызывающего все остальные.
- модель мультиэкземпляров "сваривает" функциональный блок и область данных экземпляра в один объект (экземпляр FB), который может быть также обработан, как одна единица. Программист не должен заботиться о управлении (создание, адресация) отдельных областей данных экземпляра. Он должен просто обеспечить экземпляр DB для "главного" FB.
- модель мультиэкземпляров поддерживает объектно-ориентированный стиль программирования.

Объектно-ориентированное программирование с использованием мультиэкземпляров

Пример: Линия прессов



Технологическое подразделение



Техническое подразделение - программа с экземплярами FB

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.17



Information and Training Center
Knowledge for Automation

Части процесса

Части процесса - физические объекты в процессе, типа сборочной линии. Это либо полная машина, либо часть машины (например, пресс или пуансон).

Части процесса могут, в свою очередь, сами состоять из частей (например, объект "Пресс" содержит части "Пуансон" и "Защита"). Части процесса могут быть, таким образом, разбиты на более мелкие "подчасти".

Объектно-ориентированный стиль программирования.

Вы можете поддерживать объектно-ориентированный стиль программирования с помощью функциональных блоков. Техническое описание частей процесса выполняется в виде экземпляров FB. Разделение программы пользователя на части достигается объявлением экземпляров FB низшего уровня в FB более высокого уровня. Таким образом, то же самое разделение процесса на части достигнуто в программе пользователя как в существующей системе (Концепция объектно-ориентированного программирования).

Возможность многократного использования программного обеспечения

Эта иерархическая концепция предлагает широкие возможности многократного использования однажды созданного программного обеспечения и, таким образом, имеет большой потенциал экономии ресурсов при создании, модификации и сопровождении программ пользователя:

- Всякий раз, когда изготовитель создает "подчасть" процесса (клапан, двигатель, и т.д.), он также предоставляет FB для управления этой "подчастью"
- Всякий раз, когда такая физическая часть процесса включена, в следующую большую часть, экземпляр FB "подчасти" также объявлен в FB части более высокого уровня.

FB - основные компоненты программы управления. На стадии планирования программы, FB имеют ту же самую задачу, как интегральные схемы (ICS) в производстве печатных плат. Структура программ пользователя состоит из заранее созданных FB, которые должны быть просто связаны.

Осуществление "Линии прессов" в STEP 7



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.18Information and Training Center
Knowledge for Automation

Модель мультиэкземпляров

Когда используется модель мультиэкземпляров, экземпляр DB содержит данные для нескольких функциональных блоков иерархии вызовов. Для этого, используя символические идентификаторы, в разделе объявлений вызывающего FB, объявлены экземпляры вызываемых функциональных блоков.

"Главный" экземпляр FB (здесь: FB10 "Press_line") должен вызываться абсолютно или символически вместе с собственным экземпляром DB (здесь: DB10).

Объявления

В секции stat раздела описаний FB10 ("Press_line") объявлены два экземпляра (переменные) с типом данных FB1 ("Press") с именами #Press_1 и #Press_2, а также экземпляр с типом данных FB2 ("Transport") с именем #Transport.

В разделе описаний FB1 объявлены экземпляр FB4 ("Punch") с именем #Punch и экземпляр FB5 ("Guard") с именем #Guard.

В разделе инструкций FB1 ("Press"), вызываются соответствующие экземпляры FB, используя символические имена #Punch и #Guard, которые были объявлены в разделе описаний.

Обратите внимание

Объявление экземпляра в разделе описаний функционального блока возможно только, если FB, которого экземпляр объявляется, уже существует.

При проектировании подобной иерархии вызовов, те FB, которые должны вызываться последними, должны быть созданы первыми.

Мультиэкземпляр DB

Мультиэкземпляр DB имеет ту же самую структуру, что и разделы объявлений соответствующих функциональных блоков. Если экземпляр вызван в разделе инструкций, то ему автоматически становятся доступны данные в соответствующей секции экземпляра DB (DB10).

Свойства модели мультиэкземпляров

Преимущества модели мультиэкземпляров:

- Только один DB требуется для нескольких экземпляров
- Не нужно никакое дополнительное управление во введенных "частных" областях данных для соответствующих экземпляров
- Модель мультиэкземпляров делает возможным "объектно-ориентированный стиль программирования" (возможность многократного использования посредством "включения")
- Максимальная глубина вложения 8

Предпосылки для FB:

- Прямой (I, Q) доступ к сигналам процесса в пределах FB не возможен
- Доступ для обработки сигналов или связи с другими частями процесса может быть только через параметры FB
- FB может только сохранять состояние процесса в статических переменных, но не в глобальных DB или меркерах

Обратить внимание:

- К данным экземпляра можно обращаться также с "внешней" стороны. Напр. в OB1: L "Press line".Press_2.Punch.<VarName>

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.19Information and Training Center
Knowledge for Automation

Выгоды модели мультиэкземпляров

С помощью модели мультиэкземпляров Вы можете

хранить соответствующие секции данных нескольких экземпляров одной и той же иерархии вызовов в одном DB.

Только один DB требуется для нескольких экземпляров.

В модели мультиэкземпляров не нужны никакие меры для администрирования локальных данных FB.

Модель мультиэкземпляров поддерживает концепцию объектно-ориентированного программирования. Код и данные, которые необходимы для управления частью процесса, суммированы в FB.

Если часть процесса состоит из иерархических "подчастей", тогда точно эта структура может быть отражена в программе пользователя посредством модели мультиэкземпляров. Программа управления может быть разработана с теми же самыми экземплярами FB, что и машина, которую они описывают.

В модели мультиэкземпляров STEP7 поддерживает глубины вложения 8.

Предпосылки для мультиэкземпляров

Чтобы использовать FB как мультиэкземпляр необходимо твердо придерживаться следующих пунктов:

- При управлении производственным процессом недопустим никакой прямой доступ к глобальным адресам CPU (типа входов и выходов). Каждый доступ к глобальному входу и выходу нарушают возможность многократного использования.
- Связь с процессом или с другими секциями программы (FB) должна быть выполнен, используя только параметры FB. Только после того, как FB интегрирован в единицу более высокого уровня, можно сделать "назначения" через список параметров FB при вызове FB.
- Состояния или другая информация относительно объекта, которым нужно управлять должны "запоминаться" FB в собственных статических переменных.

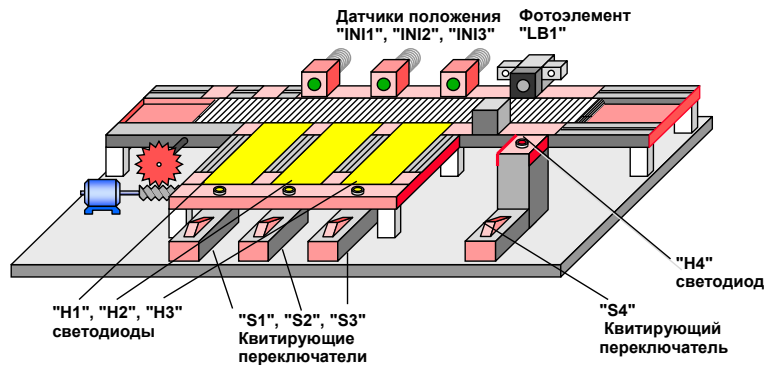
Упражнение 6.2: Модель конвейера как сборочная линия

Последовательность процессов для рабочего места

- Обработка детали
- Обработка закончена
- Деталь на ленте транспортера
- Ждать следующую необработанную деталь
- Взять необработанную деталь с ленты

Последовательность процессов для транспортной ленты

- Ждать окончания обработки
- Транспортировка на заключительную сборку
- Окончание сборки, положить новую деталь
- Транспортировка на сборку



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.20Information and Training Center
Knowledge for Automation

Цель

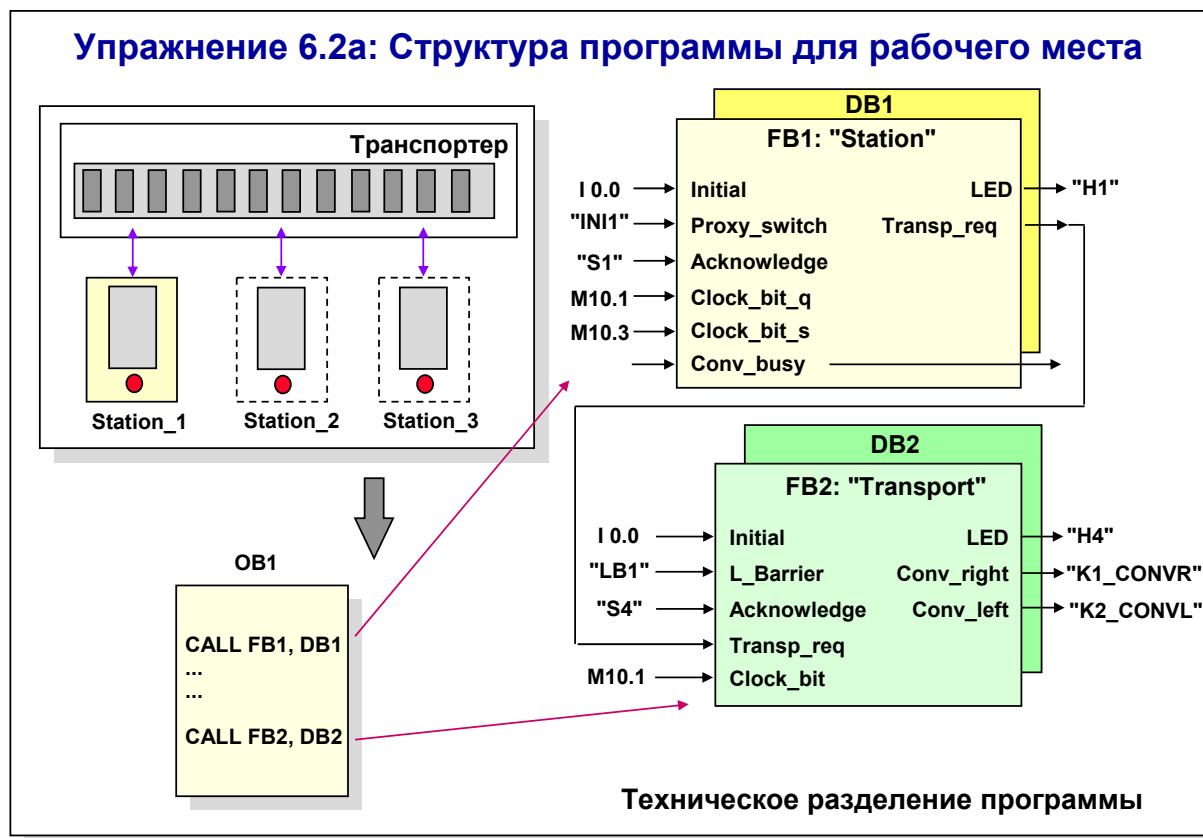
Сборочная линия должна быть автоматизирована. Решение задачи должно быть осуществлено, используя программирование FB. Отдельные FB используются в каждом случае: для управления рабочим местом 1 и лентой транспортера. FB для рабочего места должен быть пригодным для использования в модели мультиэкземпляров. В следующем упражнении функциональные возможности сборочной линии должны быть расширены посредством модели мультиэкземпляров, чтобы работали места 2 и 3.

Принцип функционирования модели конвейера

Модель конвейера должна работать как сборочная линия со следующими функциональными возможностями (пока только для одного рабочего места):

1. Система находится в начальном состоянии, то есть рабочее место 1 имеет одну деталь, которая является в настоящее время обрабатываемой. Это обозначено зажженным светодиодом "H1" на рабочем месте 1. Лента транспортера свободна, то есть "LB1" не занят. Двигатель конвейера выключен.
2. После того, как операция с деталью закончена, оператор подтверждает это переключателем "S1". Светодиод "H1" мигает.
3. Оператор размещает обработанную деталь на "пустой" ленте перед датчиком "INI1". Светодиод "H1" выключается.
4. Лента транспортирует обработанную деталь на заключительную сборку. Светодиод "H4" мигает во время транспортировки. Когда место заключительной сборки достигнуто, "H4" светодиод светится непрерывно.
5. Оператор на месте заключительной сборки берет обработанную деталь с ленты и размещает новую необработанную деталь на ленте. Он подтверждает это выключателем "S4".
6. Лента транспортирует новую необработанную деталь назад для обработки на рабочем месте 1. Светодиод "H4" мигает во время транспортировки. Когда достигнут датчик "INI1", начинает мигать светодиод "H1" на рабочем месте 1.
7. Оператор может брать необработанную деталь с ленты и размещать ее на рабочем месте 1 и начинать обрабатывать. Светодиод "H4" зажжен непрерывно. Процесс работы начинается снова с шага 1.

Упражнение 6.2а: Структура программы для рабочего места



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.21Information and Training Center
Knowledge for Automation

Задача

На первом шаге в сборочной линии существует только одно рабочее место. Управление всей системой разделено на два процесса:

- Рабочее место: Управление первым рабочим местом осуществляет FB1 с символическим именем "Station" (глобальная таблица символов)
- Транспортировка: Управление линией транспортера осуществляет FB2 с символическим именем "Transport"

Для управления всей сборочной линией оба FB вызываются в OB1, каждый с собственным экземпляром DB.

Что делать

Для упражнения 6.2 Вы найдете функциональные блоки FB1 и FB2, а также соответствующую таблицу символов в папке программ Chap_06_2 проекта "Pro2_e_x51".

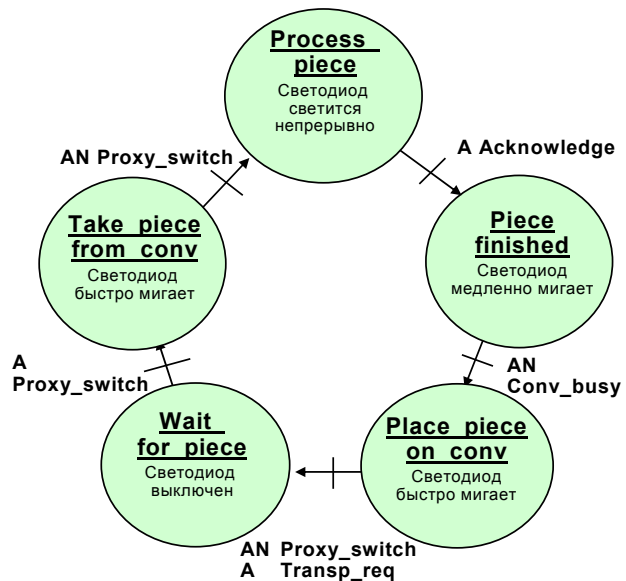
1. Откройте проект "Pro2_e_x51" и вставьте конфигурацию аппаратных средств Вашей станции.
2. Для CPU назначьте тактовым байтом MB10 и затем загрузите данные конфигурации в CPU.
3. Вызовите FB1 с экземпляром DB1 в OB1, и назначьте сигналы процесса параметрам интерфейса.
Назначте параметру входа #Clock_bit_q - M10.1, а #Clock_bit_s - M10. 3.
4. Программа вызывает FB2 (экземпляр DB2) после FB1 в OB1.
Назначте входные и выходные параметры согласно описанию FB1 и FB2.
5. Загрузите блоки в CPU и проверьте функционирование Вашей программы.

Метод функционирования FB1 "Station"

Объявления в FB1:

FB1: "Station"	
IN-параметры:	Data type:
→ Initial	BOOL
→ Proxy_switch	BOOL
→ Acknowledge	BOOL
→ Clock_bit_q	BOOL
→ Clock_bit_s	BOOL
OUT-параметры:	
← LED	BOOL
← Transp_req	BOOL
I/O-параметры:	
↔ Conv_busy	BOOL
Stat. Var.:	
State	STRUCT
Process_piece	BOOL
Piece_finished	BOOL
Place_part_on_conv	BOOL
Wait_for_piece	BOOL
Take_piece_from_conv	BOOL
END_STRUCT	

Модель состояний:



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.22Information and Training Center
Knowledge for Automation

Initialization

Импульсом во входном параметре **#Initial**, FB1 может быть initialized состоянием **#Process_piece**. Назначте **#Initial** I 0.0.

#Process_piece

В этом состоянии деталь обрабатывается. Светодиод "H1" непрерывно горит, указывая на обработку.

Переход к состоянию **#Piece_finished** происходит, когда оператор подтверждает завершение обработки выключателем "S1".

#Piece_finished

В этом состоянии, светодиод мигает с частотой от **#Clock_bit_s** (медленно). Оператор ждет разрешения от ленты транспортера. (Важно для расширения до 3 станций!).

Если конвейер пуст (**#Conv_busy=0**), это немедленно определяется как занято (**#Conv_busy=1**) и идет в состояние **#Place_piece_on_Conv**.

#Place_piece_on_conv

В этом состоянии светодиод мигает с частотой от **#Clock_bit_s** (быстро) - оператор может размещать деталь на конвейере. С сигналом **#Proxy_switch=1** сигнал **#Transp_req=1** также устанавливается, таким образом вызывая движение конвейера к месту заключительной сборки. Переход к состоянию **#Wait_for_piece** происходит, когда деталь останавливается у датчика (**#Proxy_switch=0**).

#Wait_for_piece

В этом состоянии оператор ждет прибытия новой необработанной детали; светодиод перед станцией выключен.

С прибытием (новой необработанной детали (**Proxy_switch=1**)) имеет место переход к состоянию **#Take_part_from_conv**.

#Take_piece_from_conv

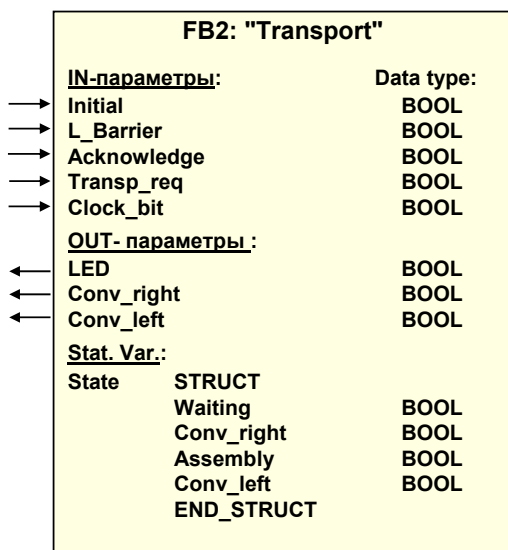
В этом состоянии светодиод мигает от **#Clock_bit_q** (быстро)

Деталь может быть взята с конвейера.

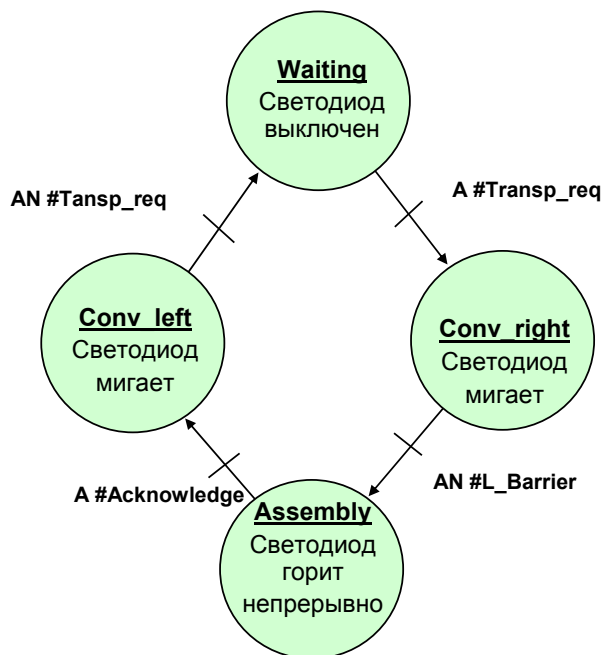
Переход к состоянию **#Process_piece** имеет место, когда принята принята (**#Proxy_switch=0**).

Метод функционирования FB2 "Transport"

Интерфейс FB2:



Модель состояний:



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.23Information and Training Center
Knowledge for Automation

Initialization

FB2 может быть инициализирован состоянием *#Waiting* через входной сигнал *#Initial*. Назначьте входному параметру *#Initial* I0.0.

#Waiting

В этом состоянии лента транспортера ждет окончания обработки детали, которая помещена на конвейер в одно из рабочих мест. Пока лента транспортера находится в состоянии *#Waiting* мотор остановлен и светодиод "H4" выключен.

Из состояния 1 при сигнале *#Transport_req* имеет место переход в состояние *#Conv_right*.

#Conv_right

В этом состоянии деталь транспортируется в направлении места заключительной сборки. Пока конвейер перемещается, светодиод "H4" мигает с частотой (M10. 1), определяемой входным параметром.

Место заключительной сборки достигнуто, то есть состояние *#Assembly* имеет место, когда обработанная деталь проходит фотоэлемент "LB1".

#Assembly

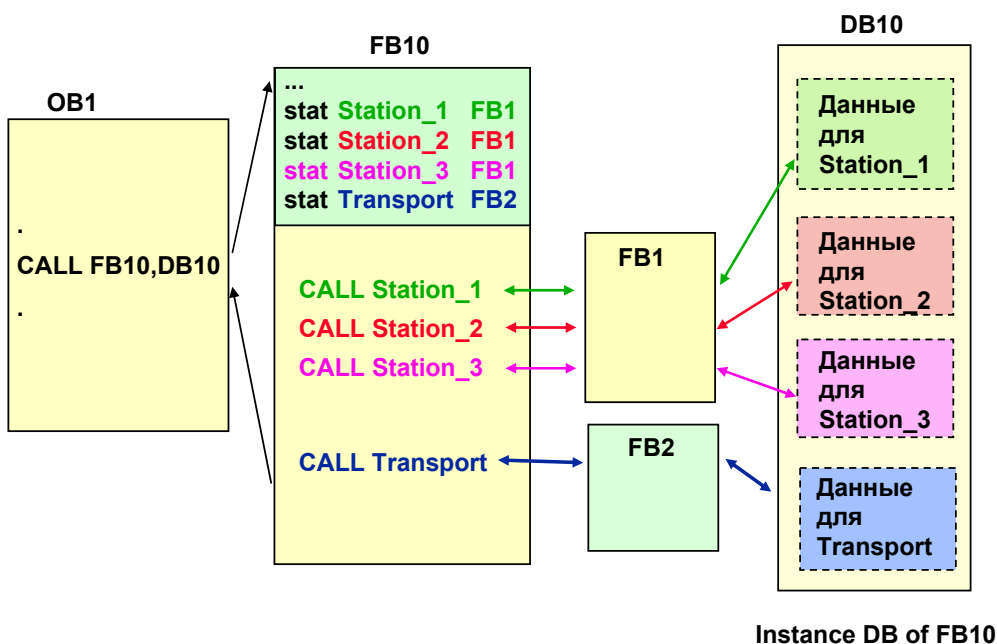
В этом состоянии оператор обменивает обработанную деталь на необработанную. В этом состоянии светодиод "H4" зажат непрерывно. Сигнал оператора для завершения этой задачи - выключатель "S4". Этот сигнал также ведет к переходу в состояние *#Conv_left*.

#Conv_left

В этом состоянии необработанная деталь транспортируется в направлении рабочего места. Пока конвейер перемещается, светодиод "H4" мигает с частотой, определяемой входным параметром *#Clock_bit*.

Транспортировка останавливается, когда входной сигнал *#Transp_req* установлен повторно. Происходит переход к состоянию *#Waiting*.

Упражнение 6.2b: Расширение до 3 рабочих мест

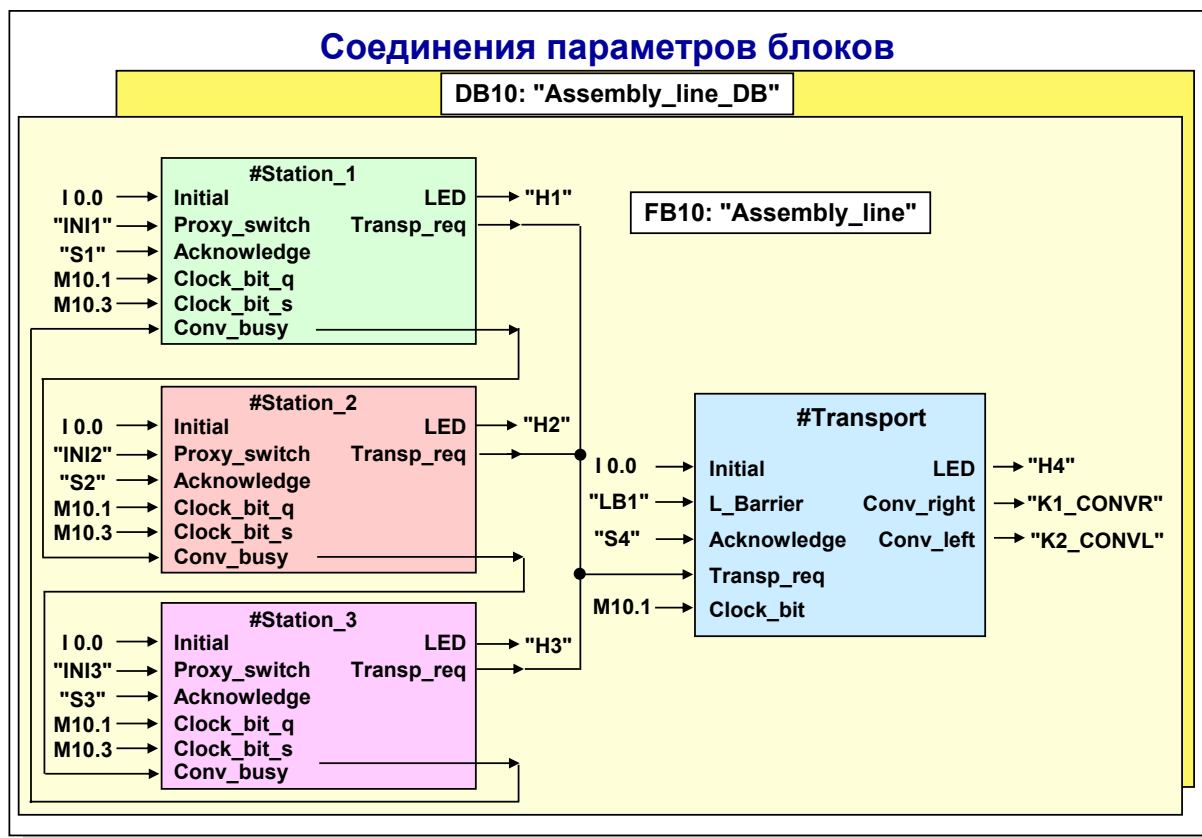


SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_06E.24Information and Training Center
Knowledge for Automation

Структура программы Во второй части этого упражнения должны быть достигнуты полные функциональные возможности модели конвейера для всех трех рабочих мест. Для этого управление полной моделью конвейера (3 рабочих места и один транспортер) должно быть перемещено в один FB (FB10). В пределах FB10 управление относительноными тремя рабочими местами осуществляется, как отдельные экземпляры FB1, а управление транспортером осуществляется как экземпляр FB2.



SIMATIC S7

Siemens AG 1999. All rights reserved.

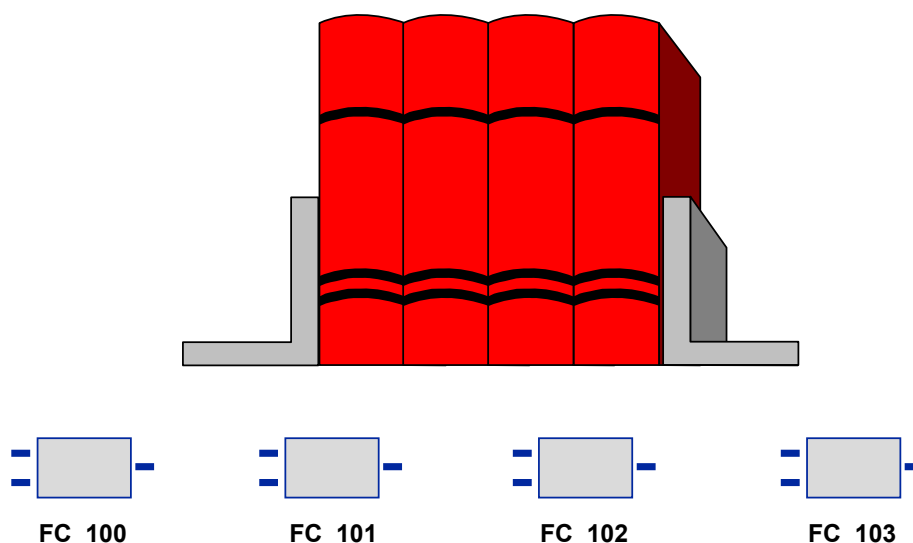
Date: 04.11.2005
File: PRO2_06E.25Information and Training Center
Knowledge for Automation**Что делать**

1. Сначала создайте FB10. В разделе объявлений в секции stat. Var. объявите три экземпляра FB1 с именами: #Station_1, #Station_2 и #Station_3 и один экземпляр FB2 с именем #Transport.
2. В FB10, прежде всего вызовите последовательно #Station_1, #Station_2, #Station_3 и #Transport и свяжите параметры блока согласно схеме, приведенной выше.
Обратите внимание на соединение in_out-парамет #Conv_busy. Как это может быть осуществлено? Могут временные или статические вспомогательные переменные использоваться здесь? Также обратите внимание на присоединение выходного параметра #Transp_req (логическое ИЛИ) к входному параметру #Transp_req управление транспортером. Как такое соединение может быть осуществлено?
3. Создайте явно DB10 со связанным FB10. Редактируйте DB10, и проверьте структуру объявлений в представлении data view DB-редактора.
4. Вызовите FB10 с экземпляром DB10 в OB1.
5. Загрузите участвующие блоки в CPU и проверьте результат.

Вопросы

- Что является преимуществами и неудобствами такого подхода?
- Как нужно изменить управление так, чтобы "пустая" сборочная линия могла также быть "заполнена" или "заполненная" линия могла бы также быть "освобождена".

Использование библиотек



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.1Information and Training Center
Knowledge for Automation

Содержание

Стр

Интересные факты о библиотеках	2
Конфигурация и содержание стандартной библиотеки	3
Интересные факты о системных библиотеках	4
Обзор системных функций (часть 1)	5
Обзор системных функций (часть 2)	6
Обзор системных функций (часть 3)	7
Обзор системных функций (часть 4)	8
Обзор системных функций (часть 5)	9
Вызов системных функций и системных функциональных блоков	10
Оценка сообщений об ошибках	11
Упражнение 7.1: Генерирование DB с атрибутом "UNLINKED"	12
Упражнение 7.2: Тестирование блока данных (SFC 24: только для S7-400)	13
Упражнение 7.3: Создание DB (SFC 22)	14
Упражнение 7.4: Копирование DB из загрузочной памяти в рабочую память (SFC 20)	15
Дополнительное упражнение 7.5: Инициализация DB (SFC 21 FILL)	16
Дополнительное упражнение 7.6: Запись сообщения в диагностический буфер (SFC 52)	17
Дополнительное упражнение 7.7: Счетчик	18
Библиотека: S5-S7 Converting Blocks	19
Библиотека: TI-S7 Converting Blocks (Часть 1)	20
Библиотека: TI-S7 Converting Blocks (Часть 2)	21
Библиотека: Communication Blocks	22
Библиотека: PID Control Blocks	23

Интересные факты о библиотеках

Цель:

- Хранение компонентов программы многократного использования
- Не возможна прямая передача в CPU и тестирование

Конфигурация библиотеки:

- Библиотека может содержать несколько программных папок
- Библиотека не может содержать "Hardware" (аппаратной части)
- Каждая программная папка содержит:
 - папки "Blocks", "Source Files", "Symbols"
 - папку "Charts" (только для программного пакета: S7-CFC)

Использование библиотек:

- **С SIMATIC Manager:**
 - библиотеки могут быть созданы (но не с теми же именами, что проекты)
 - блоки могут быть копироваться между библиотеками и проектами
 - библиотеки могут архивироваться

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.2



Information and Training Center
Knowledge for Automation

Краткий обзор

Библиотеки используются, чтобы хранить компоненты программы многократного использования для SIMATIC S7/M7. Компоненты программы могут быть скопированы из существующих проектов в библиотеку или они могут быть созданы непосредственно в библиотеке, независимой от проектов.

Те же самые функциональные возможности, что и у проектов, доступны для содержимого папки S7-Programs в библиотеке - за исключением тестирования.

Конфигурация

Точно так же как проекты, библиотеки конфигурируются иерархическим способом:

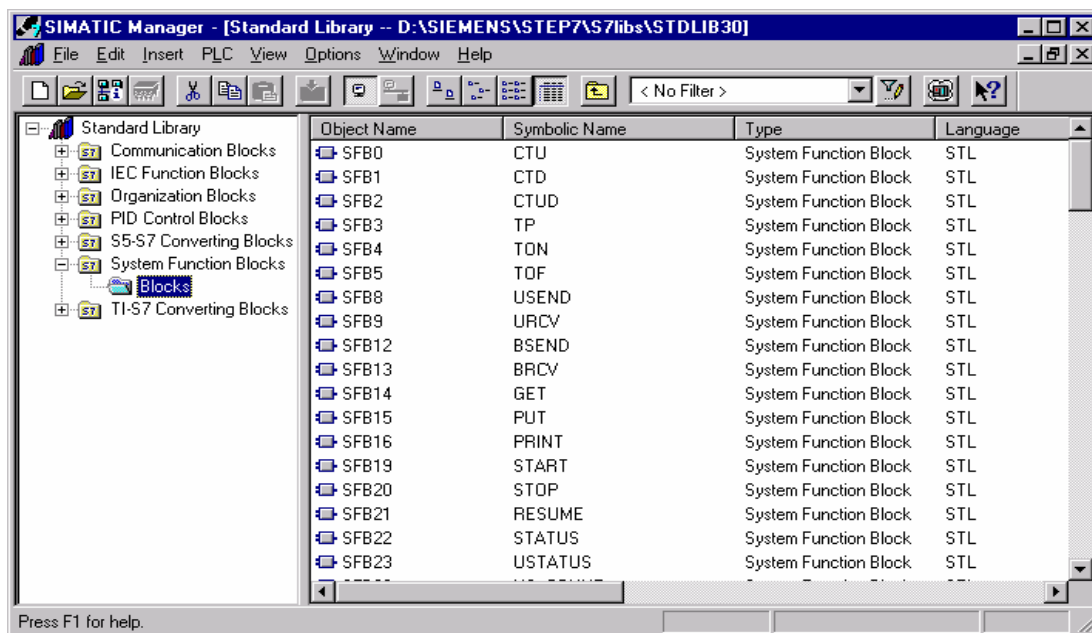
- библиотеки могут содержать S7-Programs.
- S7-Program может содержать одну папку Blocks, одну - Source Files, одну папку Charts, а также один объект Symbols (символьную таблицу).
- папка Blocks содержит блоки, которые могут быть загружены в S7-CPU. Таблицы переменных (VAT) и определенные пользователем типы данных, содержатся в ней же.
- папка Source Files содержит исходные файлы для программ, созданных на различных языках программирования.
- папка Charts содержит CFC-диаграммы (только для программного пакета S7-CFC)

Когда Вы вставляете новую папку S7-Program, папки Blocks и Source Files автоматически создаются в ней.

Использование библиотек

Блоки, которые должны использоваться многократно, могут быть сохранены в библиотеках. Оттуда, они могут быть скопированы в программу пользователя и вызываться другими блоками

Конфигурация и содержание стандартной библиотеки



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.3



Information and Training Center
Knowledge for Automation

Введение

Две стандартные библиотеки автоматически устанавливаются на жестком диске при установке программного обеспечения STEP 7 :

- стандартные библиотеки stdlibs (V2) для версии 2 и
- стандартная библиотека V3.x для версии 3.

От этих библиотек Вы можете копировать нужные блоки в Ваш проект.

Открытие библиотеки

Чтобы открыть библиотеку, используйте команду: *File -> Open* или соответствующую иконку в Toolbar.

Появится окно диалога, в котором Вы можете выбрать нужный проект или библиотеку.

Стандартная библиотека

Standard Library V3.x - стандартная библиотека - содержит следующие папки S7-Programs:

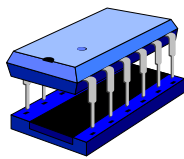
- Communication Blocks: содержит функции для подключения модулей распределенного ввода - вывода при использовании S7-300 Profibus CP
- IEC Converting Blocks: содержит блоки для функций IEC, например, для работы с переменными типов данных DATE_AND_TIME и STRING (см. гл. 5).
- Organization Blocks: содержит все организационные блоки S7-300/400
- PID Control Blocks: содержит блоки PID-управления
- S5-S7 Converting Blocks: содержит стандартные блоки, которые требуются при преобразовании S5-Programs к S7
- System Function Blocks: содержит все системные функции S7-300/400.
- TI-S7 Converting Blocks: содержит различные стандартные функции, например, масштабирование аналоговых величин и т.д.

Обратите внимание

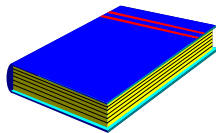
Дополнительные библиотеки устанавливаются во время установки дополнительных программных пакетов.

Описание S7 Библиотек *PID Control Blocks* и *TI - S7 Converting Blocks* расположены по следующим адресам: *Taskbar -> SIMATIC -> S7 manuals -> PID Control, Standard Functions 2*.

Интересные факты о системных библиотеках



Системные функции (SFC и SFB) хранятся в операционной системе CPU.



Справочник по программному обеспечению для S7-300/400 “Системные и стандартные функции”



Расширенная online- помощь в STEP 7

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.4



Information and Training Center
Knowledge for Automation

Введение

Системные функции представляют собой часто используемые стандартные функции и функции, которые не могут быть выполнены инструкциями STEP 7 (как например, генерирование DB). Системные функции интегрированы в операционную систему CPU. Системные функции вызываются в программе пользователя инструкциями CALL SFC или CALL SFB. Для блоков SFB также необходимы связанные блоки данных.

Возможность использования системной функции зависит от программируемого логического контроллера и используемого типа CPU. Используя программное обеспечение STEP 7, Вы можете, например, отобразить системные блоки в SIMATIC-Manager.

Системные блоки защищены, поэтому, когда Вы читаете их, то отображается только описательная часть с входными параметрами, выходными параметрами и параметрами ввода-вывода.

В системах S7-300 и S7-400 системные блоки имеют одинаковый номер, функцию и одинаково вызываются.

Руководство

Справочное руководство по системному программному обеспечению для S7-300/400, Системные и стандартные функции (перевод на русский язык имеется в представительстве фирмы Siemens в Москве) содержит расширенное описание системных функций.

Online- помощь

Имеется также расширенное описание системных функций в программном обеспечении STEP 7. Вызовите меню help в программном редакторе и выберите тему: *Help topics --> Block help --> Help with SFBs/SFCs*.

Обзор системных функций (часть 1)

Функц. группа	Функция	Блок	S7-300	S7-400
Копирование и блоковые функции	Перенос блока	SFC 20	X	X
	Заполнение массива	SFC 21	X	X
	Создание блока DB	SFC 22	X	X
	Удаление DB	SFC 23	-	X
	Тестирование DB	SFC 24	-	X
	Сжатие	SFC 25	-	X
	Замена значения в ACCU 1	SFC 44	X ¹⁾	X
Управление программой	Прерыв. для м-компьютинга	SFC 35	-	X ²⁾
	Повторный запуск врем. цикла	SFC 43	X	X
	Перевод CPU в STOP	SFC 46	X	X
	Задержка обработки	SFC 47	X ¹⁾	X
Управление часами	Установка времени	SFC 0	X	X
	Чтение времени	SFC 1	X	X
	Синхронизация	SFC 48	-	X
Управление счетчиком времени	Установка счетчика	SFC 2	X ¹⁾	X
	Запуск и останов	SFC 3	X ¹⁾	X
	Считывание счетчика	SFC 4	X ¹⁾	X
	Чтение системного времени	SFC 64	X	X

1) не для CPU 312IFM

2) только для новых CPU

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.5Information and Training Center
Knowledge for Automation

Функции копирования и блоковые функции

- **SFC 20** копирует содержимое области памяти.
- **SFC 21** заполняет область памяти содержимым другой области.
- **SFC 22** создает блок данных в рабочей памяти.
- **SFC 23** удаляет блок данных в рабочей памяти. DB не должен быть открыт.
- **SFC 24** определяет имеется ли DB в рабочей или загружаемой памяти, длину блока в байтах и защищен ли он от записи.
- **SFC 25** производит сжатие рабочей памяти. Если блоки корректировались, то занимаемые ими пространства в рабочей памяти ("дыры") удаляются.
- **SFC 44** (вызов из OB 122 или OB121) передает при ошибках в ACCU новое подходящее значение, с которым продолжается обработка программы.

Управление программой

- **SFC 35** запускает синхронно OB 60 во всех CPU (для мультикомпьютинга).
- **SFC 43** заново запускает контроль времени цикла CPU.
- **SFC 46** переводит CPU в состояние останова (STOP).
- **SFC 47** вводит задержку времени до 32767 µs.

Обработка часов

- **SFC 0** устанавливает новую дату и время в CPU.
- **SFC 1** считывает текущую дату и время из CPU.
- **SFC 48** синхронизирует часы ведомых устройств по часам ведущего CPU. Функция должна запускаться только для ведущего CPU (master).

Управление счетчиком времени

- CPU имеет определенное число счетчиков рабочего времени, в которые Вы можете записывать время работы оборудования.
- **SFC 2** устанавливает счетчик рабочего времени на заданное значение.
 - **SFC 3** запускает или останавливает счетчик рабочего времени.
 - **SFC 4** считывает текущее время из счетчика и его статус.
 - **SFC 64** считывает системное время CPU.

Системное время - это автономный счетчик времени, который увеличивается каждые 10 ms (для S7-300) или 1 ms (для S7-400). Максимальное значение счетчика - 2147483647 ms, после которого счет снова начинается с 0.

Обзор системных функций (часть 2)

Функцион. группа	Функция	Блок	S7-300	S7-400
Передача наборов данных	Запись динамических параметров	SFC 55	X	X
	Запись фиксированных параметров	SFC 56	X	X
	Параметризация модулей	SFC 57	X	X
	Запись набора данных	SFC 58	X	X
	Чтение набора данных	SFC 59	X	X
Прерывания по времени	Установка прерывания	SFC 28	X ¹⁾	X
	Отмена прерывания	SFC 29	X ¹⁾	X
	Активация прерывания	SFC 30	X ¹⁾	X
	Опрос	SFC 31	X ¹⁾	X
Прерывания с задержкой	Запуск	SFC 32	X ¹⁾	X
	Отмена	SFC 33	X ¹⁾	X
	Опрос	SFC 34	X ¹⁾	X
Синхронные ошибки	Маскирование ошибок	SFC 36	X	X
	Демаскирование	SFC 37	X	X
	Чтение регистра состояний	SFC 38	X	X
Ошибки прерываний и асинхронные	Блокировка новых прерываний	SFC 39	X	X
	Разрешение новых прерываний	SFC 40	X	X
	Задержка обработки новых прерываний	SFC 41	X	X
	Разрешение прерываний с высшим приоритетом	SFC 42	X	X

1) не для CPU 312IFM

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.6Information and Training Center
Knowledge for Automation

Передача наборов данных

Имеется область системных данных, которая может считываться, а также и записываться для параметрирования модулей. Эта область содержит наборы данных (записи), которые нумеруются от 0 до 255, и которые содержат данные-параметры и диагностические данные. Например,

- **SFC 55** передает новые параметры адресуемому модулю. Область SDB, находящаяся CPU, и связанная с этим модулем не переписывается.
- **SFC 56** передает параметры (набор данных RECNUM) из в адресуемый модуль.

- **SFC 57** передает все наборы данных из SDB в модуль.
- **SFC 58** передает набор данных RECORD в модуль.
- **SFC 59** считывает набор данных RECORD в модуль.

Прерывания по времени

Имеются блоки, которые используются для процессов, управляемых по времени (OB 10 - OB17). Параметры этих блоков могут быть заданы в STEP 7 или с помощью системных функций в прикладной программе.

- **SFC 28** устанавливает новую дату и время для запуска OB.
- **SFC 29** стирает дату и время запуска OB (с 10 по 17).
- **SFC 30** разрешает прерывание OB в установленное время.
- **SFC 31** считывает информацию о статусе OB (загружен, активирован, ...)

Прерывания по задержке

Прерывания с задержкой (OB 20 до 27) вызываются спустя время после того как событие произошло. С помощью **SFC 32** Вы определяете время задержки и номер OB прерывания. С помощью **SFC 33** Вы можете отменить прерывание с задержкой. **SFC 34** считывает статус OB.

Синхронные ошибки

С помощью этих функций Вы можете маскировать синхронные ошибки. Так что, OB ошибки не вызывается или, напротив, демаскируется. Считывая

Прерывания и асинхронные ошибки

регистр статуса (**SFC38**), Вы можете определить были ли ошибки и их тип. Эти SFC Вы используете для разрешения или блокировки прерываний, или OB асинхронных ошибок, или задержки прерываний, имеющих более высокий приоритет.

Обзор системных функций (часть 3)

Функц. группа	Функция	Блок	S7-300	S7-400
Системная диагностика	Чтение стартовой информации	SFC 6	-	X
	Чтение состояния системы	SFC 51	X	X
	Запись в диагностический буфер	SFC 52	X	X
Отображение процесса для области I/O	Актуализация области PII	SFC 26	-	X
	Актуализация области PIQ	SFC 27	-	X
	Установка битового массива	SFC 79	-	X
	Очистка битового массива	SFC 80	-	X
Адресация модулей	Определение логического адреса	SFC 5	-	X
	Определение слота	SFC 49	X	X
	Определение всех логич. адресов	SFC 50	X	X
Децентрализованная периферия	Прерывание от процесса	SFC 7	1)	1)
	Синхронизация DP-Slaves	SFC 11	1)	1)
	Чтение данных диагностики	SFC 13	1)	1)
	Чтение данных	SFC 14	1)	1)
	Запись данных	SFC 15	1)	1)
Передача глобальных данных	Передача GD пакета	SFC 60	-	X
	Прием GD пакета	SFC 61	-	X

1) только у CPU с DP-интерфейсом, например, CPU 315-2 DP

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.7



Information and Training Center
Knowledge for Automation

Системная диагностика

- **SFC 6** читает стартовую информацию OB, вызванного последним и не закончившим свою обработку, и текущего OB.
- **SFC 51** считывает часть списка состояний системы, который содержит: системные данные, данные диагностики, диагностический буфер.
- **SFC 52** записывает сообщение в диагностический буфер.

Отображение процесса

- **SFC 26** актуализирует всю или часть таблицы отображения входов.
- **SFC 27** передает всю или часть таблицы отображения выходов на выходные модули.
- **SFC 79/80** служат для установки/сброса битового массива в I или Q области. Функции могут работать под управлением функции Master Control Relay.

Адресация модулей

- **SFC 5** определяет логический адрес по расположению модуля (по географическому адресу).
- **SFC 49** определяет географический адрес по логическому адресу модуля.
- **SFC 50** определяет все логические адреса модуля.

Децентрализованная периферия

- **SFC 7** запускает процесс прерываний (OB40) на станции-мастере от интеллектуальной станции -slave (CPU 315-2DP).
- **SFC 11** синхронизирует одну или несколько групп DP-slaves
- **SFC 13** считывает данные диагностики от DP slave.
- **SFC 14** считывает консистентные данные (информация неделимая, как одно целое) из DP slave.
- **SFC 15** записывает консистентные данные в DP slave.

Передача глобальных данных

Глобальные данные передаются циклически (например, каждый 8-ой цикл) без использования SFC. С помощью **SFC 60** и **SFC 61** в программе пользователя может запускаться процедуры приема и передачи GD-пакета. Использование этих функций связано, в основном, с необходимостью обеспечить однородность передаваемых данных.

Обзор системных функций (часть 4)

Функц. группа	Функция	Блок	S7-300	S7-400
Обмен данными через SFB, конфигурируемые соединения	Опрос состояния	SFC 62	-	X
	Передача без координации	SFB 8	-	X
	Прием без координации	SFB 9	-	X
	Передача блока	SFB 12	-	X
	Прием блока	SFB 13	-	X
	Чтение данных из удаленного CPU	SFB 14	-	X
	Запись данных в удаленный CPU	SFB 15	-	X
	Передача данных на принтер	SFB 16	-	X
	Выполнение полного рестарта	SFB 19	-	X
	Перевод в стоп удаленный CPU	SFB 20	-	X
	Выполнение повторного пуска	SFB 21	-	X
	Опрос состояния удаленного CPU	SFB 22	-	X
	Прием состояния удаленного CPU	SFB 23	-	X
Обмен данными через SFC, не-конфигурируемые соединения	Расширенная передача данных	SFC 65	1)	1)
	Расширенный прием данных	SFC 66	1)	1)
	Расширенный прием данных	SFC 67	1)	1)
	Расширенная запись данных	SFC 68	1)	1)
	Отмена расширенных соединений	SFC 69	1)	1)
	Внутренний прием данных	SFC 72	1)	1)
	Внутренняя запись данных	SFC 73	1)	1)
	Отмена внутренних соединений	SFC 74	1)	1)

1) только для новых CPU

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.8Information and Training Center
Knowledge for Automation

Обмен данными через SFB

SFB обеспечивают обмен данными и управление программами.

В зависимости от того, где вызываются SFB, у одного или у двоих участников по коммуникации, различают *одностороннюю* или *двустороннюю* коммуникацию.

- **SFC 62** определяет состояние SFB по его экземпляру и состояние соответствующего соединения.
- **SFB 8** посылает данные удаленному партнеру без координации.
- **SFB 9** является дополнением к SFB 8 (прием данных).
- **SFB 12** посылает данные (до 64 Кбайт) удаленному партнеру с подтверждением.
- **SFB 13** принимает данные удаленного партнера с подтверждением.
- **SFB 14** читает данные удаленного CPU (односторонняя коммуникация).
- **SFB 15** записывает данные на удаленный (односторонний обмен).
- **SFB 16** посылает данные с форматированием удаленному принтеру.
- **SFB 19** запускает комплексный рестарт на удаленном CPU.
- **SFB 20** переводит удаленный в STOP.
- **SFB 21** выполняет рестарт на удаленном CPU.
- **SFB 22** опрашивает состояние удаленного CPU (режим работы, признак ошибки).
- **SFB 23** принимает статус состояния удаленного CPU.

Обмен данными через SFC

Для этой коммуникации, также рассматриваемой в качестве основной, имеются следующие различия по сравнению с коммуникацией через SFB:

- Возможна также для S7-300
- Не требуется конфигурирования соединения
- Не требуется связанных блоков данных
- Максимальная длина данных 76 байт
- Активная конфигурация соединения
- Коммуникация по MPI или K шине

Обзор системных функций (часть 5)

Функ. группа	Функция	Блок	S7-300	S7-400
Встроенное регулирование	Непрерывное регулирование	SFB 41	3)	-
	Шаговое регулирование	SFB 42	3)	-
	Формирователь импульсов	SFB 43	3)	-
Гибкая технология	Вызов ассемблерного блока	SFC 63	1)	-
Интегрированные функции	Высокоскоростной счетчик	SFB 29	2)	-
	Частотомер	SFB 30	2)	-
	A/B счетчик	SFB 38	3)	-
	Позиционирование	SFB 39	3)	-
IEC - таймер и IEC счетчик	Генерирование импульса	SFB 3	X	X
	Задержка включения	SFB 4	X	X
	Задержка выключения	SFB 5	X	X
	Прямой счет	SFB 0	X	X
	Обратный счет	SFB 1	X	X
	Прямой и обратный счет (реверс)	SFB 2	X	X
Управление сообщениями	Сообщения без индикации квитирования	SFB 36	-	X
	Сообщения с индикацией квитирования	SFB 33	-	X
	Сообщения с 8 сигналами	SFB 35	-	X
	Сообщения без сопровождающих сигналов	SFB 34	-	X
	Передача архивных данных	SFB 37	-	X
	Блокировка сообщений	SFC 10	-	X
	Разрешение сообщений	SFC 9	-	X

1) только для CPU 614

2) только для CPU 312 IFM

3) только для CPU 314IFM

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.9Information and Training Center
Knowledge for Automation**Встроенное регулирование**

Эти блоки будут интегрированы в последующие версии CPU. Их аналоги находятся в библиотеке PID-Control.

Гибкая технология

Для CPU 614 (S7-300) отдельные блоки могут быть созданы на языке ассемблера (или на языке "C", а затем транслированы). Системная функция SFC 63 запускает такие блоки.

Интегрированные функции

Эти блоки имеются только у CPU 312 IFM (S7-300).
Описание этих системных функций находится в руководстве Интегрированные функции.

- **SFB 29** подсчитывает импульсы на встроенных входах CPU.
- **SFB 30** измеряет частоту сигнала на встроенных входах.

**IEC таймер
IEC счетчик**

Эти блоки реализуют функции времени и счетчика в соответствии со стандартом IEC 1131-3. Остающиеся в языке STEP 7 таймерные функции и функции счета реализуются также, как и для SIMATIC S5, по причине совместимости. Таймеры и счетчики стандарта IEC обладают большим диапазоном значений.

Сообщения, зависящие от блоков

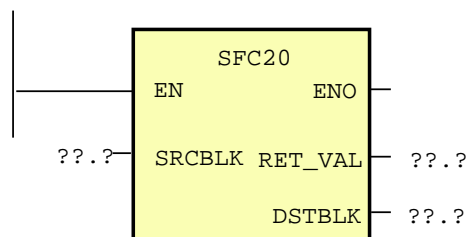
Эти блоки служат для создания системных сообщений, например для контроля системы.
Необходимой предпосылкой является то, что зарегистрировано хотя бы одно дисплейное устройство (OP). Применяется концепция централизованного квитирования. Это означает, что если Вы подтвердили сообщение на устройстве индикации, то ответ посылается в CPU, порождающее сообщение. Оттуда информация посылается по всем зарегистрированным абонентам. Сообщения запускаются по изменению фронта сигнала на входе.

Вызов системных функций и системных функциональных блоков

Системные функции:

Красное → `CALL SFC 20`
`SRCBLK :=`
`RET_VAL :=`
`DSTBLK :=`

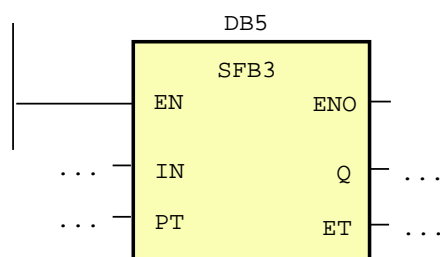
Вызов в STL



Вызов в LAD

Системные функциональные блоки:

Черное → `CALL SFB 3, DB5`
`IN :=`
`PT :=`
`Q :=`
`ET :=`



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.10



Information and Training Center
Knowledge for Automation

Системные функциональные блоки

Системные функциональные блоки SFB - функциональные блоки, которые интегрированы в операционную систему S7-CPU.
 В результате, SFB не нужно загружать в CPU как часть программы пользователя.
 Точно так же, как FB, SFB - блоки "с памятью". При их вызове в программе пользователя им должен быть назначен экземпляр DB.

Системные функции Системные функции - функции, которые интегрированы в операционную систему S7-CPU.
 SFC могут вызываться из программы пользователя точно так же, как FC.
 Точно так же как FC, SFC - блоки "без памяти".

Вызов

Когда вызывается системная функция, она автоматически копируется в программу пользователя.
 Кроме того, все системные функции хранятся в библиотеке *Standard Library V3.x, S7-Program System Function Blocks*. Вы можете также копировать SFC и SFB в программу пользователя *непосредственно* из этой библиотеки. В библиотеке существует таблица символов (с английскими обозначениями). Символы используемых блоков автоматически копируются в таблицу символов программы пользователя.

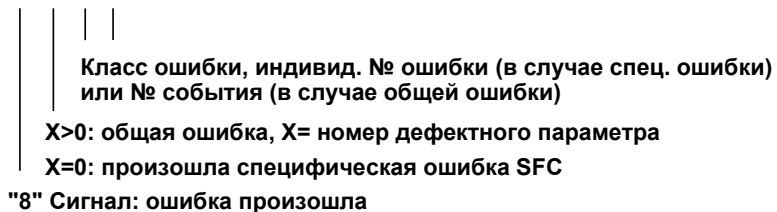
Оценка сообщений об ошибках

Опрос BR-bit (Binary Result) возвращает RLO=0 при ошибочной обработке блока и RLO=1 при обработке без ошибки.

- Опрос BR в STL: A BR
- Опрос в LAD использует выходной параметр ENO

Большинство системных функций возвращают коды ошибки со следующей структурой в выходном параметре RET_VAL (INT):

- RET_VAL=W#16#8 X Y Z



- Примеры:
 - W#16#8081 код специфической ошибки SFC.
 - W#16#823A код общей ошибки; ошибка была вызвана параметром № 2.

SIMATIC S7

Siemens AG 1999. All rights reserved.

 Date: 04.11.2005
 File: PRO2_07E.11

 Information and Training Center
 Knowledge for Automation

Информация об ошибке

Выполненный SFC информирует Вас в программе пользователя, смог ли CPU успешно выполнить функцию SFC или нет. Вы получаете соответствующую информацию об ошибке двумя способами:

- в BR-бите слова состояния и
- в выходном параметре RET_VAL (возвращаемое значение).

Обратите внимание

Вы должны всегда поступать следующим образом перед использованием значения выходного параметра SFC:

- сначала оценить BR-бит слова состояния. Если он равен 1, то
- проверить выходной параметр RET_VAL, чтобы определить, какая именно ошибка произошла.

Если произошла ошибка (BR=1 и RET_VAL<>0), то значениями выходных параметров пользоваться нельзя.

Общие ошибки

Код общей ошибки указывает на ошибки, которые могут происходить со всеми системными функциями. Общий код ошибки состоит из следующих двух чисел:

- номер параметра между 1 и 127 (биты в RET_VAL от 8 до 14), 1- первый параметр, 2 - второй параметр и т.д. у вызванного SFC .
- номер события между 0 и 127 (биты в RET_VAL от 0 до 7). Номер события указывает на синхронную ошибку.

Полное описание общих кодов ошибок может быть найдено в руководстве: "Системные и стандартные функции" или в Online Help.

Специфические ошибки

Некоторые системные функции (SFC) могут через возвращаемое значение, передавать код специфической ошибки. Этот код ошибки указывает, что во время обработки функции произошла ошибка, которая принадлежит только данной системной функции.

Описание кодов специфических ошибок можно найти в Online Help для системных функций.

Упражнение 7.1: Генерирование DB с атрибутом "UNLINKED"

Address	Name	Type	Initial Value	Actual Value	Comment
0.0	Recipe[1]	INT	1	1	
2.0	Recipe[2]	INT	2	2	
4.0	Recipe[3]	INT	3	3	
6.0	Recipe[4]	INT	4	4	
8.0	Recipe[5]	INT	5	5	
10.0	Recipe[6]	INT	6	6	
12.0	Recipe[7]	INT	7	7	
14.0	Recipe[8]	INT	8	8	
16.0	Recipe[9]	INT	9	9	
18.0	Recipe[10]	INT	10	10	
20.0	Recipe[11]	INT	11	11	
22.0	Recipe[12]	INT	12	12	
24.0	Recipe[13]	INT	13	13	
26.0	Recipe[14]	INT	14	14	
28.0	Recipe[15]	INT	15	15	
30.0	Recipe[16]	INT	16	16	
32.0	Recipe[17]	INT	17	17	

Загрузочная память

Перенос

DB 20

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.12Information and Training Center
Knowledge for Automation

Цель упражнения Вы производит блок данных с атрибутом "UNLINKED".

Задача

Так как рабочая память имеет ограниченный (обычно слишком мала) размер, несколько блоков данных с различными значениями одного и того же рецепта для управления этим рецептом должны быть сохранены только в загрузочной памяти.

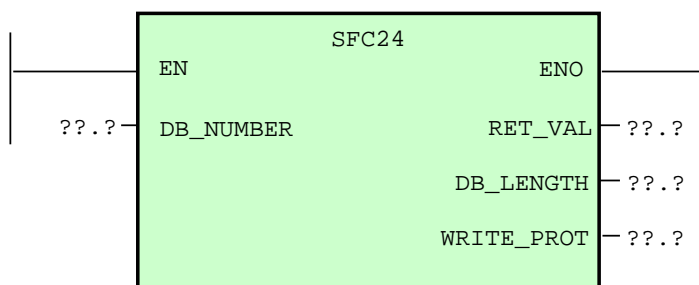
Только рабочий DB, в котором сохранен текущий рецепт, присутствует в рабочей памяти. Для замены рецепта, требуемые значения копируются в загрузочную память из рабочей памяти.

С помощью атрибута "UNLINKED" Вы добиваетесь того, что блоки данных во время передачи из PG в CPU попадают только в загрузочную память и автоматически не скопируются в рабочую память.

Что делать

1. Вставить DB20.
2. Объявить в DB20 переменную "Recipe" типа ARRAY[1..20] OF INT.
3. С помощью команды меню View -> Data View, инициализировать индивидуальные компоненты массива в порядке возрастания.
4. Выбрать свойства блока, и установить атрибут "UNLINKED".
5. Передать блок данных DB 20 в CPU.
6. Что случается когда Вы, например, доступ к DB 20 в пользовательской программе организуете с помощью инструкции L DB20.DBW0?

Упражнение 7.2: Тестирование блока данных (SFC 24: только для S7-400)



Параметры	Объявления	Тип данных	Области памяти	Описание
DB_NUMBER	INPUT	WORD	I, Q, M, D, L, Const.	Номер DB для проверки
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Информация об ошибке
DB_LENGTH	OUTPUT	WORD	I, Q, M, D, L	Кол-во байт в выбранном DB
WRITE_PROT	OUTPUT	BOOL	I, Q, M, D, L	Информация о защите блока от записи (1 - блок защищен)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.13Information and Training Center
Knowledge for Automation

Цель упражнения С помощью SFC 24 Вы должны определить, существует ли определенный блок данных в рабочей памяти или нет.

Задача Созданный блок FC 72 с помощью SFC 24 определяет, существует ли DB в рабочей памяти, в загрузочной памяти или не существует вообще в CPU:

- во входном параметре #DB_NUM (WORD) в FC 72 передается номер блока, который будет проверен.
- FC 72 возвращает результат вызывающему блоку в параметре #RET_VAL (INT) :
 - 1: DB существует в загрузочной памяти
 - 0: DB существует в рабочей памяти
 - -1: DB не существует

Обратите внимание SFC 24 в #RET_VAL возвращает следующие идентификаторы ошибки:

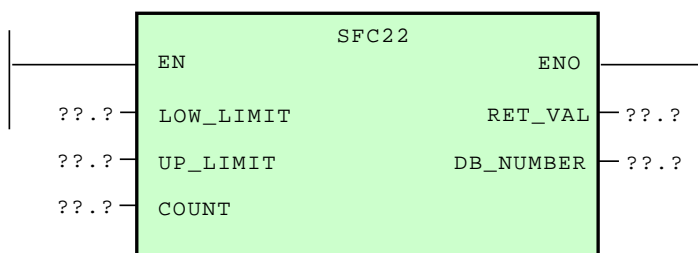
- **W#16# 0000** Нет ошибки
- **W#16# 80A1** Некорректное число в параметре DB_NUMBER (0 или > максимального номера DB)
- **W#16# 80B1** DB не существует в CPU
- **W#16# 80B2** DB был создан с атрибутом "UNLINKED" (находится только в загрузочной памяти)

Что делать

1. Создать FC 72 блок
2. Создать OB1, вызвать из него FC 72 для проверки DB 20. Вывести возвращаемую информацию на дисплей тренажера.
3. Загрузить блоки в CPU, и проверить Вашу программу.

Обратите внимание Системная функция SFC 24 существует только в S7-400!

Упражнение 7.3: Создание DB (SFC 22)



Параметры	Объявления	Тип данных	Область памяти	Описание
LOW_LIMIT	INPUT	WORD	I, Q, M, D, L, Const.	Наименьший номер DB
UP_LIMIT	INPUT	WORD	I, Q, M, D, L, Const.	Наибольший номер DB
COUNT	INPUT	WORD	I, Q, M, D, L, Const.	Кол-во <i>байт</i> в блоке; здесь должно быть определено <i>четное</i> число.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Информация об ошибке
DB_NUMBER	OUTPUT	WORD	I, Q, M, D, L	Номер созданного DB, лежит между LOW_LIMIT и UP_LIMIT

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.14Information and Training Center
Knowledge for Automation

Цель упражнения Вы знакомитесь с созданием нового DB в программе.

Задача При запуске OB100 DB 10 должен быть сгенерирован в рабочей памяти. Позже значения рецепта должны быть скопированы из загрузочной памяти в этот DB

Что делать

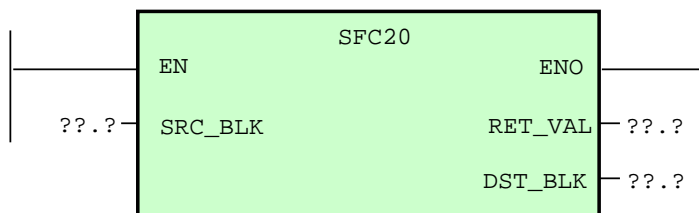
1. Создать OB 100.
2. Создать DB 10 длиной в 20 слов в OB100.
Для этого использовать SFC 22 (см. выше). Сохранить параметр #RET_VAL в MW 0, а параметр #DB_NUMBER показать на дисплее тренажера.
3. Загрузить OB 100 в CPU и проверить Вашу программу.

Обратите внимание Вы должны обратить внимание, что доступ к "медленной" загрузочной памяти требует значительно большего количества времени, чем доступ к "быстрой" рабочей памяти.
Если копируются большие объемы в OB1, может оказаться что нужно запустить контроль времени цикла снова (SFC 43).

Идентификаторы ошибок Системная функция SFC 22, используя параметр #RET_VAL, обеспечивает следующие сообщения об ошибках:

- W#16# 0000 Нет ошибок
- W#16# 8091 Глубина вложения превышена
- W#16# 8092 Сжатие в настоящее время активно
- W#16# 80A1 Неправильный номер DB
- W#16# 80A2 Неправильная длина
- W#16# 80B1 Нет доступного номера DB (DB уже существует)
- W#16# 80B2 Не достаточно памяти
- W#16# 80B3 Не достаточно непрерывной памяти (требуется сжатие)

Упражнение 7.4: Копирование DB из загрузочной памяти в рабочую память (SFC 20)



Параметры	Объявление	Тип данных	Область памяти	Описание
SRC_BLK	INPUT	ANY	I, Q, M, D, L	Область памяти для копирования (=массив-источник). Эта область может представлять не последовательные блоки в загрузочной памяти (DB, может быть откомпилирован с атрибутом UNLINKED)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Код ошибки SFC
DST_BLK	OUTPUT	ANY	I, Q, M, D, L	Область памяти, куда копируется блок (целевой массив)

SIMATIC S7

Siemens AG 1999. All rights reserved.

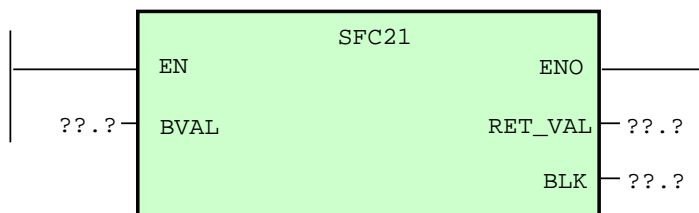
Date: 04.11.2005
File: PRO2_07E.15Information and Training Center
Knowledge for Automation

Цель упражнения Вы знакомитесь с системной функцией SFC 20 (BLKMOV).

Задача Значения рецепта (DW0-DW19) должны быть скопированы из блока данных DB 20 в DB10 (DW0-DW19) в рабочей памяти. Копирование имеет место при возникновении положительного фронта на входе I 0.0.

- Что делать**
1. Создать OB1, который копирует значения рецепта из DB 20 в DB 10 с помощью SFC20 (BLKMOV) при возникновении положительного фронта на входе I 0.0.
 2. Показать значение #RET_VAL на цифровом дисплее тренажера
 3. Загрузить Вашу программу в CPU и проверить ее.

Дополнительное упражнение 7.5: Инициализация DB (SFC 21)



Параметры	Объявление	Тип данных	Область памяти	Описание
BVAL	INPUT	ANY	I, Q, M, D, L	Исходный массив
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Код ошибки SFC
BLK	OUTPUT	ANY	I, Q, M, D, L	Целевая область, инициализируемая содержимым BVAL

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.16Information and Training Center
Knowledge for Automation

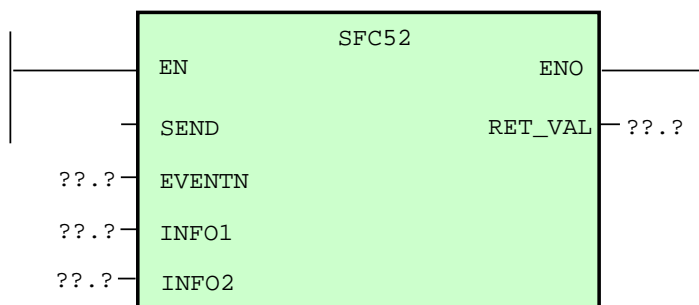
Цель упражнения Знакомство с использованием системных функций .

Задача Создать FC 75, с помощью которого блоки данных могут быть инициализированы. FC 75 должна иметь следующие функциональные возможности:

- FC 75 имеет следующие входные параметры :
 - # DB_NUM (WORD): Номер DB, для инициализации
 - #INI (BYTE): образец байта, который копируется во все ячейки памяти DB.
- FC 75 сначала должна определить, существует ли нужный DB в рабочей памяти. Если он существует, то определяется его длина. Далее FC 75 инициализирует блок переданным байтом.
- FC 75 передает в # RET_VAL (BOOL)
 - TRUE: DB был успешно инициализирован.
 - FALSE: DB не был инициализирован, то есть DB не существует в рабочей памяти

- Что делать**
1. Создать FC 75.
 2. Вызвать FC 75 в OB1 так, чтобы DB 10 был инициализирован значением "0" при положительном фронте на I1.1.
 3. Загрузить Вашу программу в CPU и проверить ее.

Дополнительное упражнение 7.6: Запись сообщения в диагностический буфер (SFC 52)



Параметры	Объявление	Тип данных	Область памяти	Описание
SEND	INPUT	BOOL	I, Q, M, D, L, Const.	Разрешение отправки сообщения во все зарегистрированные узлы
EVENTN	INPUT	WORD	I, Q, M, D, L, Const.	Номер или тип события (идентификатор события)
INFO1	INPUT	ANY	I, Q, M, D, L	Дополнит. информация (1 слово)
INFO2	INPUT	ANY	I, Q, M, D, L	Дополнит. информация (2 слово)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Код ошибки

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.17Information and Training Center
Knowledge for Automation

Цель упражнения Научиться передавать входные сообщения в диагностический буфер.
Задача Создать FC 76 со следующими функциональными возможностями:

- При ошибке системы (моделируемой фронтом импульса в I1. 2), введется сообщение в диагностический буфер.

Что делать

1. Создайте блок FC 76, который вводит сообщение в диагностический буфер, когда имеется "ошибка системы" (фронт в I1. 2).
2. Активизируйте функцию "CPU Messages" в SIMATIC Manager
3. Вызовите FC 76 в OB1, и проверьте Вашу программу.

Обратите внимание Используйте следующие параметры для SFC 52:

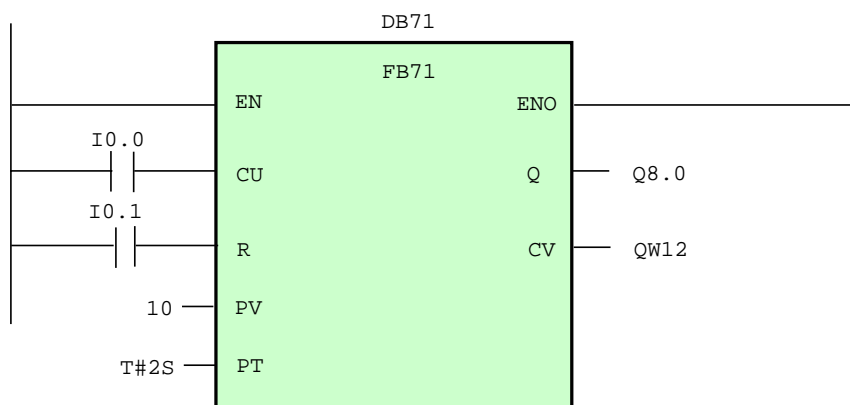
- EVENTN W#16# 9B0A (ошибка конечного состояния, приходящее событие, внешняя ошибка, вход в диагностический буфер)
- INFO1 W#16# 8 (Например, предел числа переключений)
- INFO2 DW#16# 1 (Например, тип выключателя)

Событие с номером 9 (Event ID=9) - доступ для пользователя (см.руководство "Системные и стандартные функции").

Коды ошибок Следующие сообщения об ошибке передаются SFC 52 через #RET_VAL :

- 8083 Не допустимый тип данных INFO1
- 8084 Не допустимый тип данных INFO2
- 8085 Не допустимый код EVENTN
- 8086 Не допустимая длина INFO1
- 8087 Не допустимая длина INFO2
- 8091 Узел не зарегистрирован
- 8092 Пересылка в данный момент невозможна (буфер пересылки заполнен)

Дополнительное упражнение 7.7: Счетчик



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.18Information and Training Center
Knowledge for Automation

Задача

Создать блок -16-разрядный счетчик (прямой счет) FB71 "CU" со следующими свойствами:

- Счетчик увеличивается на 1 положительным импульсом более длинным, чем время PT на входе CU.
- В противном случае счетчик имеет те же самые характеристики, что и IEC-счетчик SFB0 "CTU"(прямой счет).
- Выход Q указывает, является ли текущее значение счетчика большим или равным, чем заданное на входе PV.

Параметры

Параметр	Декларация	Тип данных	Описание
CU	INPUT	BOOL	Вход счетчика (прямой счет)
R	INPUT	BOOL	Вход сброса, доминирует над CU.
PV	INPUT	INT	Инициализирующее значение.
PT	INPUT	TIME	Период времени, в течение которого уровень сигнала должен быть в 1 после фронта, чтобы счетчик был увеличен на 1.
Q	OUTPUT	BOOL	Статус счетчика: Q =1 если CU >PV, иначе 0
CV	OUTPUT	INT	Текущее значение

Что делать

1. Создайте FB71 с необходимыми свойствами. Используйте системные функциональные блоки SFB0 и SFB4.
2. Вызовите блок-счетчик FB71 с экземпляром DB71 в OB1. Назначте блоку параметры со следующими фактическими значениями:

- CU = I 0.0	- R = I 0.1
- PV = I W4	- PT = T#1000MS
- Q = Q8.0	- CV = QW12 (числовой дисплей на симуляторе)
3. Загрузите блоки в CPU и протестируйте программу.

Библиотека: S5-S7 Converting Blocks



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.19



Information and Training Center
Knowledge for Automation

Введение

Эта библиотека содержит стандартные S7-блоки, необходимые для преобразования S5-программ. Это означает, например, что если в S5-программе есть стандартный блок FB 240, то он будет заменен на FC 81. Так как преобразователь только передает вызов блока FC 81, Вы должны скопировать указанный блок из библиотеки в Вашу S7-программу.

Содержание библиотек

Библиотечные блоки разделены на следующие функциональные группы:

- Арифметика с плавающей запятой, типа сложения и вычитания
- Сигнальные функции
- Интегрированные функции, типа преобразования BCD-кода в двоичный
- Основные логические функции, типа LIFO

Руководства

Подробно блоки описаны в руководстве "Преобразование программ STEP 5".

Интерактивная помощь

В текстовом программном редакторе, Вы можете вызвать *Help --> Help topics --> References --> Additional reference aids --> Help with S5/S7 functions*.

Замечание

Так называемые оперативные флаги также используются для этих блоков, как это типично для SIMATIC S5.

Библиотека: TI-S7 Converting Blocks (Часть 1)

Блок	Имя	Описание
FC 80	TONR	Запуск таймера с задержкой включения
FC 81	IBLKMOV	Косвенный перенос области данных
FC 82	RSET	Сброс области меркеров или области I/O
FC 83	SET	Установка области меркеров или области I/O
FC 84	ATT	Ввод значений в таблицу
FC 85	FIFO	Вывод первой табличной величины
FC 86	TBL_FIND	Поиск значения в таблице
FC 87	LIFO	Вывод последней табличной величины
FC 88	TBL	Выполнение табличной операции
FC 89	TBL_WRD	Копирование значения из таблицы
FC 90	WSR	Сохранение данных в сдвиговом регистре
FC 91	WRD_TBL	Логическое объедин. с элементом таблицы и его сохранение
FC 92	SHRB	Сдвиг бита в сдвиговом регистре
FC 93	SEG	Создание битового образа для 7-сегментного дисплея
FC 94	ATH	Преобразование ASCII-кода в 16-ичное число
FC 95	HTA	Преобразование 16-ичного числа в ASCII-код
FC 96	ENCO	Установка определенного бита в слове
FC 97	DECO	Считывание номера бита наименьшего значащего разряда
FC 98	BCDCPL	Создание десятичного дополнения
FC 99	BITSUM	Вычисление числа установленных битов

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.20Information and Training Center
Knowledge for Automation**FC 80**

Функция FC80 - таймер с задержкой включения (TONR). FC80 "отсчитывает" время, пока текущее значение времени (#ET) не станет больше или равно заданному времени (* PV). Так как функция TONR для текущего времени использует время вызова последнего OB, то ее целесообразно применять в циклических OB (OB1, OB30-OB38).

FC 81

С помощью функции FC81 (IBLKMOV) Вы можете передавать данные области данных, состоящих из байтов, слов, целых чисел (16 бита), двойных слов, или двойных целых чисел (32 бит) из источника в приемник. Параметры #S_DATA и #D_DATA типа POINTER - указатели на начало источника и приемника. Длина области, которую нужно копировать, определяется через отдельные параметры.

FC 82/83

Устанавливают состояния битов в указанной области в "1" (FC 83) или в "0" (FC 82), если бит MCR равен "1". Если бит MCR равен "0", то состояние битов в области не изменяется.

FC 84-FC92

Эти функции предназначены для работы с таблицами, например, реализуют функцию FIFO. Входные значения должны быть в формате слова, их количество регулируется.

FC 93-FC 99

В этой группе содержатся различные функции преобразования.

Библиотека: TI-S7 Converting Blocks (Часть 2)

Блок	Имя	Описание
FC 100	RSETI	Сброс области ввода
FC 101	SETI	Установка области ввода
FC 102	DEV	Среднеквадратическое отклонение
FC 103	CDT	Коррелированные таблицы данных
FC 104	TBL TBL	Логические операции с таблицами
FC 105	SCALE	Масштабирование значения
FC 106	UNSCALE	Немасштабированная величина
FB 80	LEAD_LAG	Алгоритм опережения/задержки
FB 81	DCAT	Дискретное управляющее прерывание
FB 82	MCAT	Прерывание управления двигателем
FB 83	IMC	Сравнение индексов матриц
FB 84	SMC	Сканирование матриц
FB 85	DRUM	DRUM (процессор последовательностей)
FB 86	PACK	Сбор/распределение табличных данных

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.21Information and Training Center
Knowledge for Automation

FC 100-FC 101

Функции установки (RSETI) и сброса (SETI) битового массива в области периферии (P-область). Если бит MCR равен "1" - функции устанавливают или сбрасывают массив. Если бит MCR равен "0", состояние битов не изменяется.

FC 102

Функция вычисляет среднеквадратическое отклонение (DEV) группы значений, сохраненных в таблице (TBL). Результат возвращается в OUT. Среднеквадратичное отклонение рассчитывается согласно следующей формуле:

$$\text{Стандартное отклонение} = \sqrt{\frac{(N \times \text{SqSum}) - \text{Sum}^2}{N \times (N - 1)}}$$

где:

- #Sum = сумма значений в TBL N = число значений в TBL
- #SqSum = сумма квадратов всех значений в TBL

FC 103

Функция сравнивает значения коррелированных таблиц данных (CDT) на входах (#IN) и (#IN_TBL) и ищет первое значение, которое больше или равно, чем значение на входе. С помощью индекса обнаруженного значения, величина затем копируется в соответствующее значение выхода (#OUT) в таблице выходных значений (#OUT_TBL).

FC 104-FC 105

Используется для масштабирования значений аналоговых входных/выходных модулей.

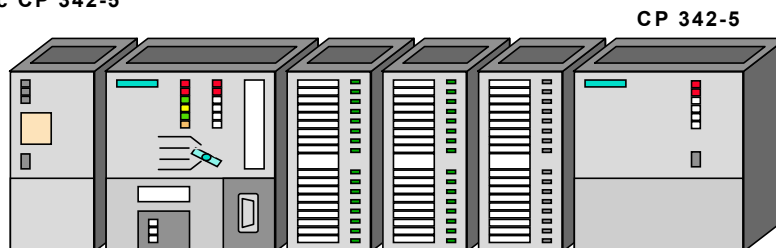
FB 80- FB 86

См. электронное руководство.

Библиотека: Communication Blocks

Блок	Имя	Описание
FC 1	DP_SEND	Посылка данных PROFIBUS-CP
FC 2	DP_RECV	Прием данных PROFIBUS-CP
FC 3	DP_DIAG	Чтение диагностических данных
FC 4	DP_CTRL	Посылка управляющего задания в CP

Исключительно для конфигурации:
S7-300 CPU с CP 342-5



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.22



Information and Training Center
Knowledge for Automation

Краткий обзор

Библиотечные функции FC1, FC2, FC3 и FC4 используются исключительно в следующей конфигурации:

- S7-300 CPU с внешним PROFIBUS CP 342-5

Во всех других случаях, то есть для S7-300 со встроенным интерфейсом PROFIBUS-DP и для системы S7-400 соответствующие функции могут быть выполнены, используя стандартные загрузки и переходы (L. ..., T. ...) или используя SFC14 (DPRD_DAT), SFC15 (DPWR_DAT), SFC11 (DPSYC_FR) и SFC13 (DPNRM_DG).

FC1

Блок DP_SEND передает данные из указанной области в PROFIBUS-CP для передачи модулю распределенного ввода - вывода.

FC2

Блок DP_RECV принимает данные процесса из модулей распределенного вывода, а также информацию о состоянии DP-модуля.

FC3

Блок DP_DIAG используется для запроса диагностической информации. Дифференцирование сделано между следующими типами задач:

- Запрашивается список станций DP;
- Запрашивается список диагностический список DP;
- Запрашивается диагностика отдельного DP ;
- Нециклическое чтение входных / выходные данные DP-slave;
- Чтение рабочего режима DP.

FC4

Блок DP_CTR передает задачи управления в PROFIBUS-CP. Дифференцирование сделано между следующими типами задач:

- Глобальное управление нециклическое / циклическое;
- Удалить старую диагностику;
- Установка текущего рабочего режима DP ;
- Установка рабочего режима DP для останова PLC/CP;
- Циклическое чтение данных ввода - вывода;
- Установка рабочего режима DP-slave.

Библиотека: PID Control Blocks

Блок	Имя	Описание
FB 41	CONT_C	Блок для непрерывного PID управления
FB 42	CONT_S	Блок для для шагового PI управления
FB 43	PULSEGEN	Блок для генерирования импульсов

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_07E.23Information and Training Center
Knowledge for Automation**FB 41**

SFB "CONT_C" (регулятор непрерывного действия) используется в SIMATIC S7 программируемые логических контроллерах для управления техническими процессами с непрерывным входом и выходом. При назначении параметров, Вы можете включать или отключать подфункции PID регулятора для адаптации регулятора к процессу.

Вы можете использовать регулятор, как PID-регулятор с фиксированной уставкой или в многоконтурных регуляторах, как каскад. Функции регулятора основаны на PID алгоритме управления, который вызывается периодически, с аналоговым выходным сигналом, в случае необходимости расширенным включением каскада импульсного генератора, чтобы генерировать выходные сигналы в виде широтно-модулированных импульсов для двух или трех-шагового управления пропорциональными исполнительными механизмами.

FB42

SFB "CONT_S" (шаговый регулятор) используется в SIMATIC S7 программируемых логических контроллерах, чтобы управлять техническими процессами с цифровыми выходными сигналами для интегрирующих исполнительных механизмов. При назначении параметра, Вы можете включать или отключать подфункции шагового PI-регулятора для адаптации регулятора к процессу.

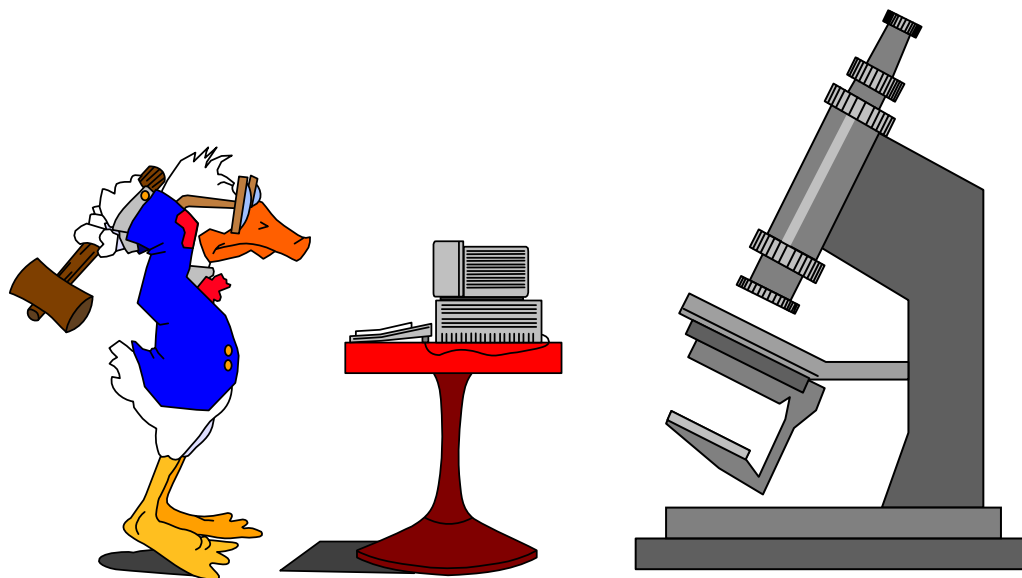
Вы можете использовать регулятор, как PI-регулятор с фиксированной уставкой или во вторичных петлях при каскадном управлении, однако не как первичный регулятор. Функции регулятора основаны на PI алгоритме управления, повторяемом циклически, дополненном функциями для производства двоичного выходного сигнала из аналогового сигнала возбуждения.

FB43

SFB43 "PULSEGEN" (импульсный генератор) используется, чтобы создавать структуры PID регуляторов с импульсным выходом для пропорциональных исполнительных механизмов.

При использовании SFB "PULSEGEN" он может быть сконфигурирован для двух- или трех-уровневого PID-регулирования с широтно-импульсной модуляцией. Функция обычно используется вместе с регулятором непрерывного действия "CONT_C".

Обработка синхронных и асинхронных ошибок



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.1Information and Training Center
Knowledge for Automation

Содержание

Стр

Обработка асинхронных ошибок	2
Обработка ошибок организационными блоками	3
Пример OB асинхронной ошибки	4
Обработка синхронных ошибок	5
Стартовая информация для программных ошибок в OB121	6
Стартовая информация для ошибок доступа OB122	7
Маскирование синхронных ошибок	8
SFC 36 для маскирования синхронных ошибок	9
Структура фильтра ошибок программирования	10
Структура фильтра ошибок доступа	11
SFC 37 для демаскирования синхронных ошибок	12
SFC 38 для чтения регистра ошибок	13
Пример: тестирование блока данных	14
Упражнение 8.1: обработка ошибок в FC81	15

Обработка асинхронных ошибок

Асинхронные ошибки не ставятся в соответствие какому-либо месту программы, то есть они кажутся асинхронными к выполнению программы.

Тип ошибки	Пример	ОВ- - обработчик
Временная ошибка	Превышено максимальное время цикла	ОВ 80
Ошибка источника питания	Неисправна буферная батарея	ОВ 81 ²⁾
Диагностическое прерывание	Обрыв провода на входе блока, способного к диагностике	ОВ 82
Прерывание при вставке/удалении модуля	Удалени сигнального модуля S7-400 в рабочем режиме	ОВ 83 ¹⁾
Аппаратная ошибка CPU	Дефектный уровень сигнала в MPI интерфейсе	ОВ 84 ¹⁾
Ошибка последовательности программы	Ошибка при обновлении таблицы отображения процесса (дефект модуля)	ОВ 85
Дефект носителя модулей (rack)	Ошибка источника питания в удаленном носителе модулей	ОВ 86 ¹⁾
Коммуникационные ошибки	Некорректный идентификатор сообщения	ОВ 87

¹⁾ только в S7-400

²⁾ не переходит в STOP при отсутствии ОВ ошибки

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.2



Information and Training Center
Knowledge for Automation

Введение

Эти ошибки не назначены никакому месту в программе.

Временные ошибки

В цикле сканирования контролируется максимальное время цикла, имеющее значение по умолчанию 150 ms. Система распознает ошибку времени, если продолжительность цикла больше, чем 150 ms. Если ошибка встречается дважды в том же самом цикле (подряд), CPU входит в состояние останова.

Ошибка источника питания

Встречается при выходе из строя или разрядке резервной батареи и, кроме того, в S7-400 при выходе из строя источника питания 24 V в центральной стойке или стойке расширения.

В отличие от других типов ошибок, CPU при данной ошибке и при отсутствии ОВ ошибки, остается в рабочем состоянии и зажигает красный светодиод ошибки на CPU.

Диагностическое прерывание

Блоки способные к диагностике, например специальные аналоговые блоки, могут вызывать диагностическое прерывание в случае ошибки. Блоку должны быть назначены параметры так, что диагностическое прерывание деблокируется.

Прерывание при удалении / вставке модуля

Вызывается при вставке или удалении модуля в системе PLC S7-400. При вставка блоков, операционная система проверяет, был ли вставлен блок правильного типа. Эта функция делает передвижение и вставку блоков возможной в течение цикла программы.

Аппаратная ошибка CPU

В S7-400 ошибки в MPI интерфейсе в К-шине или в интерфейсном блоке для распределенного ввода - вывода.

Ошибка программной последовательности

Следует при ошибке доступа при вводе - выводе при модификации таблицы изображения процесса или например, при неправильной параметризации ОВ прерывания по "дате и времени".

Выход из строя носителя модулей

Возникает, когда выходят из строя носитель модулей, подсеть в PLC системах с сетевой структурой или станции распределенного ввода - вывода.

Обработка ошибок организационными блоками

- Чтобы предотвратить останов CPU в случае ошибки, необходимо загрузить пустой организационный блок ошибки
- Вы можете запрограммировать желательную реакцию в ОБ ошибки и, если требуется, вызвать состояние останова с помощью системной функцией SFC 46 после выполнения ОБ ошибки
- Дополнительный идентификатор ошибки сохраняется в стартовой информации организационного блока ошибки, который может быть оценен в программе
- Описание организационного блока ошибки может быть найдено в Интерактивной справке или в Руководстве по системным и стандартным функциям
- Передача ОБ ошибок, не поддерживаемых CPU, отвергается с сообщением об ошибках

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.3Information and Training Center
Knowledge for Automation

Стартовая информация Для каждого организационного блока, в разделе описаний определены временные переменные. В этих переменных операционная система хранит стартовую информацию. В частности, операционная система хранит там дополнительную информацию относительно того, почему вызван блок. Например Вы можете видеть стартовую информацию OB 81.

Name	Type	Initial Value	Comment
OB81_EV_CLASS	BYTE		16#39, Event class 3, Entering ev
OB81_FLT_ID	BYTE		16#XX, Fault identification code
OB81_PRIORITY	BYTE		26/28 (Priority of 1 is lowest)
OB81_OB_NUMBR	BYTE		81 (Organization block 81, OB81)
OB81_RESERVED_1	BYTE		Reserved for system
OB81_RESERVED_2	BYTE		Reserved for system
OB81_MDL_ADDR	INT		Reserved for system
OB81_RESERVED_3	BYTE		Reserved for system
OB81_RESERVED_4	BYTE		Reserved for system
OB81_RESERVED_5	BYTE		Reserved for system
OB81_RESERVED_6	BYTE		Reserved for system
OB81_DATE_TIME	DATE_AND_TIME		Date and time OB81 started

Переменная OB81_FLT_ID имеет следующие значения :

- B#16#21: По крайней мере одна резервная батарея центральной стойки - пуста (BATTF)
- B#16#22: Резервное напряжение в центральной стойке пропало (BAF).
- B#16#23: Выход из строя/удаление источника питания 24V в центральной стойке.
- B#16#31: По крайней мере одна резервная батарея стойки расширения пуста
- B#16#32: Резервное напряжение в одной из стоек расширения пропало
- B#16#33: Выход из строя источника питания 24V в стойке расширения

Пример ОВ асинхронной ошибки

```

OB81: Error OB: Power supply failure (выход из строя источника питания)

Network 1: Выход из строя батареи, приходящее событие

L   #OB81_FLT_ID      // Загрузка идентификатора ошибки
L   B#16#22           // Идентификатор: выход из строя батареи в CR
==I
=   M   81.1          // Установка доп.меркерного бита
L   #OB81_EV_CLASS    // Идентификатор: приходящее, уходящее
L   B#16#39           // Идентификатор: приходящее событие
==I
=   M   81.2          // Доп.меркерный бит приходящего события
A   M   81.1          // Батарея вышла из строя и
A   M   81.2          // приходящее событие
S   M   81.0          // Установка доп.меркерного бита для показа
                        // ошибки

Network 2: Сброс доп.меркерного бита, если батарея О.К.

L   #OB81_EV_CLASS    // Идентификатор: приходящее, уходящее
L   B#16#38           // Идентификатор: уходящее
==I
R   M   81.0          // Сброс доп.меркерного бита

```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.4



Information and Training Center
Knowledge for Automation

Задача

Выход из строя батареи должен привести к индикации ошибки на пульте управления. После замены батареи индикация должна автоматически исчезнуть.

Описание

При ошибках питания, например, выходе из строя батареи, организационный блок ошибки вызывается операционной системой один раз. После того, как ошибка устранена, ОВ 81 вызывается еще раз.

В примере программы переменная #OB81_FLT_ID оценивается, чтобы определить, имелся ли выход из строя батареи. В этом случае переменная содержит значение 22. При этом выполняется команда сравнения и устанавливается меркер M 81.1.

Индикация ошибки должна появиться, когда батарея выходит из строя (приходящее событие) и пропасть после того, как ошибка была устранена (уходящее событие).



Следующие идентификаторы(определители) находятся в переменной #OB81_EV_CLASS:

- В #16 # 39 Приходящее событие
- В #16 # 38 Уходящее событие.

Установка и сброс вспомогательного маркера памяти M 81.0 достигается через оценку этих переменных.

В циклической программе вспомогательный маркер памяти M81. 0 может быть связан с выходом. Выход подключен к индикатору, который светится, пока батарея пуста или удалена.

Обработка синхронных ошибок

- Синхронные ошибки соответствуют определенному месту в программе пользователя
- Ошибки в арифметических инструкциях (переполнение, неправильное действительное (REAL) число)
 -  Установка битов слова статуса
- Ошибки в выполнении STL инструкций (синхронные ошибки)
 -  Вызов ОВ синхронных ошибок

Тип ошибки	Пример	ОВ-обработчик
Программная ошибка	Вызванный блок не существует в CPU	ОВ 121
Ошибка доступа	Прямой доступ к неисправному или несуществующему блоку	ОВ 122

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.5Information and Training Center
Knowledge for Automation

Синхронные ошибки Операционная система CPU генерирует синхронное прерывание, когда встречается ошибка непосредственно в связи с выполнением программы. ОВ121 вызывается ошибкой программирования. ОВ122 вызывается ошибкой доступа. Если блок синхронных ошибок не загружен в CPU, то он переключается в режим STOP, если встречается синхронная ошибка. ОВ синхронных ошибок имеет тот же самый приоритет, что и блок, в котором ошибка произошла. По этой причине регистры прерванного блока могут быть доступны в ОВ синхронных ошибок и именно поэтому программа из ОВ синхронных ошибок может также возвращать регистры (в случае необходимости с измененным содержанием) прерванному блоку.

Маскировка синхронной ошибки S7 имеет следующие SFC, с помощью которых Вы можете маскировать и демаскировать вызов ОВ121 во время обработки Вашей программы:

- SFC36 (MSK_FLT): Маскирует определенные ошибки
- SFC37 (DMSK_FLT): Демаскирует ошибки, которые были маскированы SFC36
- SFC38 (READ_ERR): Читает регистр ошибок

Стартовая информация для программных ошибок в OB121

Имя переменной	Тип данных	Описание, назначение
OB121_EV_CLASS	BYTE	B#16#25=Класс события - вызов OB121
OB121_SW_FLT	BYTE	Код ошибки (смотри текст)
OB121_PRIORITY	BYTE	Класс приоритета OB, в котором произошла ошибка
OB121_OB_NUMBR	BYTE	Номер OB (B#16#79 = 121)
OB121_BLK_TYPE	BYTE	Тип блока, в котором произошла ошибка (только в S7-400) OB: B#16#88, DB: B#16#8A, FB: B#16#8E, FC: B#16#8C
OB121_RESERVED_1	BYTE	Дополнение к коду ошибки (смотри текст)
OB121_FLT_REG	WORD	OB121: источник ошибки
OB121_BLK_NUM	WORD	Номер блока, в котором произошла ошибка
OB121_PRG_ADDR	WORD	Относительный адрес команды MC7, вызвавшей ошибку (только в S7-400)
OB121_DATE_TIME	DT	Дата и время, когда был вызван OB121 (когда произошла ошибка)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.6Information and Training Center
Knowledge for AutomationКод ошибки
(#OB121_SW_FLT)

B#16#21: ошибка BCD-преобразования. Переменная #OB121_FLT_REG содержит идентификатор соответствующего регистра (W#16#0000: ACCU1).

B#16#22: Ошибка длины области для чтения

B#16#23: Ошибка длины области для записи

B#16#28: Косвенный доступ для чтения к BYTE, WORD или DWORD, битовый адрес которого не равен 0.

B#16#29: Косвенный доступ для записи к BYTE, WORD или DWORD, битовый адрес которого не равен 0. В этом случае #OB121_FLT_REG содержит ошибочный байтовый адрес и #OB121_RESERVED_1 содержит тип доступа и область памяти:

Биты с 7 по 4 -тип доступа:

0: к биту

1: к байту

2: к слову

3: к двойному слову

Биты с 3 по 0 -область памяти:

0: I/O область

1: PII

2: PIQ

3: меркеры

4: Глобальный DB

5: Экземпляр DB

6: Собств. лок. данные

7: Лок. данные

вызывающего блока

B#16#24: Ошибка области при чтении

B#16#25: Ошибка области при записи

#OB121_FLT_REG содержит в младшем байте идентификатор B#16#86: область собственных локальных данных

B#16#26: Ошибка номер таймера (неверный номер в #OB121_FLT_REG)

B#16#27: Ошибка номер счетчика (неверный номер в #OB121_FLT_REG)

B#16#30: Попытка записи в защищенный от записи глобальный DB (номер в #OB121_FLT_REG)

B#16#31: Попытка записи в защищенный от записи экземпляр DB (номер в #OB121_FLT_REG)

B#16#32: Ошибочный номер при доступе к глобальному DB (номер в #OB121_FLT_REG)

B#16#33: Ошибочный номер при доступе к экземпляру DB (номер в #OB121_FLT_REG)

B#16#34: Ошибочный номер при вызове FC (номер в #OB121_FLT_REG)

B#16#35: Ошибочный номер при вызове FB (номер в #OB121_FLT_REG)

B#16#3A: Обращение к незагруженному DB (номер в #OB121_FLT_REG)

B#16#3C: Обращение к незагруженному FC (номер в #OB121_FLT_REG)

B#16#3D: Обращение к незагруженному SFC (номер в #OB121_FLT_REG)

B#16#3E: Обращение к незагруженному FB (номер в #OB121_FLT_REG)

B#16#3F: Обращение к незагруженному SFB (номер в #OB121_FLT_REG)

Стартовая информация для ошибок доступа OB122

Имя переменной	Тип данных	Описание, назначение
OB122_EV_CLASS	BYTE	B#16#29=класс события - вызов OB122
OB122_SW_FLT	BYTE	Код ошибки (доступные значения: B#16#42, B#16#43, B#16#44, B#16#45)
OB122_PRIORITY	BYTE	Класс приоритета OB, в котором произошла ошибка
OB122_OB_NUMBR	BYTE	Номер OB (B#16#7A=122)
OB122_BLK_TYPE	BYTE	Тип прерванного блока (только в S7-400) OB: B#16#88, DB: B#16#8A, FB: B#16#8E, FC: B#16#8C
OB122_MEM_AREA	BYTE	Дополнение к коду ошибки (смотри текст)
OB122_FLT_REG	WORD	OB122: идентификатор адреса, где произошла ошибка
OB122_BLK_NUM	WORD	Номер блока, вызвавшего ошибку
OB122_PRG_ADDR	WORD	Относительный адрес команды, вызвавшей ошибку (только в S7-400)
OB122_DATE_TIME	DT	Дата и время, когда был вызван OB121 (когда произошла ошибка)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.7Information and Training Center
Knowledge for Automation**Коды ошибок**

B#16#42

Переменная #OB122_SW_FLT имеет следующие значения:

S7-300: ошибочный доступ к периферии для чтения

S7-400: ошибка при первом обращении для чтения после появления ошибки

B#16#43:

S7-300: ошибочный доступ к периферии для записи

S7-400: ошибка при первом обращении для записи после появления ошибки

B#16#44:

Только для S7-400: ошибка при n-м обращении (n>1) для чтения после появления ошибки

B#16#45:

Только для S7-400: ошибка при n-м обращении (n>1) для записи после появления ошибки

OB122_MEM_AREA

Переменная #OB122_MEM_AREA содержит информацию о типе доступа и области памяти:

Биты с 7 по 4 - тип доступа:

0: к биту

1: к байту

2: к слову

3: к двойному слову

Биты с 3 по 0 - область памяти:

0: периферия

1: PI

2: PIQ

Маскирование синхронных ошибок

Недостатки ОВ синхронных ошибок :

- Код для управления процессом и для обработки ошибок распределен по крайней мере среди двух блоков
- Проблемы с последующими изменениями или с обслуживанием

Улучшения:

- Код для управления процессом и для обработки ошибок находится в том же самом блоке

Маскирование синхронных ошибок:

- Перед "критических" инструкций:
SFC 36 MSK_FLT: маскирование синхронных ошибок (вызовов OB12x)
- Выполнение "критических" инструкций
- Оценка ошибки, если она произошла
SFC 38 READ_ERR: чтение регистра ошибок
- Разрешение вызовов OB12x снова:
SFC 37 DMSK_FLT: демаскирование синхронных ошибок

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.8Information and Training Center
Knowledge for Automation

Недостатки ОВ синхронных ошибок

Обработка синхронных ошибок посредством ОВ синхронных ошибок имеет многочисленные недостатки:

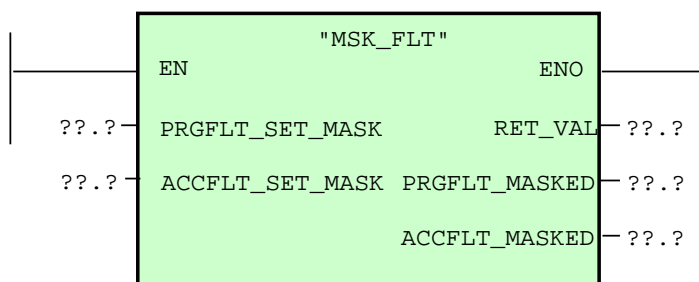
- Для квалифицированной обработки ошибок, соответствующая оценка ошибки должна быть сделана в ОВ синхронных ошибок для каждого блока, инструкции которые могут вызывать синхронную ошибку. Следовательно должна быть проведена значительная работа, чтобы локализовать ошибку в программе пользователя и затем соответственно реагировать.
- Каждое изменение в существующем блоке влечет за собой соответствующие изменения в ОВ синхронных ошибок.
- Блоки не могут быть объединены в программе пользователя без того, передачи информации в ОВ синхронных ошибок.

Альтернативы ОВ синхронных ошибок

S7 предлагает с помощью функции "Masking Synchronous Errors" механизм, который допускает, что код для управления процессом и для связанной обработки ошибок будет находиться в том же самом блоке. Это происходит, например, в следующей последовательности:

1. Перед выполнением "критических" инструкций (например открытие DB, или доступ к DB неизвестной длины), соответствующие синхронные ошибки могут быть скрыты посредством SFC 36 (MSK_FLT). Если команда затем терпит неудачу, ОВ синхронных ошибок не вызывается.
2. После выполнения "критических" инструкций Вы можете проверить посредством SFC 38 (READ_ERR), произошли или нет ошибки внутри критической секции и соответственно реагировать.
3. В заключении предварительно скрытые синхронные повреждения могут затем снова демаскированы и, таким образом, вызовы ОВ синхронных ошибок вновь разрешены.

SFC 36 для маскирования синхронных ошибок



Параметр	Объявление	Тип данных	Область памяти	Описание
PRGFLT_SET_MASK	INPUT	DWORD	I, Q, M, D, L, Const.	новый (дополнит.) фильтр программной ошибки (маскируемые ошибки программирования)
ACCFLT_SET_MASK	INPUT	BYTE	I, Q, M, D, L, Const.	новый (дополнит.) фильтр ошибки доступа (маскируемые ошибки доступа)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Возвращаемое значение SFC, W#16#0001: новый фильтр перекрывает существующий фильтр
PRGFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Полный фильтр программной ошибки (маскированные ошибки программирования)
ACCFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Полный фильтр ошибки доступа (маскированные ошибки доступа)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.9Information and Training Center
Knowledge for Automation

Маскирование синхронных ошибок С SFC 36 (MSK_FLT), Вы задерживаете вызов ОВ синхронных ошибок, используя фильтр ошибки. Устанавливая "1", Вы выделяете в фильтрах ошибки, для которых ОВ синхронных ошибок не должны быть вызваны (синхронные ошибки маскированы).

Указанное маскирование происходит дополнительно к уже замаскированным ошибкам (логическая операция ИЛИ для битов фильтра). SFC36 возвращает ошибку (W#16#0001) , если маскируемая ошибка уже замаскирована (хотя бы одна). SFC36 возвращает в выходных параметрах все в настоящее время замаскированные ошибки с помощью сигнала "1".

Реакция CPU

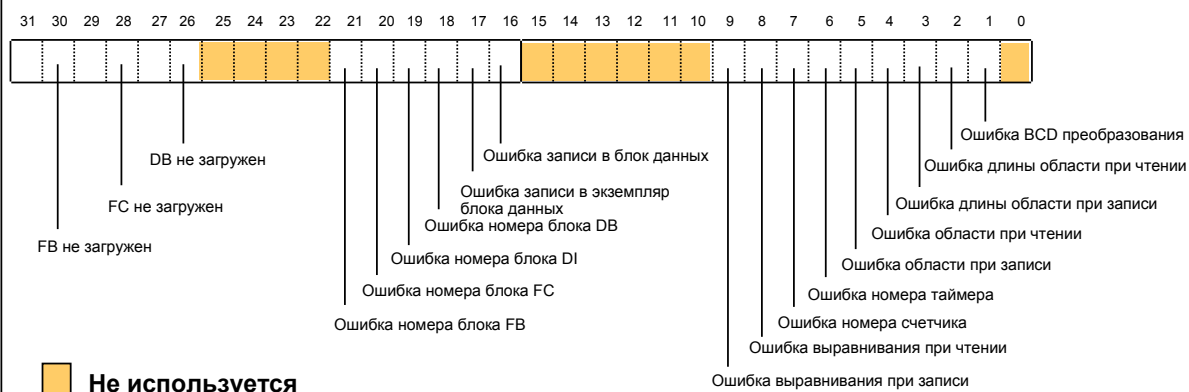
Если программная ошибка или ошибка доступа замаскирована, CPU реагирует следующим образом:

1. ОВ ошибки не вызывается при ошибке доступа или программирования.
2. Результат ошибки вводится в регистр ошибок. Регистр ошибок может быть прочитан с помощью SFC38 (READ_ERR).
3. Операционная система записывает в диагностический буфер синхронную ошибку независимо от маскирования.

Области действия маскирования

Маскирование имеет силу только для класса приоритета, в котором SFC 36 был вызван. Если Вы, например, задерживаете вызов ОВ синхронных ошибок в основной программе, ОВ синхронных ошибок будет все же вызван, если ошибка встречается в программе прерывания.

Структура фильтра ошибок программирования



Замечание: соответствующие биты выходного параметра PRGFLT_MASKED устанавливаются следующим образом:
 Значение = 1: Ошибка маскирована.
 Значение = 0: Ошибка не маскирована.

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
 File: PRO2_08E.10



Information and Training Center
 Knowledge for Automation

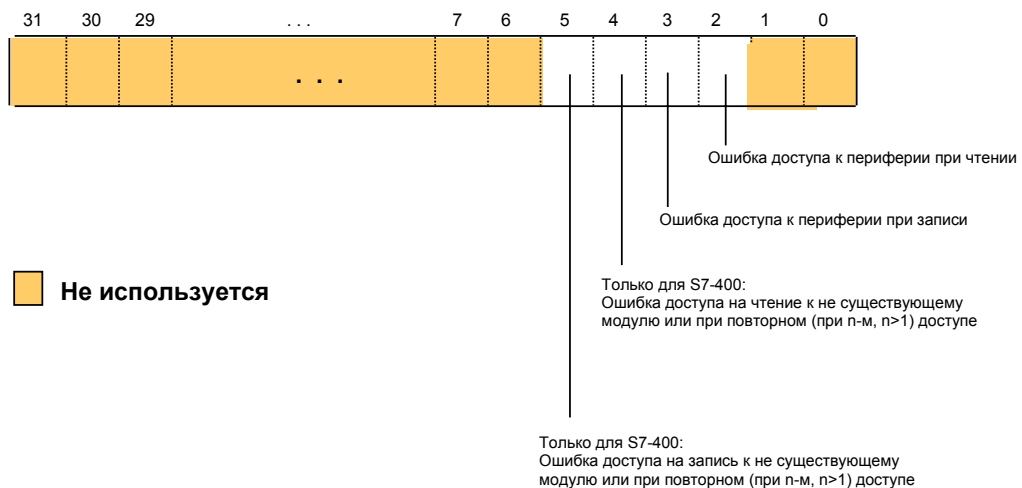
Фильтр программных ошибок

Вы управляет системной функцией для синхронной обработки ошибок с помощью фильтра ошибок.

В фильтре ошибок программирования имеется бит для каждой возможной ошибки программирования. При определении фильтра ошибки, Вы устанавливаете биты синхронных ошибок, которые Вы хотите маскировать, демаскировать или проверять.

Фильтры ошибок, переданные системными функциями, указывают значением бита "1" синхронные ошибки, которые маскированы.

Структура фильтра ошибок доступа



Замечание: соответствующие биты выходного параметра ACCFLT_MASKED устанавливаются следующим образом:

Значение = 1: Ошибка маскирована.

Значение = 0: Ошибка не маскирована.

Не используемые биты имеют значение "1".

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.11



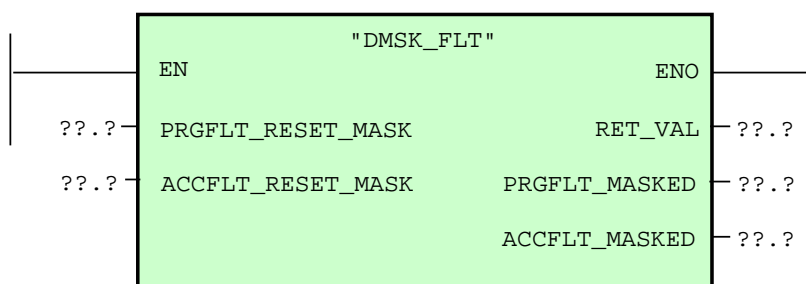
Information and Training Center
Knowledge for Automation

Фильтр ошибки доступа CPU S7-400 различает два типа ошибок доступа к периферии. Доступ к несуществующему блоку и ошибочный доступ к блоку, введенному (указанному), как существующий.

Если блок выходит из строя во время работы, тайм-аут (QVZ) наступает после небольшого времени, когда блок используется. В то же самое время этот блок вводится как "не существующий", чтобы ошибка сообщалась с каждым дополнительным доступом к периферии (PZF).

CPU также выдает сигнал ошибки доступа к периферии, когда используется не существующий блок через область периферии или косвенно через изображение процесса.

SFC 37 для демаскирования синхронных ошибок



Параметры	Объявление	Тип данных	Область памяти	Описание
PRGFLT_RESET_MASK	INPUT	DWORD	I, Q, M, D, L, Const.	Демаскируемые ошибки программирования
ACCFLT_RESET_MASK	INPUT	BYTE	I, Q, M, D, L, Const.	Демаскируемые ошибки доступа
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Возвращаемое значение SFC, W#16#0001: новый фильтр содержит биты, не установленные в сохраненном фильтре
PRGFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Все еще маскированные ошибки программирования
ACCFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Все еще маскированные ошибки доступа

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.12Information and Training Center
Knowledge for Automation

Демаскирование синхронных ошибок

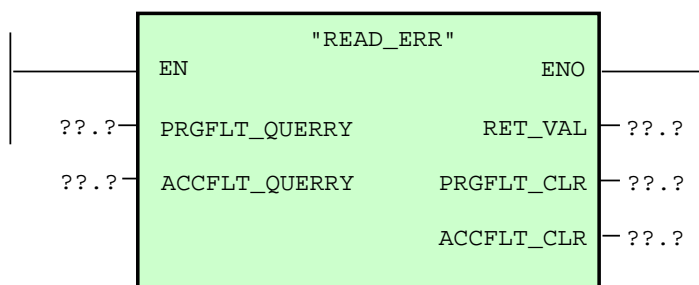
Системная функция SFC37 (DMSK_FLT) использует фильтры ошибок, чтобы разрешить вызов ОВ синхронных ошибок снова. Установкой "1", Вы выделяете в фильтре ошибки, для которых ОВ синхронных ошибок должны быть вызваны снова (синхронные ошибки - "демаскированы"). Соответствующие входы указанного демаскирования, которые находятся в регистре ошибок, удалены.

SFC37 возвращает в RET_VAL значение W#16#0001, если для указанного фильтра во входном параметре не установлен бит хотя бы для одной демаскируемой ошибки.

SFC37 возвращает в выходных параметрах все в настоящее время маскированные ошибки с помощью "1".

Если встречается демаскированная синхронная ошибка вызывается ОВ ошибки и результат вводится в регистр ошибок. Демаскирование имеет силу для текущего класса приоритета.

SFC 38 для чтения регистра ошибок



Параметр	Объявление	Тип данных	Область памяти	Описание
PRGFLT_QUERY	INPUT	DWORD	I, Q, M, D, L, Const.	Опрос ошибок программирования
ACCFLT_QUERY	INPUT	BYTE	I, Q, M, D, L, Const.	Опрос ошибок доступа
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Возвращаемое значение SFC, W#16#0001: по крайней мере одна из опрошенных ошибок не маскирована
PRGFLT_CLR	OUTPUT	DWORD	I, Q, M, D, L	Фильтр с ошибками программирования
ACCFLT_CLR	OUTPUT	DWORD	I, Q, M, D, L	Фильтр с ошибками доступа

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.13



Information and Training Center
Knowledge for Automation

Чтение регистра ошибок

Системная функция SFC38 (READ_ERR) читает регистр ошибок. Значением "1" Вы выделяете в фильтрах ошибки, которые Вы хотите читать на входах.

В возвращаемом значении RET_VAL SFC38 возвращает значение W#16#0001, если для указанного выбора в параметрах входа, не существует маскированной хотя бы одной ошибки.

SFC38 возвращает результаты с помощью "1" в выходном параметре, и удаляют эти результаты из регистре ошибок. Установленный бит означает, что связанная с ним маскированная синхронная ошибка произошла по крайней мере один раз.

Сообщаются синхронные ошибки, произошедшие в текущем классе приоритета.

Пример: тестирование блока данных

```

Network 1: Маскирование, тестирование, демаскирование
// Маска "DB не существует"
CALL SFC 36(
  PRGFLT_SET_MASK      := DW#16#4000000, // Идентификатор: DB не существует
  ACCFLT_SET_MASK      := DW#16#0,       // ошибки доступа не маскируются
  RET_VAL              := #SFC36Error,
  PRGFLT_MASKED        := #Prog36Mask,
  ACCFLT_MASKED        := #Acc36Mask);

// Тест вызова
OPN DB[DB_NO];

// Проверка ошибки программирования
CALL SFC 38(
  PRGFLT_QUERY         := DW#16#4000000, // Идентификатор: DB не существует
  ACCFLT_QUERY         := DW#16#0,       // ошибки доступа не маскируются
  RET_VAL              := #SFC38Error,
  PRGFLT_MASKED        := #Prog38Mask,
  ACCFLT_MASKED        := #Acc38Mask);

// Оценка результата
L   #Prog38Mask
L   DW#16#4000000
==D
=   #DB_NOT_THERE // Установка вспомогательной переменной, если
//DB не существует

// Демаскирование ошибки "DB не существует"
CALL SFC 37(
  PRGFLT_RESET_MASK    := DW#16#4000000, // Идентификатор: DB не существует
  ACCFLT_RESET_MASK    := DW#16#0,       // ошибки доступа не маскируются
  RET_VAL              := #SFC37Error,
  PRGFLT_MASKED        := #Prog37Mask,
  ACCFLT_MASKED        := #Acc37Mask);

```

SIMATIC S7

Siemens AG 1999. All rights reserved.

 Date: 04.11.2005
 File: PRO2_08E.14

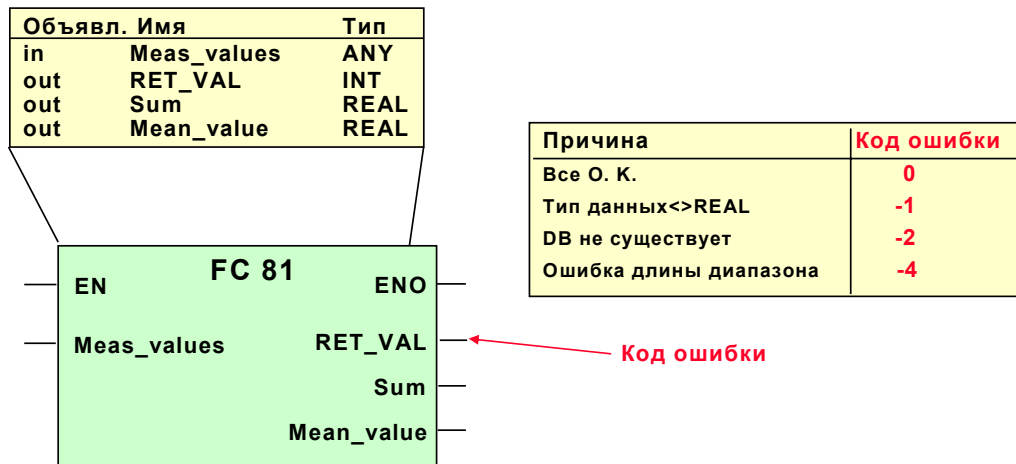
 Information and Training Center
 Knowledge for Automation

Пример

Этот пример показывает процедуру для маскирования возможной синхронной ошибки при открытии DB.

1. На 1-ом шаге "критическая" команда OPN DB ... маскирована с помощью SFC 36 (MSK_FLT).
2. После того, как это сделано, выполняется команда OPN DB [DB_NO]. Если DB нет в рабочей памяти CPU, OB121 в этом случае не вызывается.
3. С помощью SFC38 (READ_ERR), читается регистр ошибок и проверяется, выполнялась ли команда открытия DB или нет. В случае ошибки, локальная переменная #DB_NOT_THERE устанавливается в "1", чтобы позже можно было сделать оценку.
4. В конце маскированная синхронная ошибка - демаскируется с помощью SFC37(DMSK_FLT), таким образом восстанавливается первоначальное состояние.

Упражнение 8.1: обработка ошибок в FC81



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_08E.15



Information and Training Center
Knowledge for Automation

Краткий обзор

В примере 4.3 Вы создали FC43, которая определяет сумму и среднюю величину из массива действительных чисел. Пока она имеет только элементарную обработку ошибок (контроль типа данных), которая производится внутри этой FC.

Обработка ошибок должна теперь быть расширена таким образом, чтобы новый FC81 являлся "крахо-безопасным", то есть даже при неправильно назначенном параметре, не вызывается никакой обработчик синхронной ошибки.

Кроме того, FC81 возвращает в дополнительном выходном параметре #RET_VAL информацию относительно типа ошибки.

Цель

Сначала FC43 копируется в FC81 и интегрирует следующие ошибки слежения:

- если тип данных не является REAL, то FC81 завершает работу с кодом ошибки -1.
- если указан недействительный номер DB (например, номер вне допустимого диапазона или DB не существует), то FC81 выходит с кодом ошибки -2.
- если внутри цикла имеется доступ к не существующему адресу (диапазон или ошибка длины диапазона), то FC81 выходит с кодом ошибки -4.
- Во всех случаях ошибки FC81 устанавливает BR-бит в нуль и возвращает недействительное число REAL в выходных параметрах #Сумма и #Mean_Value

Что делать

1. Добавить в FC81 выходной параметр # RET_VAL (код ошибки).
2. В FC81 сделать соответствующую обработку ошибок.
3. FC81 вызывается в OB1.
4. Загрузить участвующие блоки в CPU и проверить результат.

Вопрос?

Как Вы можете привести FC81 к "краху"?

Создание программы в текстовом редакторе



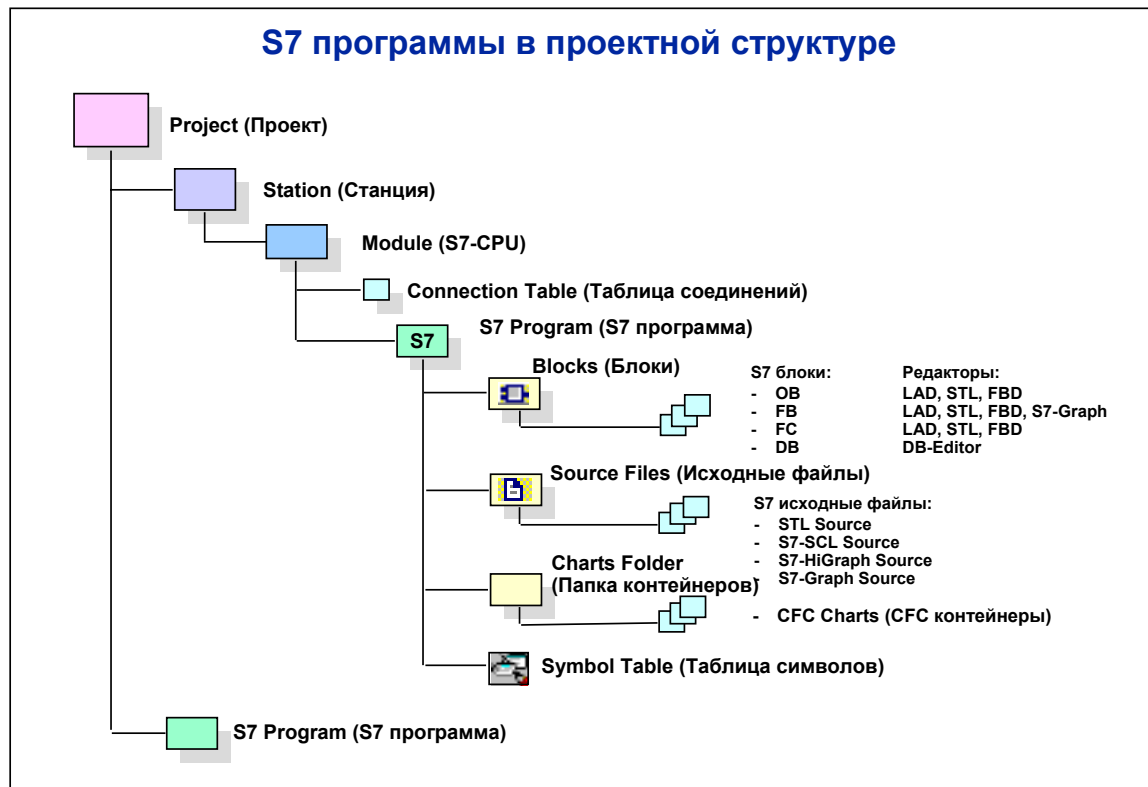
SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_09E.1Information and Training Center
Knowledge for Automation

Содержание	Стр
S7 программы в проектной структуре	2
Концепция ввода и компиляции	3
Запуск текстового редактора	4
Генерация программы из текстового редактора	5
Вставка шаблонов блока, блоков и исходных файлов	6
Общие правила ввода и структура	7
Синтаксис логического блока	8
Синтаксис блока данных	9
Правила для объявления переменных	10
Назначение атрибутов блока	11
Упражнение 9.1: Создание исходного файла	12
Упражнение 9.2: Подсчет готовых деталей	13

S7 программы в проектной структуре



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_09E.2



Information and Training Center
Knowledge for Automation

Краткий обзор

Чтобы можно было создать новую S7 программу, проект должен быть сначала создан в SIMATIC Manager. Далее имеются две возможности для установки папки S7 программы:

- Независимый блок: В этом случае Вы должны вставить папку программы для S7-программ непосредственно в корень проекта. Программы, созданные там могут позже быть назначены программируемому модулю.
- Зависимый блок: В этом случае проект должен содержать по крайней мере одну SIMATIC станцию 300/400 с программируемым модулем (CPU). Папка S7- программ затем автоматически вставляется ниже программируемого модуля.

Если Вы хотите использовать глобальные символы в вашей пользовательской программе, Вы должны сделать заранее соответствующее назначение идентификаторов и истинных адресов в таблице символов.

Блоки, источники и карты

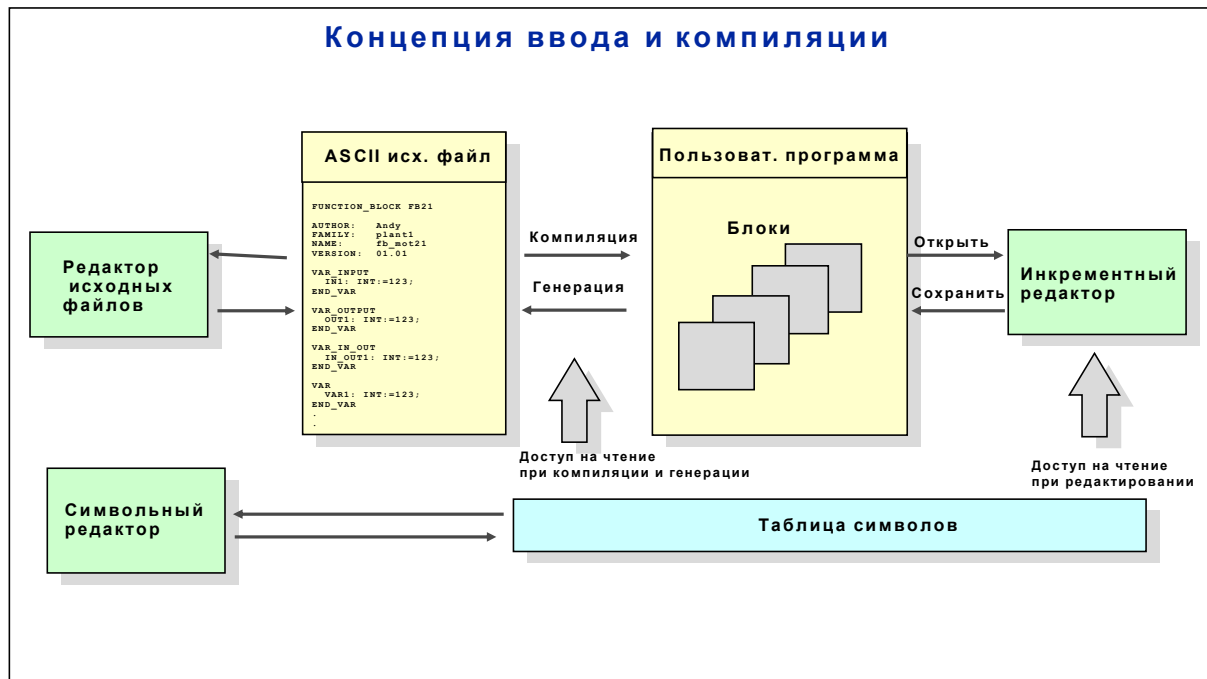
Вы можете хранить S7 программу как программу пользователя (блок), исходный файл или карту.

Исходные файлы и карты используются только, как основа для производства блоков. Только блоки могут быть разгружены в S7-CPU. Генерируете ли Вы блок, исходный файл или карту, зависит от избранного языка программирования или редактора языка.

Программа пользователя

Только блоки программы пользователя могут быть загружены в S7-CPU. В зависимости от области действия, это включает организационные блоки (OB), функции (FC), функциональные блоки (FB) и блоки данных (DB). Созданные определяемые пользователем типы данных (UDT) просто упрощают программирование, они не могут быть разгружены в S7-CPU. Тот же самое имеет силу для таблиц переменных (VAT), в которых сохраняются адреса для функции *Monitor/Modify Variables*.

Концепция ввода и компиляции



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PROJ_09E.3Information and Training Center
Knowledge for Automation

Возможности ввода В зависимости от языка программирования, который Вы выбрали для создания программы, Вы можете вводить вашу программу в инкрементном или ориентированном на исходные тексты редакторе.

- Ввод в инкрементных редакторах (STL, LAD, FBD, S7-Graph, S7-HiGraph, CFC).

Каждая строка или каждый элемент(компонент) немедленно исследуется на синтаксические ошибки после ввода. Существующие ошибки ввода обозначаются (выделяются красным) и должны быть исправлены перед сохранением.

Синтаксически правильный ввод автоматически компилируется и показывается черным цветом. При инкрементном вводе используемые символы должны уже быть определены в таблице идентификаторов, иначе ввод регистрируется красным, а соответствующее сообщение об ошибках показано в строке состояния.

- Исходно - ориентированный ввод (STL, S7-SCL).

При исходно - ориентированном вводе программа или блок редактируется в текстовом файле, текстовый файл затем компилируется, при этом ошибки определяются и индицируются компилятором.

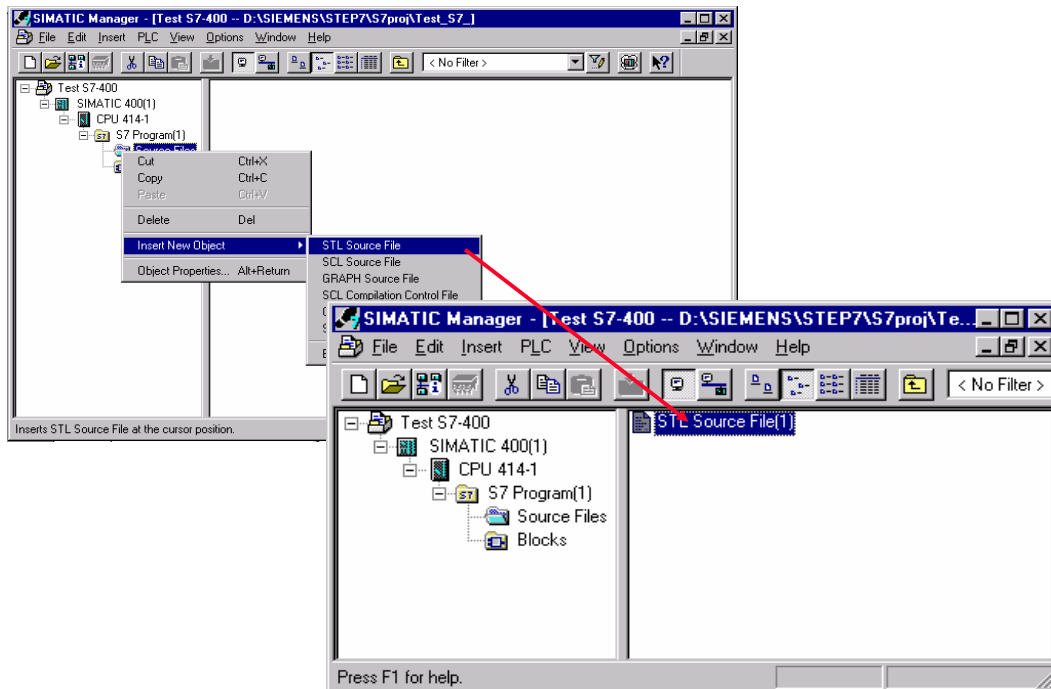
При исходно - ориентированном вводе символы должны быть определены в таблице символов во время составления. Исходные файлы имеют преимущество, что они могут экспортироваться - затем обрабатываться выбранным инструментальным средством - и затем снова импортированы.

Преимущества исходно

-ориентированного ввода

- Несколько блоков могут быть сохранены в исходном файле (блоки - должны располагаться таким образом, что вызываемые блоки всегда располагаются перед вызывающими блоками).
- Исходный файл может быть сохранен с синтаксическими ошибками.
- Вы можете создавать ваши исходные файлы другими редакторами, импортировать их в SIMATIC Manager и затем составлять из них блоки.
- Защита блока может только быть введена в режиме ASCII.
- Изменения (напр., дополнение параметров блока) во вложенных вызовах блока могут быть лучше обработаны в редакторе ASCII, чем в инкрементном редакторе.

Запуск текстового редактора



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_09E.4



Information and Training Center
Knowledge for Automation

Запуск из SIMATIC Manager

Вы запускаете текстовый редактор из SIMATIC Manager. Предпосылка - то, что Вы установили там проект с S7-Program. Вы можете создавать зависимые или независимые от оборудования программы. В текстовом редакторе Вы обрабатываете исключительно исходные файлы, из которых Вы впоследствии генерируете загружаемые блоки, которые сохраняются в папке Blocks.

Создание исходного файла

Когда Вы генерируете новый исходный файл впервые, Вы должны сначала создавать пустой файл в SIMATIC Manager, через который Вы открываете текстовый редактор. Когда Вы открыли редактор, Вы можете создавать дальнейшие исходные файлы там.

- В SIMATIC Manager выбирают папку исходных файлов и вставляют файл с помощью меню *Insert New Object -> STL Source File*. Новый исходный файл появляется в правой половине проектного окна с предварительно заданным именем.
- Непосредственно в текстовом редакторе Вы можете просто создавать новый файл, используя опцию меню *File -> New*. В диалоге дополнительного сообщения Вы вводите имя нового исходного файла.

Открытие исходного файла

Вы открываете исходный файл в SIMATIC Manager с помощью двойного щелчка на символе. В качестве альтернативы, Вы можете достигнуть этого используя опцию меню *Edit -> Open Object* или соответствующую иконку на инструментальной панели.

Генерация исходного файла

Можно также преобразовать уже существующие блоки назад в исходный файл. В текстовом редакторе выбирают для этого опцию меню *File -> Generate Source*. В диалоге дополнительного сообщения Вы можете затем выбирать все блоки, из которых Вы хотите генерировать исходный файл.

Генерация программы из текстового редактора

```

LAD/STL/FBD - [EXERCISE_5.1 -- Pro2_e_solution\CHAP_05]
File Edit Insert PLC Debug View Options Window Help

// Address assignment in FC51 is adopted to S7-300 16 bit

FUNCTION FC 51 : VOID
TITLE = Exercise 5.1: Read System Clock
//Version for 16Bit-SM
AUTHOR : PT41
FAMILY : A2_0
NAME : ST7PRO2
VERSION : 0.0

VAR_TEMP
Date_Time : DATE_AND_TIME ;           //Current Time and Date
RET_VAL_SFC1 : INT ;                  //Return value of SFC 1
END_VAR
BEGIN
NETWORK
TITLE =Call SFC 1 (READ_CLK)

CALL SFC1 (
RET_VAL      := #RET_VAL_SFC1,
CDT          := #Date_Time);

NOP 0;
NETWORK
TITLE =Display hours and minutes

LAR1 P##Date_Time;                    // Get address of #Date_Time
L LB [AR1, P#3.0];                    // Read hours
T QB 12;                              // and transfer to diq. display

```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_09E.5



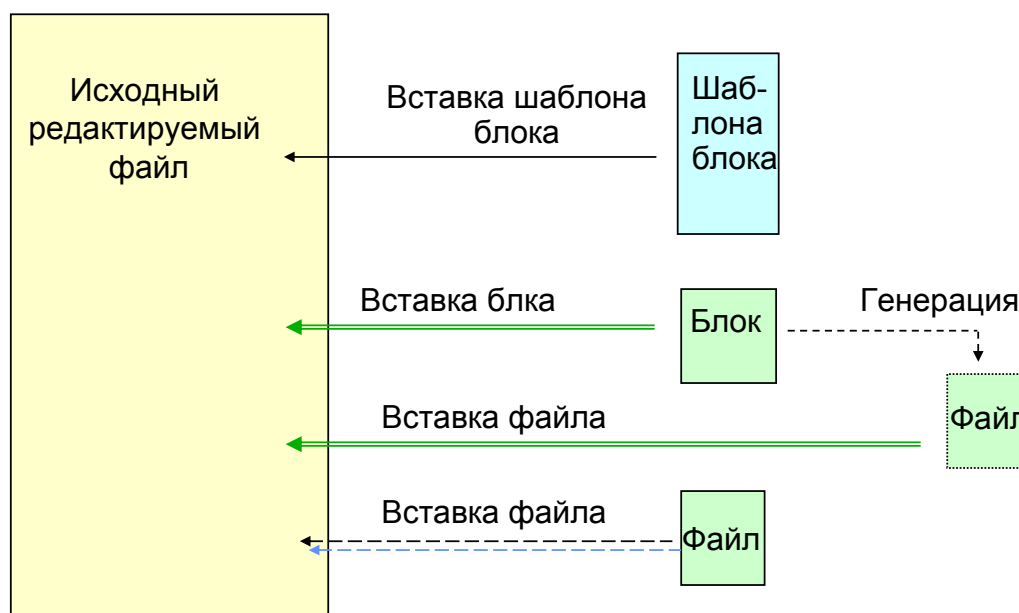
Information and Training Center
Knowledge for Automation

Текстовый редактор Вместо программирования в STL, Вы можете создавать вашу программу с помощью текстового редактора. Вы вводите ваши блоки один после другого (возможны отдельные блоки в одном исходном файле). Контроль синтаксиса не происходит.

Установки

Прежде, чем Вы начинаете программировать в текстовом редакторе, Вы должны ознакомиться с возможностями настройки, чтобы Вы могли работать удобно и согласно вашим персональным предпочтениям. Вы открываете окно диалога, используя опцию *Options ->Customize*. В закладке "Source Files" Вы можете сделать установки для компиляции исходного файла и установки для запоминания блоков.

Вставка шаблонов блока, блоков и исходных файлов



Вставка шаблона блока

Шаблоны для OB, FB, FC, DB, экземпляра DB, DB со структурой UDT и UDT интегрированы в редактор для более простого программирования. Шаблон блока содержит все требуемые ключевые слова в необходимой последовательности. Вы просто удаляете шаблоны для объявлений, которые Вам не нужны.

Шаблоны блоков облегчают ввод программы с правильным синтаксисом. Чтобы вставить блочный шаблон в ваш исходный файл, выберите опцию *Insert -> Block Template -> OB/FB/FC/DB/IDB/ DB Referencing UDT/UDT*.

Вставка блоков

Вы можете вставлять в ваш исходный файл соответствующий исходный текст блоков, которые уже были сгенерированы. Для этого выберите опцию *Insert -> Object -> Block*. В диалоге дополнительного сообщения выбирают блоки, код которых Вы желаете вставить как текст.

Исходный файл генерируется из выбранных блоков. Их содержание вставляется после позиции курсора в исходном редактируемом файле.

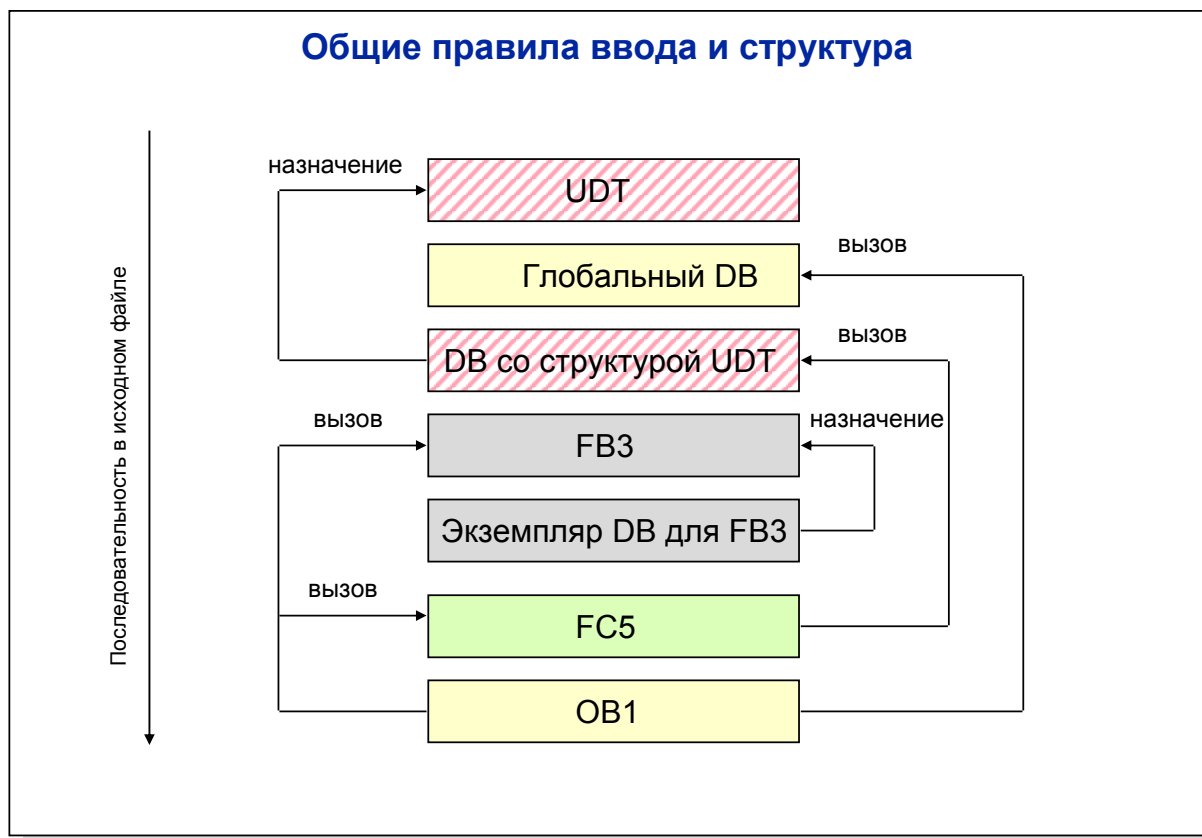
Вставка исходного файла

Вы можете вставлять содержание любых других исходных файлов в ваш исходный файл. Для этого выберите опцию меню *Insert -> Object -> File* и в диалоге дополнительного сообщения выберите файл, который будет вставлен. Таким образом, содержание любого текстового файла может быть вставлено в ваш исходный файл.

Обратите внимание

Любой текст может также быть вставлен в ваш исходный файл, используя буфер обмена Windows.

Общие правила ввода и структура



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_09E.7Information and Training Center
Knowledge for Automation

Правила ввода



Для создания программ в исходном файле следует соблюдать следующие общие правила:

- синтаксис STL команд - такой же, как в инкрементном STL редакторе. Исключение существует для обращения к блокам и объявлений массивов и структур.
- Текстовый редактор не различает верхний и нижний регистры. Исключение - метки перехода.
- Конец каждой STL команды и каждого объявления переменной идентифицируется точкой с запятой (;). Вы можете вводить больше, чем одну команду на строку.
- Каждый комментарий начинается с двух диагональных штрихов (//) и продолжается до конца строки.

Последовательность блоков

- OB1, который используется наиболее часто и который вызывает другие блоки, является последним в файле. Блоки, которые вызываются блоками, которые вызываются из OB1, должны находиться перед ними и т.д.
- Определяемые пользователем типы данных (UDT) находятся перед блоками, в которых они используются.
- Блоки данных со структурой определяемой пользовательскими типами данных (UDT), находятся после UDT.
- Глобальные блоки данных находятся перед всеми блоками, из которых они вызываются.
- Блоки данных, назначенные функциональным блокам (экземпляры DB) находятся после функционального блока.

Синтаксис логического блока

Конфигурация	Ключевые слова с примерами
Начало блока со спецификациями (абсолютное или символическое)	ORGANIZATION_BLOCK OB1 FUNCTION_BLOCK FB1 FUNCTION FC 1 : int
Заголовок блока (по выбору)	TITLE = Block title
Комментарий блока (по выбору)	// Block comment
Системные атрибуты блока (по выбору)	{Attr1 := 'block_val1'; // Атрибут блока 1 Attr2 := 'block_val2'; // Атрибут блока 2 Attr3 := 'block_val3' // Атрибут блока 3}
Свойства блока (по выбору)	KNOW_HOW_PROTECT AUTHOR: PT41 FAMILY: Motors NAME: Motorone VERSION: 0815
Раздел объявления переменных (Объявление типа, в зависимости от типа блока)	VAR_IN VAR_OUT VAR_IN_OUT VAR VAR_TEMP .. END_VAR
Часть инструкций, состоящая из сетей с заголовком сети и комментарием сети	BEGIN NETWORK TITLE=first network //
Конец блока	END_ORGANIZATION_BLOCK END_FUNCTION_BLOCK END_FUNCTION

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_09E.8Information and Training Center
Knowledge for Automation

Правила

При создании логического блока Вы должны обратить внимание на следующие правила:

- В начале блока имеется пробел между ключевым словом для типа блока и спецификацией блока. При определении символического имени блока Вы можете идентифицировать его в кавычках, чтобы гарантировать различие между именами локальных переменных и имен таблицы идентификаторов.
- Функции (FC), имеют тип. Это может быть элементарный или сложный тип данных, он определяет тип данных возвращаемого значения (# RET_VAL). Если никакое значение не должно быть возвращено, тип функции должен быть объявлен как VOID.
- спецификация сетевого номера не допускается.

Вызовы блока с "CALL"

Синтаксис для вызовов FB и FC командой CALL, используемый в инкрементном STL редакторе, здесь не используется. В исходном файле Вы вводите параметры в скобках. Индивидуальные параметры затем отделяются от друга друга запятыми.

Пример: CALL FC1 (param1: = I 0.0, param2: = I 0.1);

Комментарии в часть команд

Чтобы гарантировать 1:1 представление комментариев в инкрементном редакторе, Вы должны обратить внимание на следующее:

- Вызов блоков: когда Вы назначаете фактические параметры для формальных параметров в исходных файлах, Вы должны сохранить последовательность формальных параметров, как они находятся в объявлении переменных блока. Хотя последовательность параметров имеет выбор, комментарии к параметрам могут однако быть включены в течение трансляции источника в блоки.
- У команды, которая непосредственно следует за командой для доступа к блокам данных "OPN", возможна потеря комментариев в течение трансляции в блоки. Чтобы избежать этого, программируют в компактной форме (например L DB5. DBW20; // Коммент.) или запись команды "NOP" (например, OPN DB5; // Коммент.1 NOP 0; L DBW20; // Коммент.2).

Синтаксис блока данных

Конфигурация	Ключевые слова с примерами
Начало блока со спецификациями (абсолютное или символическое)	DATA_BLOCK DB 26
Заголовок блока (по выбору)	TITLE = <i>Block title</i>
Комментарий блока (по выбору)	// <i>Block comment</i>
Системные атрибуты блока (по выбору)	{Attr1 := 'block_val1'; // Атрибут блока 1 Attr2 := 'block_val2'; // Атрибут блока 2}
Свойства блока (по выбору)	KNOW_HOW_PROTECT AUTHOR: <i>Müller</i> FAMILY: <i>Motors</i> NAME: <i>Motorone</i> VERSION: <i>0815</i>
Часть объявлений - в зависимости от DB	
Глобальный блок данных: объявление переменных (по выбору с инициализирующими значениями)	STRUCT END_STRUCT
DB со структурой UDT: спецификация UDT (абсолютная или символическая)	UDT 16
Экземпляр DB: спецификация FB (абсолютная или символическая)	FB 20
Часть назначения с текущими (актуальными) значениями	BEGIN ..
Конец блока	END_DATA_BLOCK

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_09E.9Information and Training Center
Knowledge for Automation

Правила

В создаваемых блоках данных Вы должны обратить внимание на следующие правила:

- Вы не можете генерировать DB 0.
- Необязательно определять текущие (актуальные) значения для всех или некоторых переменных. Для переменных, которым Вы не назначаете текущие (актуальные) значения, назначается начальное значение - если доступно -, иначе назначается значение по умолчанию для типа данных.
- Комментарии в разделе назначения для текущих (актуальных) значений (между BEGIN, и END_DATA_BLOCK) не отображаются в инкрементном редакторе после трансляции в блоки. Следовательно, пишите комментарии для блоков только в разделе описания данных.

Правила для объявления переменных

Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Actual_value	WORD	W#16#0	Actual value
2.0	in	Setpoint	WORD	W#16#0	Setpoint
4.0	in	Tracking	WORD	W#16#0	if used as master controller
6.0	in	Control_word	WORD	W#16#0	Seperate Input
8.0	in	Disturbance_var	WORD	W#16#0	Input for disturbance variak
10.0	out	Control_value	WORD	W#16#0	Control Value

```

FUNCTION_BLOCK FB 10
TITLE =
AUTHOR : Birdy
NAME : Control
VERSION : 0.1

VAR_INPUT
    Actual_value : WORD ; //Actual value
    Setpoint : WORD ; //Setpoint
    Tracking : WORD ; //if used as master controller
    Control_word : WORD ; //Seperate Input
    Disturbance_var : WORD ; //Input for disturbance variable
END_VAR

VAR_OUTPUT
    Control_value : WORD ; //Control Value

```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_09E.10Information and Training Center
Knowledge for Automation

Типы переменной

В логических блоках тип объявления переменной идентифицирован ключевым словом, которое находится в собственной строке. В зависимости от типа блока допускаются только специфические типы объявления.

Объявление переменных	Ключевое слово	ОВ	FB	FC
Входн. параметры	VAR_INPUT	-	да	да
Выходн. параметры	VAR_OUTPUT	-	да	да
Проходные параметры (in/out)	VAR_IN_OUT	-	да	да
Статические переменные	VAR	-	да	-
Временные переменные	VAR_TEMP	да	да	да
Каждый раздел завершается с	END_VAR			

Правила ввода

При вводе объявления переменных Вы должны обратить внимание на следующее:

- типы переменных должны быть объявлены в последовательности, указанной выше. Все переменные одного типа объявляются вместе.
- ключевые слова записываются в каждом случае в собственной строке или отделяются пробелом.
- имя переменное вводится в начале строки и должно начинаться с символа. Оно не может совпадать с любым из зарезервированных ключевых слов.
- Факультативные системные атрибуты могут быть назначены для индивидуальных параметров после имени переменного. Системные атрибуты заключены в скобки - { } .
Пример: Var_1 {ident1: = 'string1'; ident2: = 'string2'}: INT;
Var_2 {message : = 'TRUE' ; OPERABLE : = 'TRUE' } : INT;
- тип данных отделяется двоеточием от имени или после системного атрибута. Допускаются элементарные, сложные и определяемые пользователем типы данных.
- Каждое объявление переменной заканчивается точкой с запятой.
- Комментарии отделяются от раздела описаний двумя диагональными штрихами.

Назначение атрибутов блока

Атрибут	Логические блоки (OB, FB, FC)	Блоки данных	UDT
KNOW_HOW_PROTECT	да	да	нет
AUTHOR	да	да	нет
FAMILY	да	да	нет
NAME	да	да	нет
VERSION	да	да	нет
UNLINKED	нет	да	нет
READ_ONLY	нет	да	нет

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_09E.11



Information and Training Center
Knowledge for Automation

Системные атрибуты Вы можете назначать системные атрибуты блокам, например, для диагностики процесса или конфигурации системы управления. Они управляют конфигурацией сообщения и конфигурацией соединения, функциями интерфейса оператора и конфигурацией системы управления.

Реквизиты блока Вы может определять для блока имя, семейство, версию и автора с помощью ключевых слов. Для этого допустимо следующее:

- Реквизиты блока определяются перед разделом описаний переменных.
- В конце строки не ставится (!) точка с запятой.

Защита блока Вы может устанавливать защиту для логических блоков и блоков данных, определяя ключевое слово KNOW_HOW_PROTECT:

- Когда Вы открываете откомпилированный блок в инкрементном STL редакторе, то раздел команд блока не будет видна.
- Отображаются только параметры (in, out and in/out) в разделе описаний переменных блока.

Внутренние переменные VAR и VAR_TEMP остаются скрытыми.

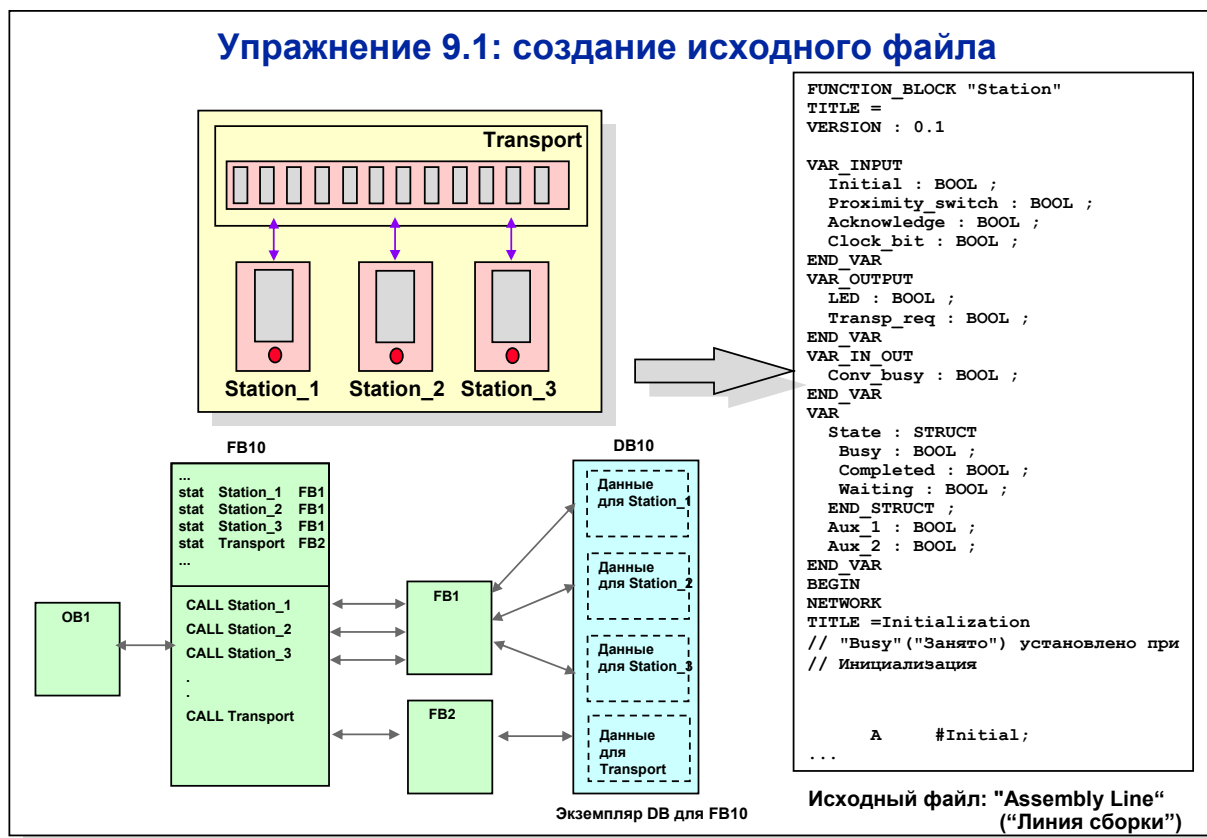
- Компилируемый блок может компилироваться в исходный файл, но только как блок без раздела команд.

Ключевое слово KNOW_HOW_PROTECT должно быть введено перед всеми другими атрибутами блока.

Защита от записи READ_ONLY Вы может устанавливать защиту от записи для блоков данных в исходных файлах, так, чтобы значения данных, сохраненные там не могли быть переписаны поверх во время выполнения программы. Для этого введите ключевое слово READ_ONLY. Оно должно быть помещено в собственной строке непосредственно перед объявлениями.

UNLINKED (нелинкующий) Атрибут UNLINKED применяется только для блоков данных. Он говорит, что DB не должен быть загружен из загрузочной памяти в рабочую память CPU.

Упражнение 9.1: создание исходного файла



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_09E.12Information and Training Center
Knowledge for Automation

Краткий обзор

Прежде всего исходный файл должен быть создан из законченной программы упражнения 6.2 "Assembly line" ("Линии сборки"). Впоследствии должны быть представлены дополнительные функциональные возможности счета в блоке для транспортировки деталей.

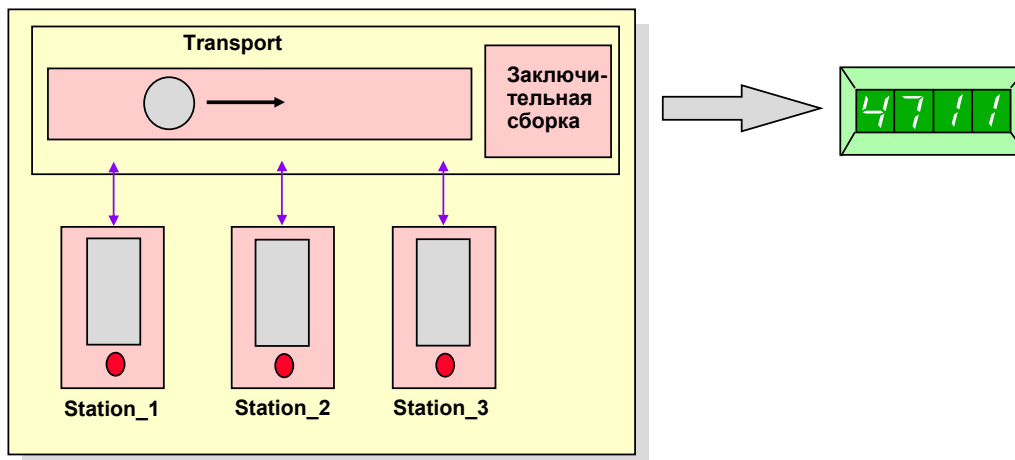
Цель упражнения

В папке *Source Files* PRO2-PROJECT (папка программы Conv), сгенерируйте исходный файл, который включает всю программу пользователя упражнения 6.2 и он должен компилироваться без сообщений об ошибках.

Процедура

1. Прежде всего откройте в папке *Blocks* папки программы Conv выбранный блок с помощью STL/LAD/FBD редактора.
2. Затем выберите опцию File -> Generate Source. Появляется диалог "New" для ввода желаемого имени исходного файла.
3. Введите имя исходного файла (например линия сборки) и подтвердите диалог кнопкой "OK". Появляется дополнительный диалог "Generate Source File STL...".
4. Выберите желаемый блок и подтвердите "OK".
Обратите внимание: Вы можете выбрать флажок "Sort according to program structure". Тем самым блоки в исходном файле автоматически размещаются в правильной последовательности.
Генерация исходного файла начата.
5. Текстовым редактором откройте созданный исходный файл.
6. С помощью опции File -> Check Consistency проверьте, может ли исходный файл компилироваться без ошибок.

Упражнение 9.2: Подсчет готовых деталей



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_09E.13



Information and Training Center
Knowledge for Automation

Цель упражнения

В функциональный блок "Transport", интегрируйте счетчик, который считает готовые детали, приходящие на заключительную сборке. Счетчик должны включить следующие функциональные возможности:

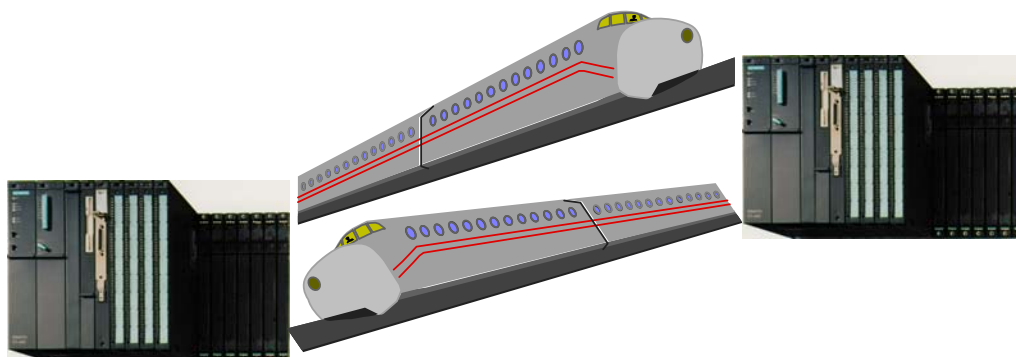
- Счетчик должен быть выполнен с помощью IEC-счетчика (IEC 1131-3) для счета вперед (SFB 0 "CTU").
- С каждым отрицательным фронтом фотоэлемента счетчик увеличивает *#Transport_right*.
- Счетчик сбрасывается входным сигналом *#Initial*.
- Текущий (актуальный) счет передается вызывающему блоку через дополнительный выходной параметр *#Count_Value* (тип данных: INT).
- Значение счета отображается на цифровом индикаторе имитатора.
- Выполните все шаги программы исключительно в исходном файле.
- Вставьте защиту блоков во все FB и DB, используя ключевое слово *KNOW_HOW_PROTECT*.

Что делать

1. Скопируйте SFB 0 из библиотеки Standard Library V3.x или FB6 в вашу папку блоков.
2. Откройте исходный файл Assembly line.
3. В FB2 "Transport", объявите статическую переменную *#Counter* типа SFB 0 или FB6 также как выходной параметр *#Count_Value* типа INT.
4. Вставьте необходимые команды для функции счета в FB2 "Transport".
5. В FB10 вставьте команды для отображения значения счета (в BCD коде) на цифровом индикаторе.
6. Скомпилируйте измененный исходный файл и загрузите новые блоки в CPU. Проверьте программу.
7. Вставьте защиту блоков во все участвующие FB и DB.



Базовые и расширенные S7- коммуникации



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.1



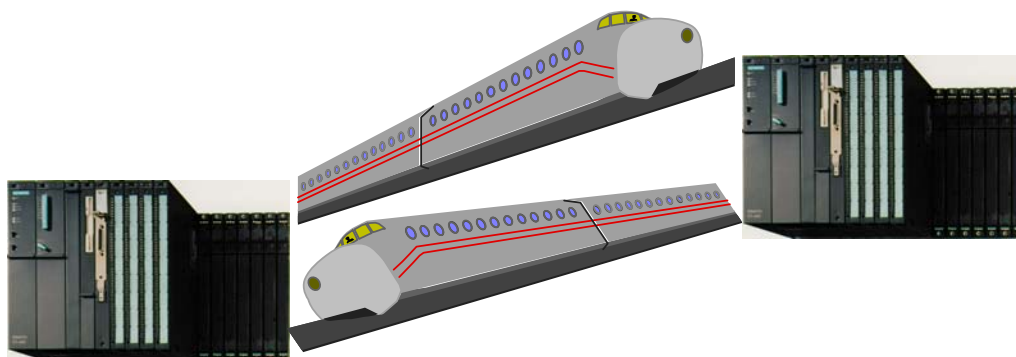
Information and Training Center
Knowledge for Automation

Содержание

Стр

Подсети в SIMATIC	3
Коммуникационные службы для SIMATIC	4
S7 коммуникационные службы для S7-300/400	5
Соединения между коммуникационными участниками	6
Назначение ресурсов соединений для S7 коммуникаций	7
Характеристики связи S7-CPU	8
SFC коммуникации: обзор	9
SFC коммуникации: обзор блоков	10
SFC коммуникации: блок X_GET (SFC 67)	11
SFC коммуникации: блок X_PUT (SFC 68)	12
SFC коммуникации: блок X_SEND (SFC 65)	13
SFC коммуникации: блок X_RCV (SFC 66)	14
SFB коммуникации: обзор	15
SFB коммуникации: обзор блоков	16
Односторонние коммуникационные службы, использующие S7 соединения	17
Двусторонние коммуникационные службы, использующие S7 соединения	18
Конфигурация сети с помощью NETPRO	19
Конфигурирование S7 соединений	20
Установка свойств соединения	21
Компилирование и загрузка данных конфигурации	22
SFB коммуникации: блок GET (SFB 14)	23
SFB коммуникации: блок PUT (SFB 15)	24
SFB коммуникации: блок USEND (SFB 8)	25

Базовые и расширенные S7- коммуникации



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.2

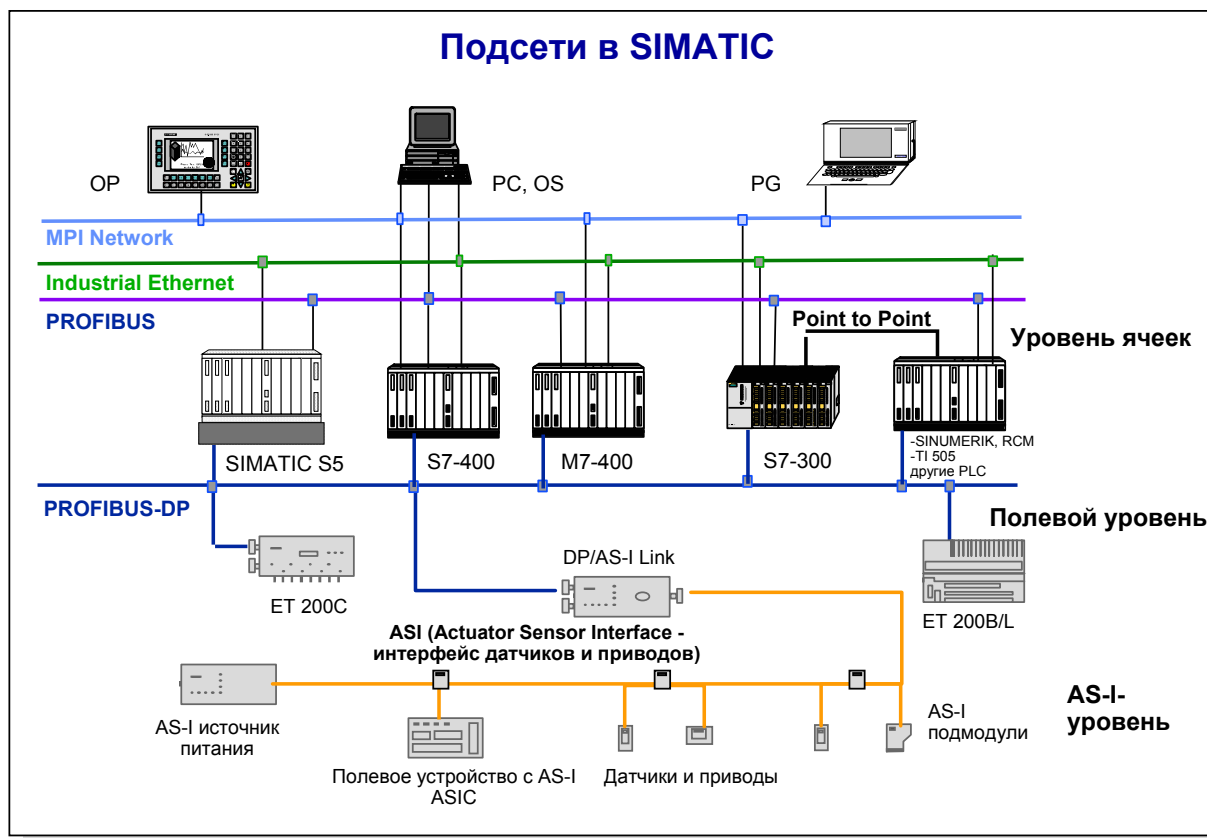


Information and Training Center
Knowledge for Automation

Содержание

Стр

SFB коммуникации: блок URCV (SFB 9)	26
SFB коммуникации: блок BSEND (SFB 12)	27
SFB коммуникации: блок BRCV (SFB 13)	28
SFB коммуникации: блок STOP (SFB20)	29
SFB коммуникации: блок START (SFB19)	30
SFB коммуникации: блок CONTROL (SFC 62)	31
Упражнение 10.1: Конфигурирование S7 соединений	32
Упражнение 10.2: Коммуникации с SFB GET/PUT	33
Упражнение 10.3: Коммуникации с SFB START/STOP	34

**SIMATIC S7**

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.3Information and Training Center
Knowledge for Automation**Краткий обзор**

SIEMENS предлагает следующие подсети, в зависимости от различных требований для задач связи на уровне ячеек (не критических ко времени) или на полевом уровне (критических ко времени).

MPI

MPI подсеть разработана для задач связи на уровне ячеек. MPI - многоточечный интерфейс в SIMATIC S7.

Он разработан как интерфейс с PG, т.е. для соединения PG (ввод в эксплуатацию и тестирование), и OP (интерфейс оператора). Кроме того, MPI подсеть может также использоваться для соединения нескольких CPU.

Industrial Ethernet

Industrial Ethernet - открытая, независимая от производителя система связи SIMATIC - сеть для уровня управления и уровня ячеек.

Industrial Ethernet разработана для не критической ко времени передачи больших количеств данных и предлагает возможность соединения локальных сетей через маршрутизаторы.

PROFIBUS

PROFIBUS - открытая, независимая от производителя система связи SIMATIC - сеть для полевого уровня и уровня ячеек. Имеются два версии, каждая с собственными характеристиками:

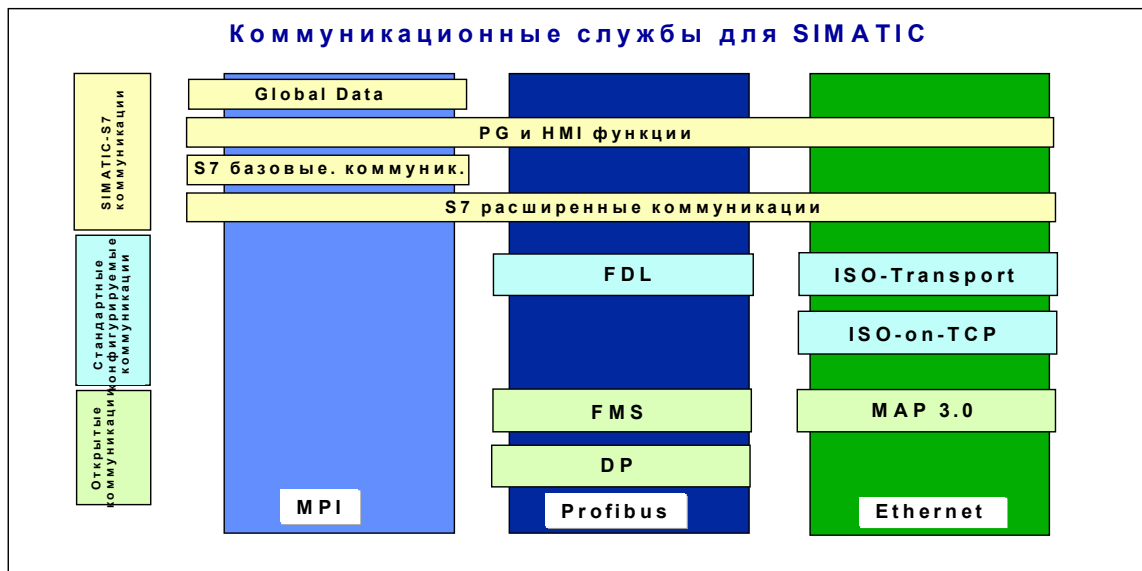
- на уровне ячеек (элементов) - PROFIBUS для не критической ко времени связи между равными, интеллектуальными узлами.
- как полевая шина PROFIBUS DP для критического ко времени, циклического обмена данными между интеллектуальными мастерами и полевыми устройствами.

PtP-соединение

Соединение точка к точке (Point-to-Point, PtP) в основном используется для не критического ко времени обмена данными между двумя устройствами или для соединения устройств, типа OP, принтеров, считывателей штрихового кода, магнитные устройства считывания с карты и т.д. к станции.

AS-Interface

Actuator-Sensor-Interface (интерфейс датчиков и приводов) является подсетью для самого низкого уровня процесса в системах PLC. С его помощью могут быть соединены в сетевую структуру двоичные датчики и приводы.



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.4



Information and Training Center
Knowledge for Automation

Службы

Коммуникационные службы описывают функции связи с определенными рабочими характеристиками типа обмена данными, управления устройствами, связи с приборами наблюдения и загрузки программ.

Глобальные данные

GD (Global Data - глобальные данные в сети) предназначены для циклического обмена данными малых объемов (в S7-400 дополнительно управляемые событиями).

S7 коммуникации

Эти утилиты связи оптимизированы для коммуникаций между S7 PLC, PG PC и OP/TD с помощью SIMATIC S7 соединения.

- функции PG; PG может быть соединен без конфигурируемого соединения.
- функции HMI; OP может быть соединен без конфигурируемого соединения.
- Базовые коммуникации выполняются с помощью SFC, которые содержатся в операционной системе CPU. (SFC связь выполняется без конфигурируемого соединения).
- Расширенная система коммуникаций происходит через конфигурированные соединения с помощью SFB (S7-400 - клиент / сервер; S7-300 - только сервер).

FDL (SDA)

Для безопасной передачи данных средних объемов между SIMATIC S7 и S5. Соответствует уровню 2 *Fieldbus Data Link (FDL)* для Profibus.

ISO Transport

Используется для безопасной передачи данных между SIMATIC S5 и S7.

ISO-on-TCP

Предназначается для пересылки средних объемов данных (до 240 байтов).

Используется для безопасной передачи средних количеств данных из SIMATIC S7 в PC или в несименсовские системы через TCP/IP сети. FDL, ISO и ISO-on-TCP утилиты становятся доступными через обращения к функции AG-SEND/AG-RECEIVE.

FMS

Fieldbus Message Specification (FMS) создает объектно-ориентированную связь между интеллектуальными партнерами также как между полевыми устройствами. Утилиты, обеспечиваемые FMS (переменные, службы областей (доменов), и т.д.) определены в EN 50170 том 2.

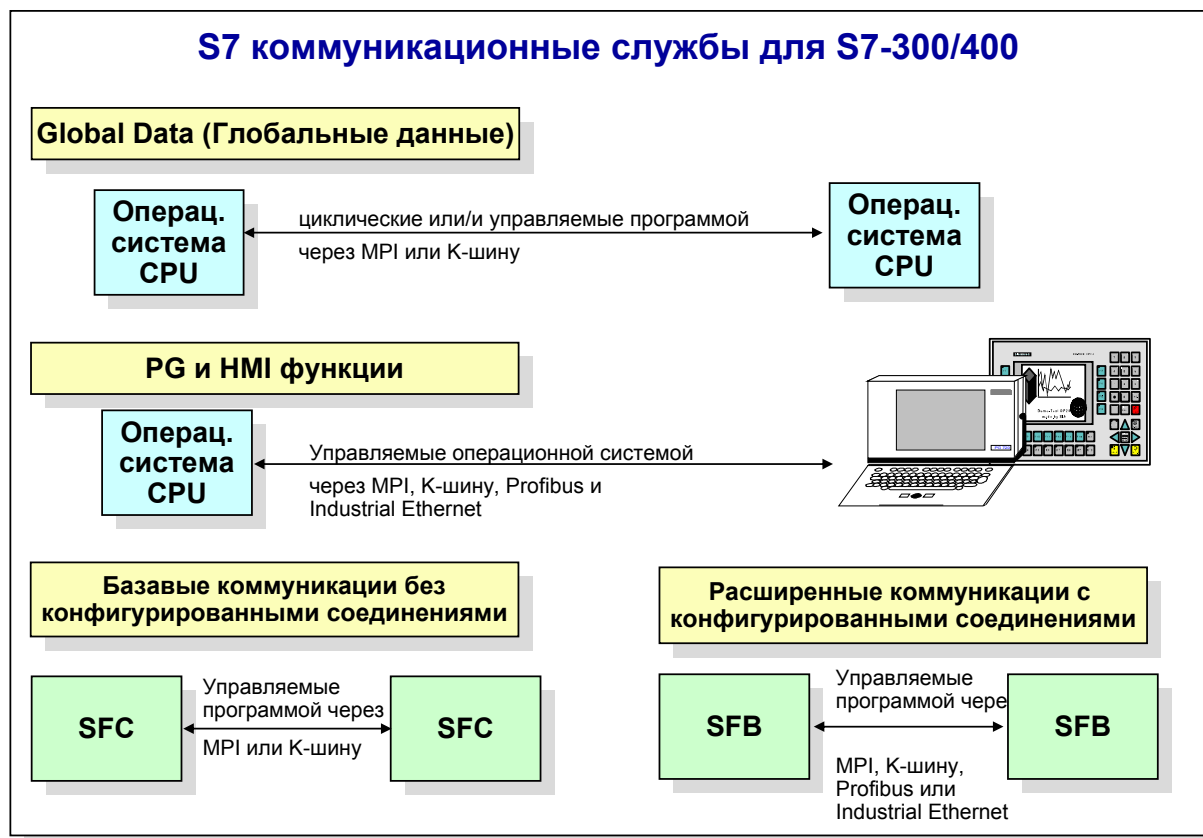
MAP

Первоначально разработанный американской автомобильной компанией "Дженерал Моторс", этот протокол предназначен для объектно-ориентированной связи между системами PLC (*MAP = Manufacturer Automation Protocol*).

DP

DP (Распределенный ввод - вывод) протокол специально оптимизирован для критичной ко времени связи между интеллектуальными устройствами управления (DP мастера) и полевыми устройства (EN 50170 том 3).

S7 коммуникационные службы для S7-300/400



SIMATIC S7

Siemens AG 1999. All rights reserved.

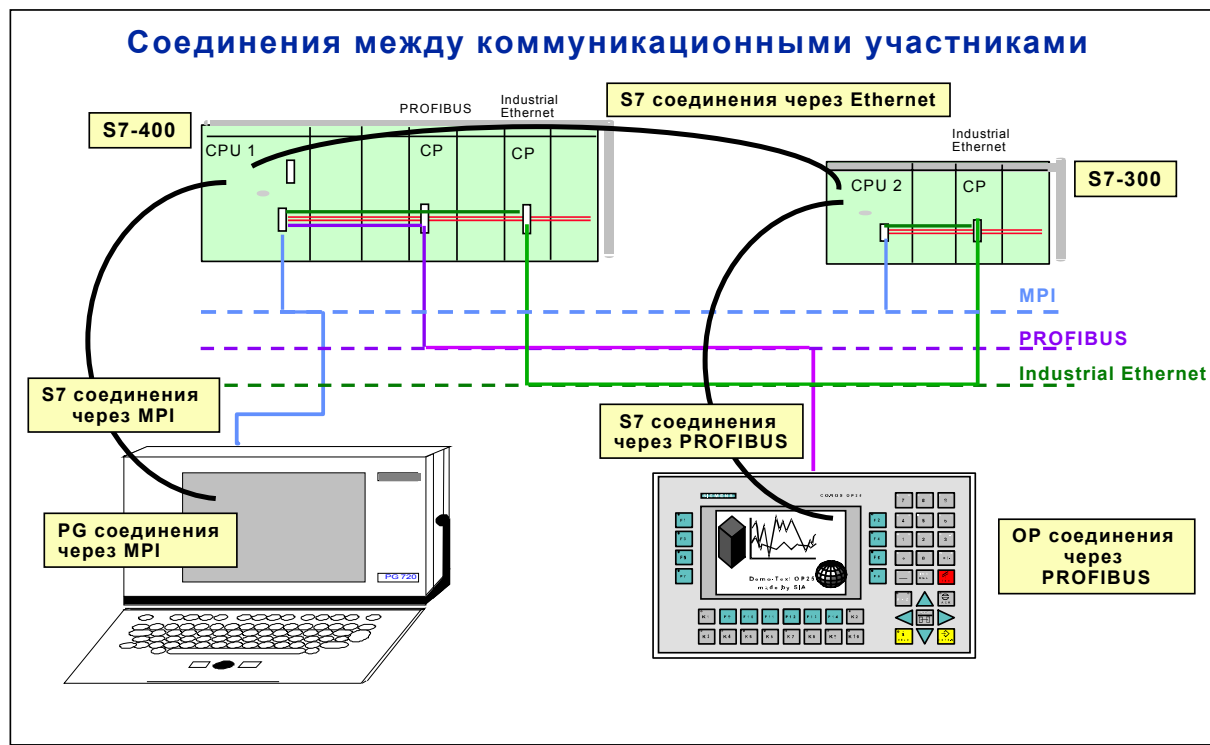
Date: 04.11.2005
File: PRO2_10E.5Information and Training Center
Knowledge for Automation

Глобальные данные Эта связь предназначена для данных, которые нужно передавать циклически между CPU, использующими интерфейс MPI и не используя программу пользователя. Обмен данными происходит в точках циклического управления вместе с модификацией изображения процесса.

PG и HMI функции Системные службы типа функций PG и HMI основан в конечном счете на расширенных S7-коммуникациях. Предпосылка для соединения PG или HMI устройства к S7-300/400 системе - наличие свободного соединения у партнера соединения (S7-CPU, M7-CPU, M7-FM, и т.д.).

Базовые коммуникации С помощью этих служб связи данные для всех S7-300/400 CPU могут передаваться с помощью MPI подсети или внутри станции через К-шину. Системные функции (SFC), типа X_SEND на посылающей стороне и X_RCV на стороне приемника, вызываются программой пользователя. Количество данных пользователя, которые могут быть перемещены в одном обращении - максимум 76 байтов.

Расширенные коммуникации Соединение с партнером связи активно конфигурируется, когда вызываются системные функции и разъединяются после передачи. Конфигурирование соединения из SIMATIC Manager для этого не нужно. Вы можете использовать эти службы связи для всех S7-400 CPU. Могут передаваться данные объемом максимум до 64 Кбайт с помощью различных подсетей (MPI, К-шины, Profibus, и Industrial Ethernet). Системные функциональные блоки (SFB) используются как интерфейс программирования. Эти SFB интегрированы только в операционную систему S7-400 CPU, они не существуют в S7-300. Помимо функций для передачи данных, коммуникационные службы также содержат функции управления PLC-партнером типа START и STOP. Связь происходит через конфигурированные соединения (таблица соединения). Эти связи конфигурируются во время параметрирования станции, включаются и существуют постоянно.



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.6



Information and Training Center
Knowledge for Automation

Соединения

Соединение - логическое назначение двух партнеров по коммуникациям для проведения служб связи. Соединение связано непосредственно с коммуникационной службой.

Каждое соединение имеет две позиции (каждая на CPU, предназначенном для S7-связей или на CP для FDL-связей), которые содержат необходимую

информацию для адресации партнера по связи, а также дополнительные атрибуты для конфигурации соединения.

Связи могут занимать один или несколько ресурсов соединения на участвующем в связи модуле (CPU, CP, FM).

Чтобы гарантировать организованную конфигурацию соединения, связи должны быть активны в одной конечной позиции и пассивны в другой конечной позиции. Иначе соединение не может быть установлено.

Приложение

В зависимости от выбранных функций связи, используются или конфигурируемые (расширенная система коммуникаций) или неконфигурируемые (базовые коммуникации) коммуникации.

Конфигурируемые коммуникации

Этот тип соединения конфигурируется с помощью STEP 7. Соединению конечных позиций назначен локальный ID, который, между прочим, идентифицирует собственный адрес и адрес партнера по связи. Коммуникационные функции, которые инициализированы SIMATIC-OP или PC, требуют конфигурированных связей. Они, однако, конфигурируются с их собственным инструментом (например, ProTool или COML).

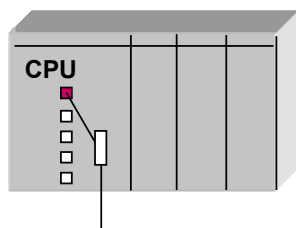
Конфигурируемые связи конфигурируются активными узлами при включении и остаются конфигурированным в течение всего времени работы.

Неконфигурируемые коммуникации

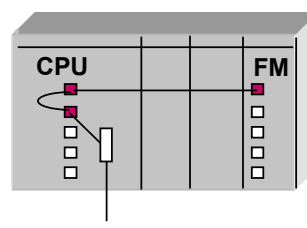
Эти связи конфигурируются, когда вызывается функция связи и, если требуется, разъединяются после того, как передача данных завершена.

Назначение ресурсов соединений для S7 коммуникаций

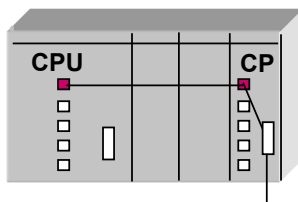
S7-300/400:
MPI или
встроенный
PROFIBUS-DP
интерфейс



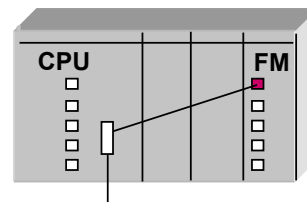
S7-300:
встроенный
PROFIBUS-DP



S7-300/400:
Industrial Ethernet
или PROFIBUS-CP

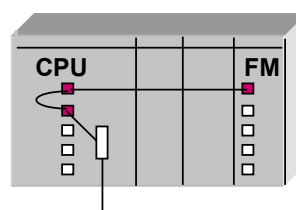


S7-300:
MPI интерфейс



□ Свободные ресурсы соединения
■ Занятые ресурсы соединения

S7-400:
MPI
или встроенный
PROFIBUS-DP



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.7



Information and Training Center
Knowledge for Automation

Краткий обзор

В участвующих станциях ресурсы соединения для конечной позиции или для переходной позиции (например, CP) требуются для каждого соединения. Число ресурсов соединения зависит от CPU/CP. Если все ресурсы соединения партнера по связи заняты, никакое новое соединение не может быть установлено.

S7 функции для CPU Для S7 функций ресурс соединения на конечной позиции резервируется через интегрированный интерфейс MPI-/PROFIBUS-DP для S7 соединения на CPU.

Для S7 функций через внешний интерфейс CP, один ресурс соединения всегда занят на CPU (для конечной позиции) и на CP (переходная позиция) для S7-соединения.

S7 функции для FM

Функции S7 для функционального модуля (FM) через внутренний MPI-/PROFIBUS-DP интерфейс занимают два ресурса соединения (для двух позиций перехода) на S7 соединение на CPU S7-400 и один ресурс соединения на FM (для конечной позиции).

Это также верно для каждого дополнительного CPU (много процессорная обработка) внутри одной и той же станции, посредством чего дополнительные CPU соединены косвенно через K-шину с MPI подсетью.

PG/OP

Каждое соединение с PG или OP/TD требует ресурса соединения на SIMATIC S7/M7-CPU. По умолчанию, ресурс соединения для соединения каждого из PG и OP/TS зарезервирован в каждом S7/M7-CPU.

Доступный ресурс соединения требуется для каждого дополнительного PG/OP соединения. Если несколько PG/OPS соединены, число доступных ресурсов соединения для S7-функций уменьшается.

Характеристики связи S7-CPU

CPU 312	IFM CPU 313	CPU 314	CPU 315/-2 DP	CPU 316	CPU 318-2
1 PG 1 OP 2 для S7 функ.	1 PG 1 OP 2 для S7 функ. 4 для SFC	1 PG 1 OP 2 для S7 функ. 8 для SFC	1 PG 1 OP 2 для S7 функ. 8 для SFC	1 PG 1 OP 2 для S7 функ. 8 для SFC	1 PG 1 OP 30 для S7 функ. или 30 для SFC

CPU 412-1	CPU 413-1/2 DP	CPU 414-1/2 DP	CPU 416-1/2DP	CPU 417-4
1 PG 1 OP1 OP 14 для S7 функ. или 14 для SFC	1 PG 1 OP 14 для S7 функ. или 14 для SFC	1 PG 1 OP 30 для S7 функ. или 30 для SFC	1 PG 1 OP 62 для S7 функ.. или 62 для SFC	1 PG 62 для S7 функ.. или 62 для SFC

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.8Information and Training Center
Knowledge for Automation

Коммуникационные ресурсы CP CP имеет следующее число ресурсов соединения:

CP для S7-300**CP 343-1**16 S7 функ.
16 ISO-Trans**CP 343-1 TCP**16 S7 функ.
16 TCP/IP**CP 342-5**16 S7 функ.
16 FDL**CP 343-5**16 S7 функ.
16 FDL
16 FMS**CP для S7-400****CP 443-1**48 S7 функ.
64 ISO-Trans.**CP 443-1 TCP**48 S7 функ.
64 TCP/IP**CP 443-5
Extended**32 S7 функ.
32 FDL**CP 443-5
Basic**32 S7 функ.
32 FDL
32 FMSS7 функ.: для S7-функций через PG/OP или SFBISO-Trans.: ISO transport соединениеTCP/IP: ISO-on-TCP соединениеFDL: FDL соединениеFMS: FMS соединение

SFC коммуникации: обзор

- Обмен данными, использующий MPI подсеть или внутри станции
- Соединения не требуют конфигурирования соединений по сравнению SFB коммуникациями
- Соединение с партнером динамически конфигурируется и разъединяется
- Объем пользовательских данных до 76 байт
- Могут использовать все CPU S7-300/400
- Переменные могут также читаться и записываться в S7-200 через PROFIBUS-DP (X_GET, X_PUT)
- Партнеры связи могут также находиться в другом S7 проекте

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.9



Information and Training Center
Knowledge for Automation

Краткий обзор

Вы можете обмениваться небольшими объемами данных между S7/M7-300/400-CPU и модулями, способными к дополнительной связи с помощью SFC для неконфигурируемых соединений.

Партнеры по связи должны или быть соединены той же самой MPI подсетью или быть доступными внутри одной и той же самой станции через K-шину или PROFIBUS-DP.

Конфигурированное соединение не нужно.

Соединение

Когда вызывается связанная SFC, соединение динамически конфигурируется с адресованным партнером по связи и после завершения передачи, в зависимости от назначения параметра (параметр: CONT), разъединяется. Для конфигурирования соединения требуется доступный ресурс соединения у каждого партнера по связи.

Если при обращении SFC нет доступного ресурса соединения, то в RET_VAL пользователю возвращается соответствующий номер ошибки.

Уже существующие связи связи SFB не могут использоваться. Если активный CPU входит в состояние STOP во время передачи данных, существующие связи разрываются.

Коммуникационная SFC не может быть удалена в режиме RUN, т.к. иначе занятые ресурсы соединения будут недоступны. (Программа изменяется только в состоянии STOP).

Размер пользовательских данных

Количество передаваемых данных пользователя - максимум 76 байтов для всех S7/M7/C7-CPU.

SFC коммуникации: обзор блоков

SFC	Имя	Краткое описание
SFC 65	X_SEND	Блок-передатчик для отправки данных блоку X_RCV (клиент)
SFC 66	X_RCV	Блок-приемник для получения данных от блока X_SEND
SFC 67	X_GET	Чтение данных из PLC-партнера
SFC 68	X_PUT	Запись данных в PLC-партнер
SFC 69	X_ABORT	Аварийный разрыв существующего соединения
SFC 72	I_GET	Чтение данных из CPU-партнера
SFC 73	I_PUT	Запись данных в CPU-партнер
SFC 74	I_ABORT	Аварийный разрыв соединения с CPU-партнером

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.10Information and Training Center
Knowledge for Automation

Краткий обзор

Коммуникационные SFC предлагают возможность квитированной передачи данных, используя неконфигурируемые S7-коммуникации.

С помощью коммуникационных SFC (X_ ...), Вы можете адресовать всех партнеров по связи в той же самой MPI подсети, с помощью SFC (I_ ...) всех партнеров по связи с адресом ввода - вывода (например, FM и т.д.) внутри той же самой станции.

Связь, использующая MPI подсеть, также возможна, если партнер по связи находится в другом S7-проекте.

Число последовательно доступных узлов связи не ограничено.

Адресация

В коммуникациях (X_ ...), использующих MPI подсеть, адресация партнера происходит, определяя адрес MPI, в коммуникациях (I_ ...) внутри той же самой станции, - определяя логический начальный адрес модуля (адрес ввода - вывода).

Если модуль имеет базовый адрес для входов (I-address) и так же для выходов (Q-address), то в вызове SFC указывается меньший из двух.

Консистентные данные Размер максимальной области данных, которая может читаться (непротиворечивые (X_PUT, I_PUT) и записываться (X_GET, I_GET), как связанный блок операционной системой с S7-300/400- CPU, обозначена как консистентные данные).

У S7-300/400 консистентные данные:

- S7-300-CPU: 8 Байты
- S7-400-CPU: 32 Байты

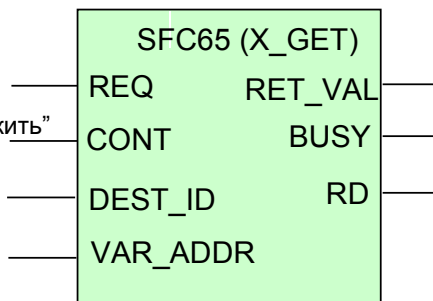
Таким образом, например, массив типа байта, слово или двойное слово может быть передан непротиворечивым до максимального размера.

SFC коммуникации: блок X_GET (SFC 67)

STL представление

с примером для назначения параметров

```
CALL SFC 67
REQ:= I 0.4           //Запрос активации
CONT:= FALSE          // Управл. парам."продолжить"
DEST_ID:= W#16#3      //MPI адрес
VAR_ADDR:= P#M20.0 BYTE 10 //Удаленная перем.
RET_VAL:= MW100        //Код ошибки
BUSY:= M 4.1          //Активность SFC
SD:= P#M0.0 BYTE 10   //Местная (локальная)
                     //переменная
```



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.11



Information and Training Center
Knowledge for Automation

Описание

С помощью SFC 67 (X_GET), Вы можете читать данные от партнера по связи, который не находится в локальной S7-станции. Нет никакого соответствующего SFC на партнере по связи.

Работа чтения активизирована после вызова SFC с REQ = 1. После этого SFC должна быть вызвана, пока прием данных не закончится (BUSY=0). При этом RET_VAL содержит длину полученного блока данных в байтах.

Удостоверитесь, что область для приема, определенная параметром RD (на приемном CPU), по крайней мере не короче области, которую нужно читать (определенную параметром VAR_ADDR на партнере по связи). Типы данных RD и VAR_ADDR должны также соответствовать.

Параметры для SFC 67 X_GET

Параметр	Вид	Тип	Назначение
REQ	INPUT	BOOL (I,Q,M,D,L, const.)	Активизация передачи при сигнале 1
CONT	INPUT	BOOL (I,Q,M,D,L, const.)	CONT=0 разрыв соединения CONT=1 сохранение соединения
DEST_ID	INPUT	WORD (I,Q,M,D,L, const.)	MPI адрес партнера
VAR_ADDR	INPUT	ANY (I,Q,M,D)	Ссылка на область (удаленный CPU), из которой читаются данные
RET_VAL	OUTPUT	INT (I,Q,M,D,L)	Возвращаемое значение с кодом ошибки
BUSY	OUTPUT	BOOL (I,Q,M,D)	BUSY=1 функция работает BUSY=0 функция завершила работу
RD	OUTPUT	ANY (I,Q,M,D,L)	Ссылка на область (локальный CPU), в которую данные чтения записываются

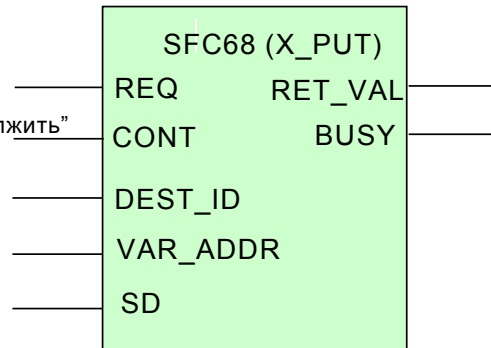
SFC коммуникации: блок X_PUT (SFC 68)

STL представление

с примером для назначения параметров

```
CALL SFC 68
REQ:= I 0.5           // Запрос активации
CONT:= FALSE          //Управл. парам."продолжить"
DEST_ID:= W#16#3      //MPI адрес
VAR_ADDR:= P#M20.0 BYTE 10 // Удаленная переменная
SD:= P#M0.0 BYTE 10  //Местная переменная
RET_VAL:= MW100       //Код ошибки
BUSY:= M 4.1          //Активность SFC
```

LAD/FBD представление



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.12



Information and Training Center
Knowledge for Automation

Описание

С помощью SFC 68 (X_PUT) Вы записываете данные партнеру по связи, который находится не в той же самой локальной S7-станции. Нет никакого соответствующего SFC на партнере по коммуникации.

Работа записи активизирована после вызова SFC с REQ = 1. После этого Вы продолжаете вызывать SFC, пока не получено квитирование с BUSY=0.

Удостоверитесь, что посылающаяся область, определенная параметром SD (на передающем CPU), имеет ту же самую длину, что и принимающая область, определенная параметром VAR_ADDR (на партнере по связи). Типы данных SD и VAR_ADDR должны также соответствовать.

Параметры для SFC 68 X_PUT

Параметр	Вмд	Тип	Назначение
REQ	INPUT	BOOL (I,Q,M,D,L, Const.)	Активизация передачи сигналом 1
CONT	INPUT	BOOL (I,Q,M,D,L, Const.)	CONT=0 соединение разорвать CONT=1 соединение сохранить
DEST_ID	INPUT	WORD (I,Q,M,D,L, Const.)	MPI адрес партнера
VAR_ADDR	INPUT	ANY (I,Q,M,D)	Ссылка на область (удаленный CPU), в который идет запись.
SD	INPUT	ANY (I,Q,M,D)	Ссылка на область (локальный CPU), которая содержит данные для записи
RET_VAL	OUTPUT	INT (I,Q,M,D,L)	Возвращаемое значение с кодом ошибки
BUSY	OUTPUT	BOOL (I,Q,M,D)	BUSY=1 функция работает BUSY=0 функция завершила работу

SFC коммуникации: блок X_SEND (SFC 65)

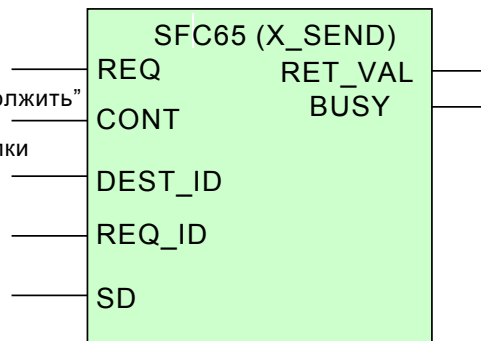
STL представление

с примером для назначения параметров

```
CALL SFC 65
REQ:= M4.0
CONT:= FALSE
DEST_ID:= W#16#4
REQ_ID:= DW#16#1
SD:= P#M20.0 BYTE 10
RET_VAL:= MW40
BUSY:= M 4.1
```

```
// Запрос активации
// Управл. парам."продолжить"
//MPI адрес
//Идентификатор посылки
//Переменная
//Код ошибки
//Активность SFC
```

LAD/FBD представление



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.13



Information and Training Center
Knowledge for Automation

Описание

С помощью SFC 65 (X_SEND) Вы посылает данные партнеру по связи, который находится вне данной S7-станции. Получение данных в партнере по связи происходит через SFC 66 (X_RCV).

Вы можете идентифицировать ваши посланные данные входным параметром REQ_ID. Этот идентификатор работы также передается. Вы можете оценивать его в партнере по связи, чтобы определить начало данных. Передающая функция начинает работать после вызова SFC с REQ = 1. Вы должны удостовериться, что передаваемая область (на посылающем CPU), определенная через параметр SD меньшая или такая же, как область приема (в партнере по связи), определенная через параметр RD в SFC 66 (X_RCV).

Параметры для SFC 65 X_SEND

Параметр	Вид	Тип	Назначение
REQ	INPUT	BOOL (I,Q,M,D,L const.)	Активизация передачи при сигнале 1
CONT	INPUT	WORD (I,Q,M,D,L Const.)	CONT=0 соединение разорвать CONT=1 соединение сохранить
DEST_ID	INPUT	WORD (I,Q,M,D,L const.)	MPI адрес партнера
REQ_ID	INPUT	DWORD (I,Q,M,D,L, const.)	ID запроса для идентификации данные в партнере
SD	INPUT	ANY (I,Q,M,D)	Указатель на область для передачи
RET_VAL	OUTPUT	INT (I,Q,M,D,L)	Возвращаемое значение с кодом ошибки
BUSY	OUTPUT	BOOL (I,Q,M,D)	BUSY=1 функция работает BUSY=0 функция завершила работу

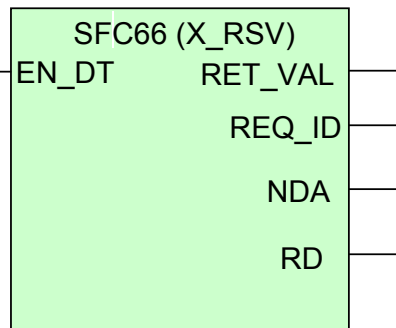
SFC коммуникации: блок X_RCV (SFC 66)

STL представление

с примером для назначения параметров

```
CALL SFC 66
EN_DT:= TRUE           //Триггер переноса данных
RET_VAL:= MW 50        //Код ошибки
REQ_ID:= MD52          //ID принятой посылки
NDA:= M40.0            //Существование данных
RD:= P#M20.0 BYTE 10  //Переменная для приема
```

LAD/FBD представление



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.14



Information and Training Center
Knowledge for Automation

Описание

С помощью SFC 66 (X_RCV) Вы получаете данные от одного или нескольких партнеров по связи, посланные функцией SFC 65 (X_SEND). Партнеры по связи находятся вне данной S7-станции.

С помощью SFC 66 (X_RCV) Вы можете:

- определить, являются ли в текущий момент времени посланные данные доступными. Они помещаются, в случае необходимости, во внутреннюю очередь операционной системой.
- копирует самый старый блок данных, который является доступным в очереди, в область приема, определенную Вами.

Выбор происходит через входной параметр EN_DT (разрешение переноса данных).

Параметры для X_RCV SFC 66

Параметр	Вид	Тип	Назначение
EN_DT	INPUT	BOOL (I,Q,M,D,L, constant)	EN_DT=0 Проверить, присутствует ли блок данных EN_DT=1 скопировать блок данных в память
RET_VAL	OUTPUT	INT (I,Q,M,D,L)	Возвращаемое значение с кодом ошибки
REQ_ID	OUTPUT	DWORD (I,Q,M,D,L,)	Идентификатор вызова для X_SEND SFC 66, чьи данные присутствуют в первой позиции очереди
NDA	OUTPUT	BOOL (I,Q,M,D,L)	NDA=0 нет блока данных NDA=1 По крайней мере 1 блок данных представлен (для EN_DT = 0) или блок данных был скопирован в память (EN_DT = 1)
RD	OUTPUT	ANY (I,Q,M,D)	Указатель на область для приема данных

SFB коммуникации: обзор

- Обмен данными, использующий MPI, K-шину, Profibus или Industrial Ethernet
- Конфигурирование соединений через таблицу соединений
- Связи конфигурируются во время полного рестарта и существуют постоянно (даже в режиме STOP)
- Объем пользовательских данных до 64 Кбайт
- Коммуникационные службы также для управления (Stop, Start) партнером
- SFB существуют для всех CPU S7-400
- Данные могут читаться и записываться в S7-300 (GET/PUT)
- Различные задачи могут быть обработаны через одно соединение

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.15



Information and Training Center
Knowledge for Automation

Краткий обзор

SFB блоки доступны на всех S7-400-CPU и используются для обмена информацией между станциями с S7/M7-300/400-CPU. Эти блоки могут передавать через различные подсети (MPI, Profibus, Industrial Ethernet) данные объемом до 64 Кбайтов.

Соединения

Коммуникационные SFB предлагают возможность защищенной передачи данных, используя конфигурированные S7-соединения. Конфигурация этих связей происходит с помощью "Verpro" (конфигурирование соединения) инструмент, который интегрирован в SIMATIC Manager.

Сконфигурированные соединения создаются во время COMPLETE RESTART станции и существуют постоянно, даже тогда, когда станция переходит в режим STOP. Во время рестарта связи снова не конфигурируются.

Коммуникации возможны исключительно между станциями S7-проекта. Партнеры по связи должны быть соединены общей MPI -, PROFIBUS- или Industrial Ethernet подсетью.

SFB

Интерфейсы для S7-коммуникаций с программой пользователя в SIMATIC S7 формируются специальными S7-блоками типа SFB. SFB ориентированы на стандарт ISO/IEC 1131-5 и предлагают пользователю однородный интерфейс.

Соединения должны быть сконфигурированы для коммуникаций. Номера соединений ссылаются на назначение узла и среду передачи посредством идентификационных номеров. Эти идентификационные номера переданы как блочный параметр "ID" во время вызова SFB.

Данные пользователя

Размер данных пользователя зависит от используемого блока и от партнера по связи:

- PUT/GET 160 байт для S7-300 и 400 байт для S7-400/M7
- USEND/UREC до 440 байт
- BSEND/BRCV до 64Кбайт

SFB коммуникации: обзор блоков

SFB/SFC	имя	Тип связи	Короткое описание
SFB 8	USEND	двусторонняя	Блок-передатчик для передачи данных URCV блоку (клиент)
SFB 9	URCV	двусторонняя	Блок-приемник для получения данных от блока USEND
SFB 12	BSEND	двусторонняя	Блок-передатчик для посылки больших блоков данных блоку BRCV (до 64 Кбайт)
SFB 13	BRCV	двусторонняя	Блок-приемник для получения больших блоков данных (до 64 Кбайт)
SFB 14	GET	односторонняя	Чтение данных PLC-партнера
SFB 15	PUT	односторонняя	Запись данных PLC-партнеру
SFB 16	PRINT	односторонняя	Посылка данных на удаленный принтер
SFB 19	START	односторонняя	Выполнение полного рестарта у партнера
SFB 20	STOP	односторонняя	Перевод партнера в состояние Stop
SFB 21	RESUME	односторонняя	Выполнение рестарта у партнера
SFB 22	STATUS	односторонняя	Опрос состояния партнера (RUN, STOP, start-up, hold)
SFB 23	USTATUS	односторонняя	Получение сообщения о состоянии партнера
SFC 62	CONTROL	---	Просмотр внутреннего состояния S7- соединения + SFB

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.16Information and Training Center
Knowledge for Automation

SFB: S7- 400

SFB для S7-коммуникаций интегрированы как системные функциональные блоки (SFB) в операционную систему CPU S7-400 .

Для интеграции в программу пользователя, пользователь может размещать блочные заголовки в S7-программе из *Standard Library V3.x* , папка *System Function Blocks* .

SFB: S7 - 300

S7-300 не содержит никаких SFB для расширенной системы коммуникаций. Однако, операционная система CPU S7-300 поддерживает функциональные возможности сервера односторонних S7-4xx коммникаций. Таким образом, например, данные из CPU 3xx могут читаться или записываться CPU с помощью блоков GET и PUT.

Классы функций

Блоки могут подразделяться на 4 функциональных класса:

- функции передачи и приема
- функции управления
- функции для мониторинга
- функция просмотра

SFB для обмена данными

SFB для обмена данными используются для передачи данных между двумя партнерами (S7/M7-CPU, M7-FMs) способными к коммуникациям:

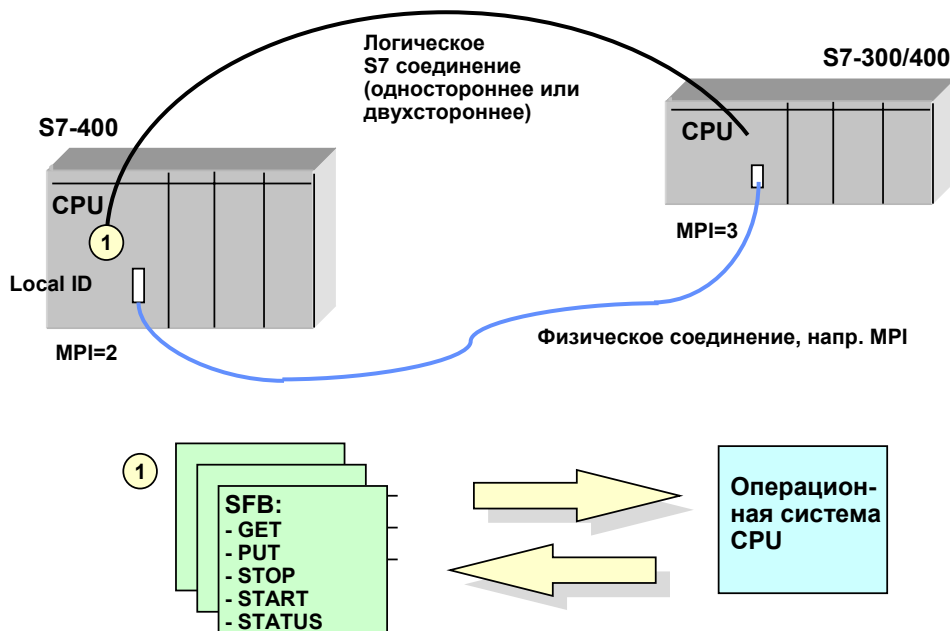
- GET, PUT (Одностороннее чтение и запись переменных)
- USEND/URCV (Двусторонняя, некоординированная передача / прием)
- BSEND/BRCV (Двусторонняя, блоковая передача / прием)

SFB для управления программой

SFB для управления пользовательской программой, наблюдения и оценки состояния используются, чтобы управлять и оценить оперативные состояния партнеров или соединений.

- START/STOP/RESUME (Функции управления)
- STATUS/USTATUS (Контролирующие функции)
- CONTROL (Функция просмотра)

Односторонние коммуникационные службы, использующие S7 соединения



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.17Information and Training Center
Knowledge for Automation

Краткий обзор

Чтобы SFB в соответствующих партнерах по коммуникации могли связываться друг с другом, S7-коммуникации должны прежде всего быть сконфигурированы. S7-коммуникации могут быть сконфигурированы для MPI, Industrial Ethernet и PROFIBUS сетей.

Односторонние S7-коммуникации

Односторонние S7-коммуникации между S7-400 и S7-300 автоматически устанавливаются инструментом конфигурации. Для односторонних коммуникаций ID локального соединения для идентификации соединения, то есть партнера по связи и среды передачи, назначается только на стороне S7-400 (клиентская сторона).

Только односторонние коммуникационные службы могут вызываться через односторонние связи. Соответствующий вызов SFB для односторонних служб связи необходим только на стороне клиента (S7-400). На другом партнере по связи (сервер), обслуживание полностью производится операционной системой. Работопользователя по программированию на стороне сервера не нужна.

Односторонние S7-коммуникации всегда конфигурируются клиентом во время старта (start-up).

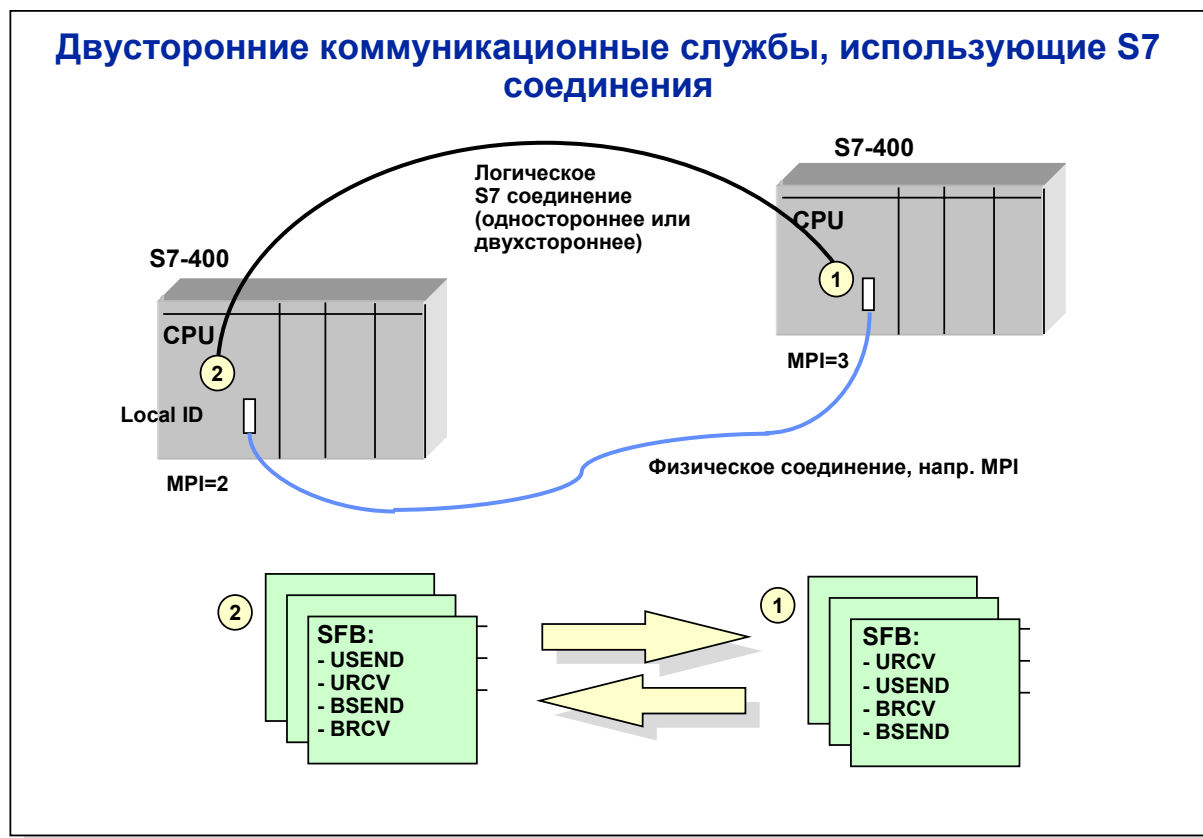
"Односторонние" SFB

SFB, которые рассматриваются как односторонние службы связи:

- GET, PUT
- STOP, START, RESUME
- STATUS, USTATUS

С односторонними службами связи программа пользователя на стороне сервера не информируется, когда были переданы новые данные.

Двусторонние коммуникационные службы, использующие S7 соединения



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.18



Information and Training Center
Knowledge for Automation

Двусторонние S7-коммуникации

Двусторонние S7-коммуникации автоматически устанавливаются при конфигурации S7-соединений между двумя CPU S7-400. ID соединения назначается на каждой стороне двустороннего соединения. Обе стороны могут затем ссылаться на соединение, используя этот ID соединения. Таким образом, каждый из двух партнеров может являться инициатором (клиентом) службы связи. Односторонний (PUT, GET и т.д.), также как двусторонние службы связи, могут быть завершены, используя двусторонние коммуникации.

"Двусторонние" SFB Блоки

- BSEND = Передатчик (Клиент) == > BRCV Приемник (Сервер)
- USEND = Передатчик (Клиент) == > URCV Приемник (Сервер)

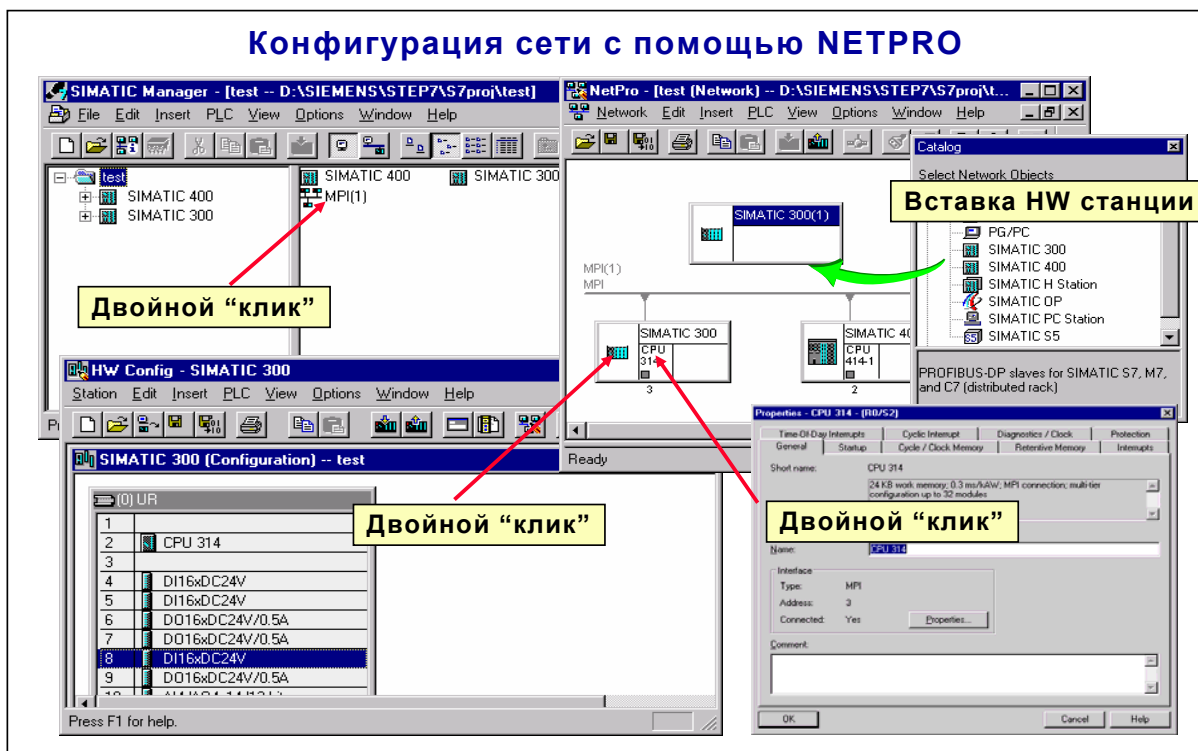
рассматриваются как двусторонние SFB.

Эти блоки должны всегда устанавливаться как блочные пары.

Двусторонние функции связи всегда устанавливаются, когда передача данных используется для специфической дальнейшей обработки данных.

С одной стороны, приемник (сервер) может определяться вызовом блока URCV или BRCV, когда приложение готово получить новые данные от передатчика для дальнейшей обработки. С другой стороны, приемник может, оценивая SFB-параметр #NDR (New Data Received - Новые Полученные Данные), быть информировано, если новые данные были получены.

Конфигурация сети с помощью NETPRO



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.19



Information and Training Center
Knowledge for Automation

Введение

Графическая конфигурация сетей (MPI, Profibus или Industrial Ethernet) может быть проведена с помощью инструмента "NETPRO". Преимущество такого конфигурирования - в ясности, документированности и простом обращении с участвующими инструментальными средствами типа Hardware Configuration.

Вызов

Инструмент вызывается с двойным щелчком на сетевом символе, например, MPI в SIMATIC Manager.

Вставка HW-станции В каталоге Вы найдете необходимые компоненты типа подсетей и станций, которые Вы можете вставлять методом "drag and drop".

Конфигурирование HW (аппаратуры) После того, как Вы вставили станции, Вы можете конфигурировать их, открывая инструмент "Configure Hardware", дважды нажимая на "Hardware Symbol" станции.

Здесь Вы можете вставлять модули в станции и назначать параметры для них. Для CPU Вы можете, между прочим, также устанавливать MPI-адрес и соединение с подсетью.

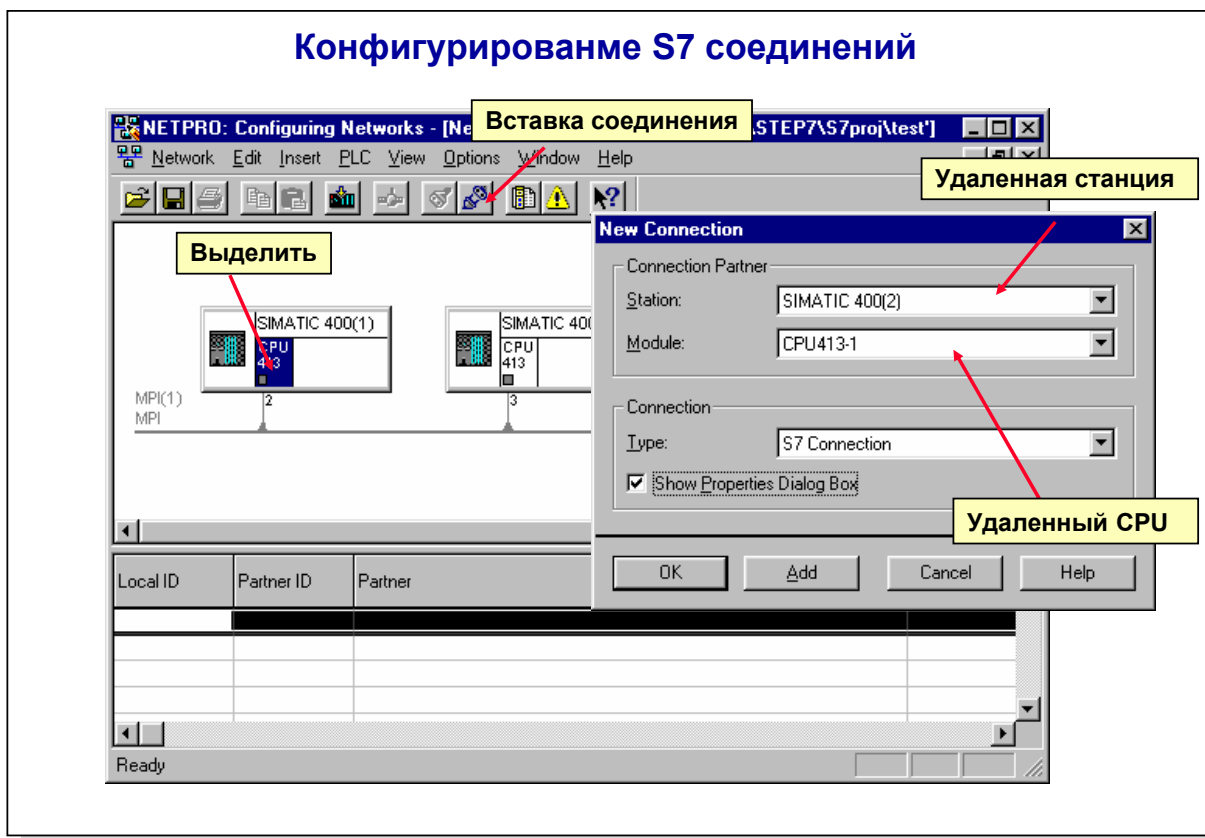
Прежде, чем Вы можете конфигурировать связи, все участвующие станции должны быть связаны с соответствующей подсетью.

Свойства CPU

Дважды нажимая на CPU внутри отображаемой станции, Вы открываете диалог "Properties- CPU". Здесь Вы можете устанавливать свойства CPU типа соединения с сетью, тактовый меркер и т.д.

Обратите внимание: Когда Вы загружаете данные конфигурации в CPU после конфигурирования внутри NETPRO, параметры CPU, типа соединения с сетью, тактовый меркер и т.д. не передаются в CPU. Данные конфигурации CPU, которые изменялись внутри NETPRO, должны быть перемещены, используя инструмент HW Config !

Конфигурирование S7 соединений



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.20



Information and Training Center
Knowledge for Automation

Краткий обзор

Создание требуемых связей - предпосылка для программы управляемого обмена данными, используя SFB. Все связи, которые имеет модуль, отображаются в таблице соединений, принадлежащей модулю.

Создание соединений

Соединения с удаленным партнером могут только быть установлены, когда локальная и удаленная станция соединены одной подсетью.

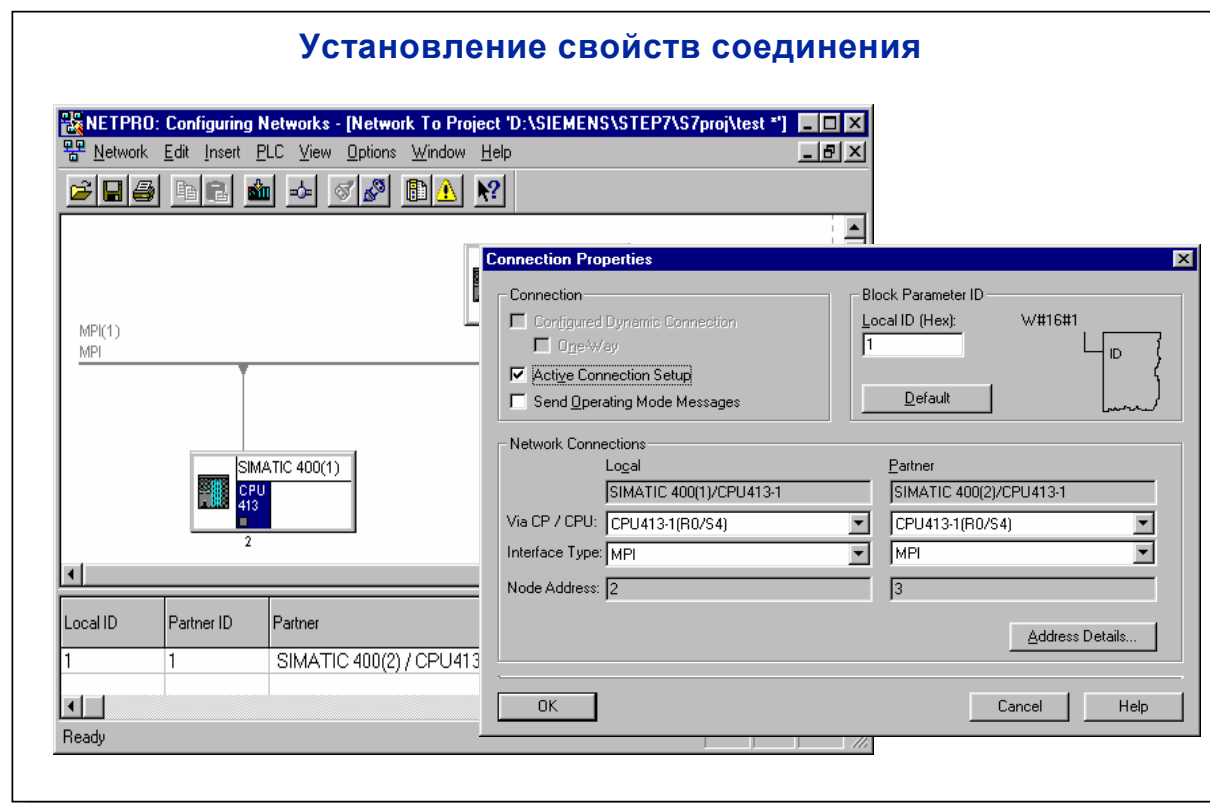
Чтобы вставить новое соединение, поступайте следующим образом:

1. В полях "Station" и "Module" выберите программируемый модуль, который Вы открываете, чтобы соединить (локальная станция).
2. Двойной щелчок на пустой строке таблицы соединений или выбор меню *Insert -> Connection...* открывает диалог "New Connection".
3. В полях "Station" и "Module" выбирают программируемый модуль, с которым нужно соединиться (партнер по соединению или "Remote Station"- "Удаленная станция").
4. В поле "Type" выбирают соединение (выберите: S7 Connection)
5. Активизируйте флажком "Show Properties Dialog Box", если после "OK" или "Add" Вы хотите посмотреть или изменить свойства соединения.
6. Подтвердите ваши действия, нажимая кнопку "OK".

Результат:

STEP 7 вводит соединение в таблицу соединений локальной станции и выдает Local ID (локальный ID) и, в случае необходимости, Partner ID (ID партнера) для этого соединения. Эти ID Вам требуются для программирования связи через SFB (значение для блочного параметра "ID").

Установка свойств соединения



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.21



Information and Training Center
Knowledge for Automation

Краткий обзор

Помимо установления партнера по соединению и типа соединения, Вы можете, в зависимости от типа соединения устанавливать дополнительные свойства.

Установка свойств объектов

Чтобы установить специальные свойства коммуникационного соединения, поступайте следующим образом:

1. Отметьте соединение, для которого Вы хотите устанавливать свойства.
2. Выберите опцию Edit -> Object Properties. Открыт диалог " Object Properties ". В этом диалоге Вы можете устанавливать следующие свойства.

Active Connection Set Up

Вы можете решать, который из двух узлов должен принять конфигурацию соединения при полном рестарте.

Send Operating Mode Messages

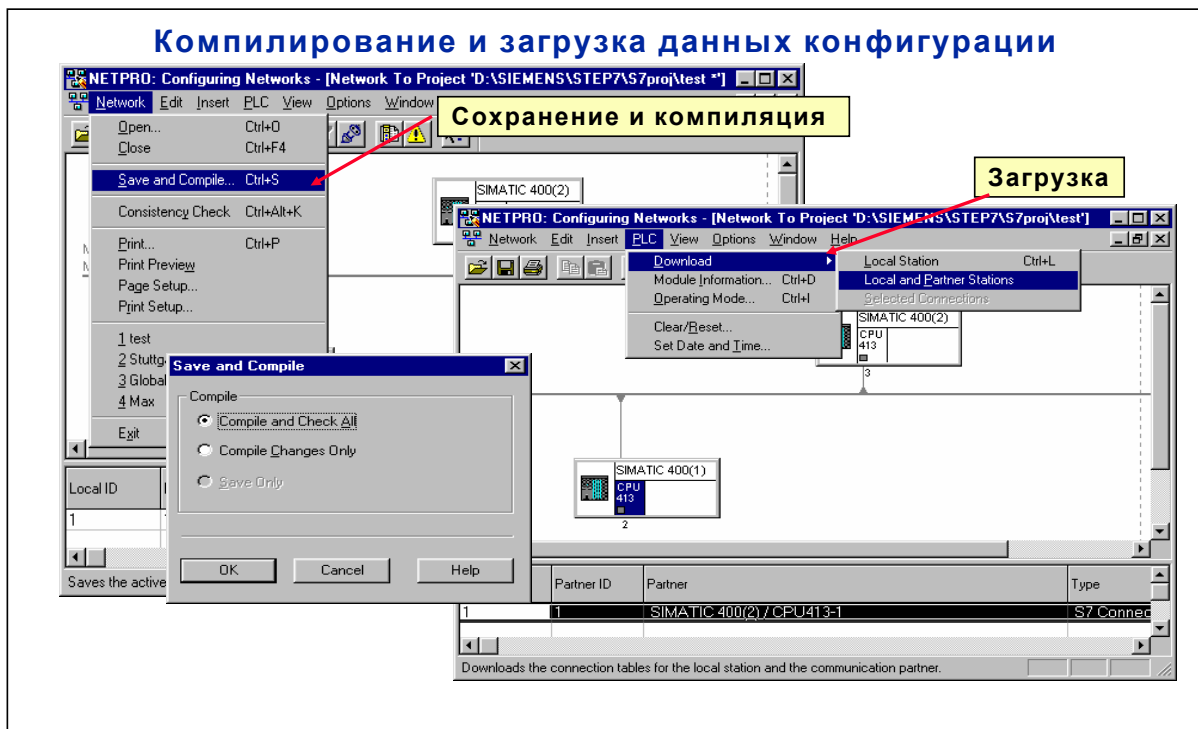
Когда активизировано, локальный узел посылает сообщения о рабочем режиме (STOP, START, HOLD,.....) партнеру или SFB 23: USTATUS в CPU партнера.

Local ID

Здесь отображается локальный ID соединения. Вы можете изменять локальный ID. Это имеет смысл, если Вы уже программировали связь SFB, и Вы опять хотите использовать ID, запрограммированный в вызове для идентификации соединения. Вы вводите новый локальный ID как шестнадцатеричный номер. Он должен быть в диапазоне значений от 1 до FFF для S7-соединения.

Network Connections Эти поля отображают, через какой путь выполняется обмен данными. Если между двумя узлами существуют различные пути связи (подсети), выбор может быть сделан, через какой путь должен быть совершен обмен данными.

Компилирование и загрузка данных конфигурации



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.22



Information and Training Center
Knowledge for Automation

Компилирование и сохранение

Прежде, чем Вы можете загрузить данные соединения в индивидуальные станции (Download in PLC), таблица соединения должна быть сохранена в NETPRO и скомпилирована в данные соединения. Это происходит с помощью опции меню *File -> Save and Compile..*

В поле диалога, которое появляется, Вы можете выбирать между двумя вариантами:

Compile and Check All: Сохраняет все связи внутри проекта и проверяет их на непротиворечивость. Все связи компилируются и сохраняются в системных данных (в SDB). В случае, если возникают противоречия, появляется поле диалога, в котором отображаются ошибки.

Выбирайте "Compile All and Check", если Вы сделали изменения в сетевой конфигурации (например изменили адрес узла, удалили узел или подсеть). Возможно, что связи больше не существуют, и только "Compile All and Check" дает эту информацию.

Compile Changes Only: Сохраняет все связи внутри проекта и компилирует те связи, которые были изменены, как последнее выполнение "Save and Compile".

Когда Вы заканчиваете конфигурацию соединения, на экране появляется вопрос, должны ли быть сохранены измененные данные или нет. После подтверждения вопроса ("Yes"), измененные данные соединения сохраняются и компилируются в системные данные.

Загрузка данных конфигурации

Из сохраненной таблицы соединений, результаты данных соединения, должен быть загруженный в участвующие модули. Загрузка таблицы соединений в модуль возможна через MPI, PROFIBUS или Industrial Ethernet.

Имеются пять способов загрузить данные в PLC:

- Download, Local Station (меню "PLC")
- Download, Local and Partner Stations (меню "PLC")
- Download, Marked Connections

(Для дальнейшей информации: см. On-line Help)

SFB коммуникации: блок GET (SFB 14)

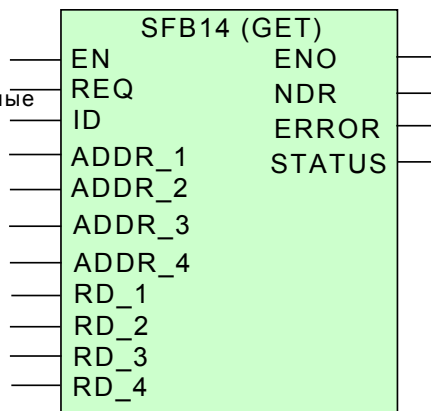
STL представление

с примерами назначения параметров

```
CALL GET, I_GET           //с экземпляром DB
REQ:=I 0.2               //старт
ID:=W#16#1              //№ соединения.
NDR:=#NDR_FLAG          //Получите новые данные
ERROR:= #ERROR_F        //окончание с ошибкой
STATUS:= #STATUS_W      //доп. информация
ADDR_1:=P#I 0.0 BYTE 1  //1 удаленная перем.
ADDR_2:=P#I 4.0 WORD 1  //2 удаленная перем.
ADDR_3:=                //3 удаленная перем.
ADDR_4:=                //4 удаленная перем.
RD_1:=P#Q 0.0 BYTE 1    //1 локальная перем.
RD_2:=P#Q 4.0 WORD 1    //2 локальная перем.
RD_3:=                  //3 локальная перем.
RD_4:=                  //4 локальная перем.
```

LAD представление

DB14 (экземпляр DB)



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.23Information and Training Center
Knowledge for Automation

Краткий обзор

С помощью SFB14 (GET), Вы можете читать данные из удаленного CPU. При положительном фронте на входе управления REQ, задача чтения послана партнеру CPU. Удаленный партнер возвращает данные. Если не произошли никакие ошибки, полученные данные копируются в сконфигурированные приемные области (RD_I) при новом вызове SFB. Завершение работы обозначается состоянием 1 в выходном параметре NDR.

Параметр	Вид	Тип	Назначение
REQ	INPUT	BOOL (I,Q,M,D,L constant)	Активизация при положительном фронте.
ID	INPUT	WORD (I,Q,M,D,L constant)	Обратитесь к таблице соединений для номера соединения.
ADDR_1 ... ADDR_4	IN_OUT	ANY (I,Q,M,D)	Указатели на области в CPU-партнере, которые нужно читать.
RD_1 ... RD_4	IN_OUT	ANY (I,Q,M,D)	Указатель на области в вашем собственном CPU, в котором результаты чтения должны быть сохранены. (Область данных CPU партнера ADDR_i == > RD_i - область данных вашего собственного CPU)
NDR	OUTPUT	BOOL (I,Q,M,D,L)	Положительный фронт сигнализирует программе пользователя, что имеются новые. « Данные перемещены от CPU-партнера без ошибок. »
ERROR	OUTPUT	BOOL (I,Q,M,D,L)	Положительный об ошибке сигнализирует (импульс).
STATUS	OUTPUT	WORD (I,Q,M,D,L)	Содержит детализированное сообщение об ошибке или предупреждение (десятичное число).

SFB коммуникации: блок (SFB 15)

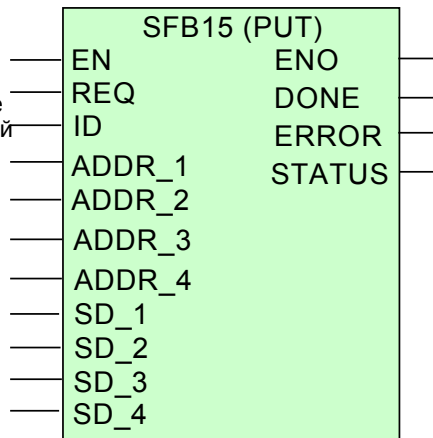
STL представление

с примерами назначения параметров

```
CALL PUT, I_PUT(экземпляр DB)
REQ:=I 0.3           //старт
ID:=W#16#1           //№ соединения.
DONE:= #DONE_F        //Успешное окончание
ERROR:= #ERROR_F      // окончание с ошибкой
STATUS:= #STATUS_W    //инф.о ошибке
ADDR_1:=P#Q 12.0 WORD 1 //1 удаленная перем.
ADDR_2:=              //2 удаленная перем.
ADDR_3:=              //3 удаленная перем.
ADDR_4:=              //4 удаленная перем.
SD_1:=P#I 2.0 WORD 1  //1 локальная перем.
SD_2:=              //2 локальная перем.
SD_3:=              //3 локальная перем.
SD_4:=              //4 локальная перем.
```

LAD представление

DB15 (экземпляр DB)



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.24



Information and Training Center
Knowledge for Automation

Краткий обзор

С помощью SFB15 (PUT), Вы может записывать данные в удаленный CPU. При положительном фронте на входе управления REQ, указатели на области, которые будут записаны (ADDR_i) и данные (SD_i), будут посланы CPU-партнеру. Удаленный партнер сохраняет требуемые данные под адресами, указанными в данных и возвращает подтверждение выполнения.

Параметр	Вид	Тип	Назначение
REQ	INPUT	BOOL (I,Q,M,D,L constant)	Активизация передачи при положительном фронте
ID	INPUT	WORD (I,Q,M,D,L constant)	Обратитесь к таблице соединений за номером соединения.
ADDR_1 ... ADDR_4	IN_OUT	ANY (I,Q,M,D)	Указатели на области данных в CPU-партнере, в которые должны быть записаны данные из посылающегося CPU.
SD_1 ... SD_4	IN_OUT	ANY (I,Q,M,D)	Указатели на области данных в вашем собственном CPU, которые будут посланы CPU-партнеру. (Область данных вашего собственного CPU - SD_1 == > ADDR_1 область данных CPU-партнера)
DONE	OUTPUT	BOOL (I,Q,M,D,L)	Положительный фронт (импульс) сигнализирует программе пользователя: передача закончена без ошибок.
ERROR	OUTPUT	BOOL (I,Q,M,D,L)	Положительный фронт сигнализирует об ошибке (импульс)
STATUS	OUTPUT	WORD (I,Q,M,D,L)	Содержит детализированное сообщение об ошибке или предупреждении (десятичного числа).

SFB коммуникации: блок USEND (SFB 8)

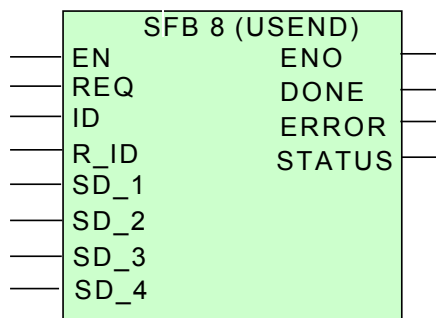
STL представление

с примерами назначения параметров

```
CALL USEND, I_USEND (экземпляр DB)
REQ:= I 0.4           //старт
ID:=W#16#3           //№ соединения.
R_ID:=DW#16#B1       //Блоковая пара
DONE:= #DONE_F       //Успешное окончание
ERROR:= #ERROR_F     //Окончание с ошибкой
STATUS:= #STATUS_W   //Информация об ошибке
SD_1 :=P#DB3.DBX0.0 BYTE 100 //1 локальная переменная
SD_2 :=P#DB3.DBX100.0 BYTE 100 //2 локальная переменная
SD_3 :=P#DB3.DBX200.0 BYTE 100 //3 локальная переменная
SD_4 :=P#DB3.DBX300.0 BYTE 154 //4 локальная переменная
```

LAD представление

DB 8 (экземпляр DB)



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.25



Information and Training Center
Knowledge for Automation

Краткий обзор

SFB8 (USEND) посылает данные удаленному партнеру SFB типа "URCV" (параметр R_ID должен быть идентичен для обоих SFB). Данные посылаются после положительного фронта во входном сигнале управления REQ. Функция выполняется без координации с SFB-партнером. Данные, которые будут посланы, указаны параметрами от SD_1 до SD_4, но не все четыре параметра, должны использоваться.

Параметр	Вид	Тип	Назначение
REQ	INPUT	BOOL (I,Q,M,D,L constant)	Активизация при положительном фронте
ID	INPUT	WORD (I,Q,M,D,L constant)	Номер соединения для S7- соединения (См. таблицу соединений)
R_ID	INPUT	DWORD (I,Q,M,D,L constant)	Параметр должен быть идентичен для SFB USEND и URCV. Назначение блочных пар
DONE	OUTPUT	BOOL (I,Q,M,D,L)	Положительный фронт (импульс) сигнализирует программе пользователя: передача завершилась без ошибки
ERROR	OUTPUT	BOOL (I,Q,M,D,L)	Положительный фронт сигнализирует об ошибке (импульс)
STATUS	OUTPUT	BOOL (I,Q,M,D,L)	Дисплей состояния, если ERROR = 1
SD_1 ... SD_4	IN_OUT	ANY (I,Q,M,D)	Указатели на области данных в вашем собственном CPU, которые будут посланы CPU-партнеру . (Область данных вашего собственного CPU - SD_i == > RD_i область данных CPU-партнера. Должны быть согласованы с соответствующими областями партнера)

SFB коммуникации: блок URCV (SFB 9)

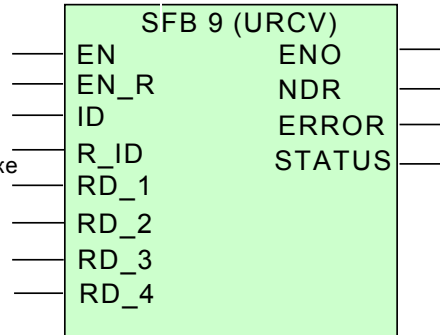
STL представление

с примерами назначения параметров

```
CALL URCV, I_URCV(экземпляр DB)
EN_R:= I 0.5           //старт
ID:= W#16#3           //S7 соединение
R_ID:= DW#16#B1       // Блоковая пара
NDR:= #NDR_F          //Прием новых данных
ERROR:= #ERROR_F      //Окончание с ошибкой
STATUS:= #STATUS_W     // Информация об ошибке
RD_1:=P#DB3.DBX0.0 BYTE 100 //1 локальная переменная
RD_2:=P#DB3.DBX100.0 BYTE 100 //2 локальная переменная
RD_3:=P#DB3.DBX200.0 BYTE 100 //3 локальная переменная
RD_4:=P#DB3.DBX300.0 BYTE 154 //4 локальная переменная
```

LAD представление

DB 9 (экземпляр DB)



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.26



Information and Training Center
Knowledge for Automation

Краткий обзор

SFB9 (URCV) асинхронно получает данные от удаленного партнера SFB "USEND". (Параметр R_ID должен быть идентичен в обоих SFB.) Если значение сигнала 1 появляется на входе управления EN_R, блок вызывается, полученные данные копируются в сконфигурированные области для приема. Эти области данных указаны в параметрах RD_1 для RD_4.

Когда блок вызывается впервые, создается "почтовый ящик для приема". При всех дальнейших вызовах, данные, которые передаются, получает почтовый ящик.

Параметр	Вид	Тип	Назначение
EN_R	INPUT	BOOL (I,Q,M,D,L constant)	Для RLO = 1 полученные данные копируются в сконфигурированные области данных.
ID	INPUT	WORD (I,Q,M,D,L constant)	Номер соединения для S7- соединения (См. таблицу соединений)
R_ID	INPUT	DWORD (I,Q,M,D,L constant)	Параметр должен быть идентичен для обоих SFB (USEND и URCV). Назначение блочных пар
NDR	OUTPUT	BOOL (I,Q,M,D,L)	Положительный фронт (импульс) сигнализирует программе пользователя: переданы новые данные.
ERROR	OUTPUT	BOOL (I,Q,M,D,L)	Положительный фронт = ошибка (импульс)
STATUS	OUTPUT	BOOL (I,Q,M,D,L)	Дисплей состояния, если ERROR = 1
RD_1 ... RD_4	IN_OUT	ANY (I,Q,M,D)	Указатель на области данных в CPU, где полученные данные должны быть сохранены. (SD_i и RD_i должны соответствовать относительно номера, длины, и типа данных.)

SFB коммуникации: блок BSEND (SFB 12)

STL представление

с примерами назначения параметров

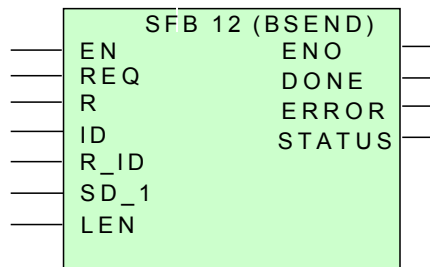
```

CALL BSEND, I_BSEND(экземпляр DB)
REQ:= I 0.4           //старт
R:= I 0.5             //Сброс BSEND
ID:= W#16#3          // S7 соединение
R_ID:= DW#16#B2       // Блоковая пара
DONE:= #DONE_F        //Успешное окончание
ERROR:= #ERROR_F      // Окончание с ошибкой
STATUS:= #STATUS_W    //Доп. информация
SD_1:= P#DB1.DBX0.0 BYTE 40000 //Посылаемые данные
LEN:= #DB_LEN         //Длина данных

```

LAD представление

DB 12 (экземпляр DB)



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.27



Information and Training Center
Knowledge for Automation

Краткий обзор

SFB12 (BSEND) посылает данные удаленному партнеру SFB "BRCV". (Параметр R_ID должен быть идентичен параметру в соответствующем SFB). При этой передаче данных может быть перемещено до 64 Кбайт (применяется к всем CPU). Передача активизируется после вызова блока и когда имеется положительный фронт во входном сигнале управления REQ. Передача данных из памяти пользователя происходит асинхронно по отношению к обработке программы пользователя.

Параметр	Вид	Тип	Назначение
REQ	INPUT	BOOL (I,Q,M,D,L constant)	Активизируется передача при положительном фронте
R	INPUT	BOOL (I,Q,M,D,L constant)	Активизирует сброс BSEND к исходному состоянию при положительном фронте
ID	INPUT	WORD (I,Q,M,D,L constant)	Номер соединения для S7-соединения (см. таблицу соединений)
R_ID	INPUT	DWORD (I,Q,M,D,L)	Параметр должен быть идентичен для обоих SFB (BSEND и BRCV). Назначение блочной пары
SD_1	IN_OUT	ANY (I,Q,M,D,L)	Данные, которые будут посланы, длина в указателе не оценивается
LEN	IN_OUT	WORD (I,Q,M,D,L)	Длина блока данных, который должен быть перемещен
DONE	OUTPUT	BOOL (I,Q,M,D,L)	Сигнализирует положительным фронтом (импульсом) о завершении вызова BSEND без ошибок
ERROR	OUTPUT	BOOL (I,Q,M,D,L)	Положительный фронт сигнализирует об ошибке (импульс)
STATUS	OUTPUT	WORD (I,Q,M,D,L)	Содержит детализированное сообщение об ошибке или предупреждение

SFB коммуникации: блок BRCV (SFB 13)

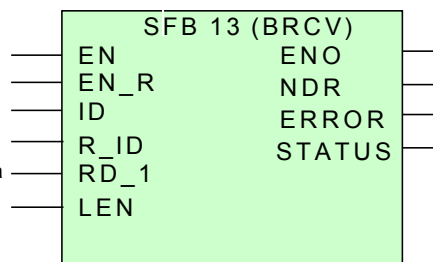
STL представление

с примерами назначения параметров

```
CALL BRCV, I_BRCV(экземпляр DB)
EN_R:= I 0.4           //старт
ID:=W#16#3             // S7 соединение
R_ID:=DW#16#B2         // Блочная пара
NDR:= #NDR_F           // Прием новых данных
ERROR:= #ERROR_F       // Окончание с ошибкой
STATUS:= #STATUS_W     // Доп. информация
RD_1:=P#DB2.DBX0.0 BYTE 40000 //почтовый ящик приема
LEN:= #DB_LEN          //длина почтового ящика
```

LAD представление

DB 13 (экземпляр DB)



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.28



Information and Training Center
Knowledge for Automation

Краткий обзор

SFB13 (BRCV) получает данные от удаленного партнера SFB "BSEND". (Параметр R_ID должен быть идентичен в обоих SFB). После того, как блок вызван и значение 1 появляется на входе управления EN_R, блок готов получить данные. Начальный адрес области приема определен в RD_1. После каждого полученного сегмента данных, посылается подтверждение партнеру SFB и параметр LEN модифицируется. Если блок вызывается вновь в течение асинхронного приема данных, это ведет к предупреждению, выводимому в параметре состояния STATUS; если обращение сделано, когда значение на входе управления EN_R равно 0, прием завершен и SFB возвращается к исходному состоянию. Прием всех сегментов данных без ошибок обозначается параметром состояния NDR, имеющим значение 1.

Параметр	Вид	Тип	Назначение
EN_R	INPUT	BOOL (I,Q,M,D,Lconst.)	RLO = 1 SFB готов для приема RLO = 0 процедура отменена
ID	INPUT	WORD (I,Q,M,D,Lconst.)	Номер соединения S7-соединения (см. таблицу соединений)
R_ID	INPUT	DWORD (I,Q,M,D,Lconst.)	Параметры должны быть идентичны для обоих SFB (BSEND и BRCV). Назначение блочных пар
RD_1	IN_OUT	ANY	Указатель на почтовый ящик для приема. Спецификация длины определяет максимальную длину блока, который будет получен. (2048 слов).
LEN	IN_OUT	WORD	Длина данных, полученных до сих пор (в байтах)
NDR	OUTPUT	BOOL	Положительный фронт сообщает программе пользователя: приняты новые данные
ERROR	OUTPUT	BOOL	Положительный фронт сигнализирует об ошибке (импульс)
STATUS	OUTPUT	WORD	Содержит детализированное сообщение об ошибке или предупреждение

SFB коммуникации: блок STOP (SFB20)

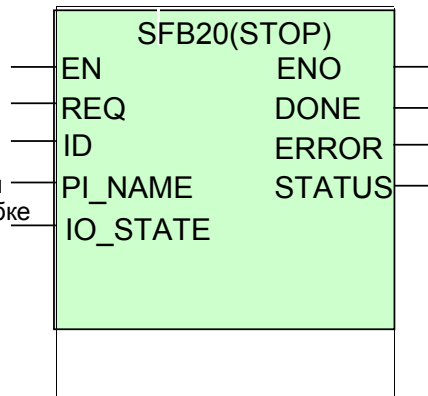
STL представление

с примерами назначения параметров

```
CALL "STOP", "I_STOP" (экземпляр DB)
REQ:= I 0.0           //стартовый фронт
ID:= W#16#1           //№ соединения.
PI_NAME:= P#M100.0 Byte 9 //См. заметку *
IO_STATE:=            //не используется
DONE:= #DONE_F_20     //Успешное окончание
ERROR:= #ERROR_F_20   //Окончание с ошибкой
STATUS:= #STATUS_W_20 //Информация об ошибке
```

LAD/FBD представление

DB20 (экземпляр DB)



* Указатель на начало строки: **"P_PROGRAM"**

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.29



Information and Training Center
Knowledge for Automation

Краткий обзор

Если имеется положительный фронт на входе управления REQ, SFB20 (STOP), активизирует состояние STOP на удаленном устройстве, адресованном ID. Изменение режима возможно, когда устройство находится в RUN, HOLD или режиме запуска (startup). Успешное выполнение работы обозначается 1 в параметре состояния DONE. Если происходят любые ошибки, они обозначены в параметрах состояния ERROR и STATUS. Возобновление изменения режима может быть начато снова в том же самом удаленном устройстве только тогда, когда предыдущий вызов SFB20 был завершен.

Параметр	Вид	Тип	Назначение
REQ	INPUT	BOOL	При положительным фронте активизирует STOP в устройстве, адресованном ID
ID	INPUT	WORD (I,Q,M,D,L, constant)	Обратитесь к таблице соединений (номер соединения).
PI_NAME	IN_OUT	ANY	Указатель на область памяти, в которой размещено имя программы, которая будет начата (код ASCII). Для S7 имя должно быть P_PROGRAM .
IO_STATE	IN_OUT	BYTE	Параметр выполнения (здесь не существенный)
DONE	OUTPUT	BOOL	Положительный фронт = функция выполнена
ERROR	OUTPUT	BOOL	Положительный фронт = ошибка
STATUS	OUTPUT	WORD	Содержит детализированное сообщение об ошибке или предупреждение (десятичное число)

SFB коммуникации: блок START (SFB19)

STL представление

с примерами назначения параметров

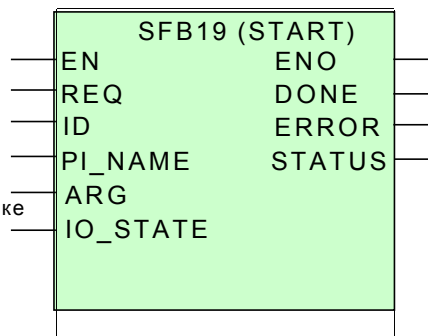
```

CALL "START","I_START" (экземпляр DB)
REQ:= I 0.1           // стартовый фронт
ID:= W#16#1           //№ соединения.
PI_NAME:= P#M100.0 Byte 9 //См. заметку *
ARG:=                 //не используется
IO_STATE:=            //не используется
DONE:= #DONE_F_20     //Успешное окончание
ERROR:= #ERROR_F_20   //Окончание с ошибкой
STATUS:= #STATUS_W_20 //Информация об ошибке

```

LAD/FBD представление

DB19 (экземпляр DB)



* Указатель на начало строки: "P_PROGRAM"

SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.30



Information and Training Center
Knowledge for Automation

Краткий обзор

Если имеется положительный фронт на входе управления REQ, SFB19 (START) активизирует полный рестарт на удаленном устройстве, адресованном ID. Следующие условия должны быть выполнены, если удаленное устройство - CPU :

- CPU должен быть в состоянии STOP.
- кнопочный переключатель CPU должен быть установлен в положение "RUN" или "RUN - P".

Когда полный рестарт завершен и посылается положительное подтверждение выполнения. Когда положительное подтверждение оценено, параметр состояния #DONE установлен в 1. Если происходят, любые ошибки они обозначены в параметрах состояния #ERROR и #STATUS.

Параметр	Вид	Тип	Назначение
REQ	INPUT	BOOL	Активизирует положительным фронтом полный рестарт в устройстве, адресованном ID
ID	INPUT	WORD (I,Q;M,D,L, constant)	Обратитесь к таблице соединений за номером соединения.
PI_NAME	IN_OUT	ANY	Указатель на область памяти, в которой размещено имя программы, которая будет начата (код ASCII). Для S7 имя должно быть P_PROGRAM.
ARG	IN_OUT	ANY	Параметр выполнения (не существенный)
IO_STATE	IN_OUT	ANY	Параметр выполнения (не существенный)
DONE	OUTPUT	BOOL	Положительный фронт = функция выполнена
ERROR	OUTPUT	BOOL	Положительный фронт = ошибка
STATUS	OUTPUT	WORD	Содержит детализированное сообщение об ошибке или предупреждение (десятичное число)

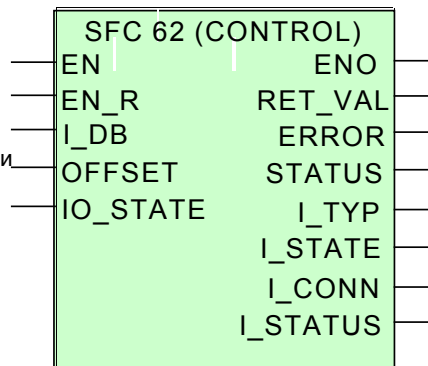
SFB коммуникации: блок CONTROL (SFC 62)

STL представление

с примерами назначения параметров

```
CALL "CONTROL"
EN_R:= I 0. 2           // старт
I_DB:= W#16#F           //№ экземпляра DB
OFFSET:= W#16#0         //для мультиэкземпляров
RET_VAL:= MW4           //Информация об ошибке
ERROR:= Q 0.4          //Окончание с ошибкой
STATUS:= MW 4           //Информация о состоянии
I_TYP:= MB 52           //Тип SFB
I_STATE:= MB 53         //Состояние SFB
I_CONN:= M 54.0         //Состояние соединения
I_STATUS:= MW102        //Статус SFB
```

LAD представление



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.31



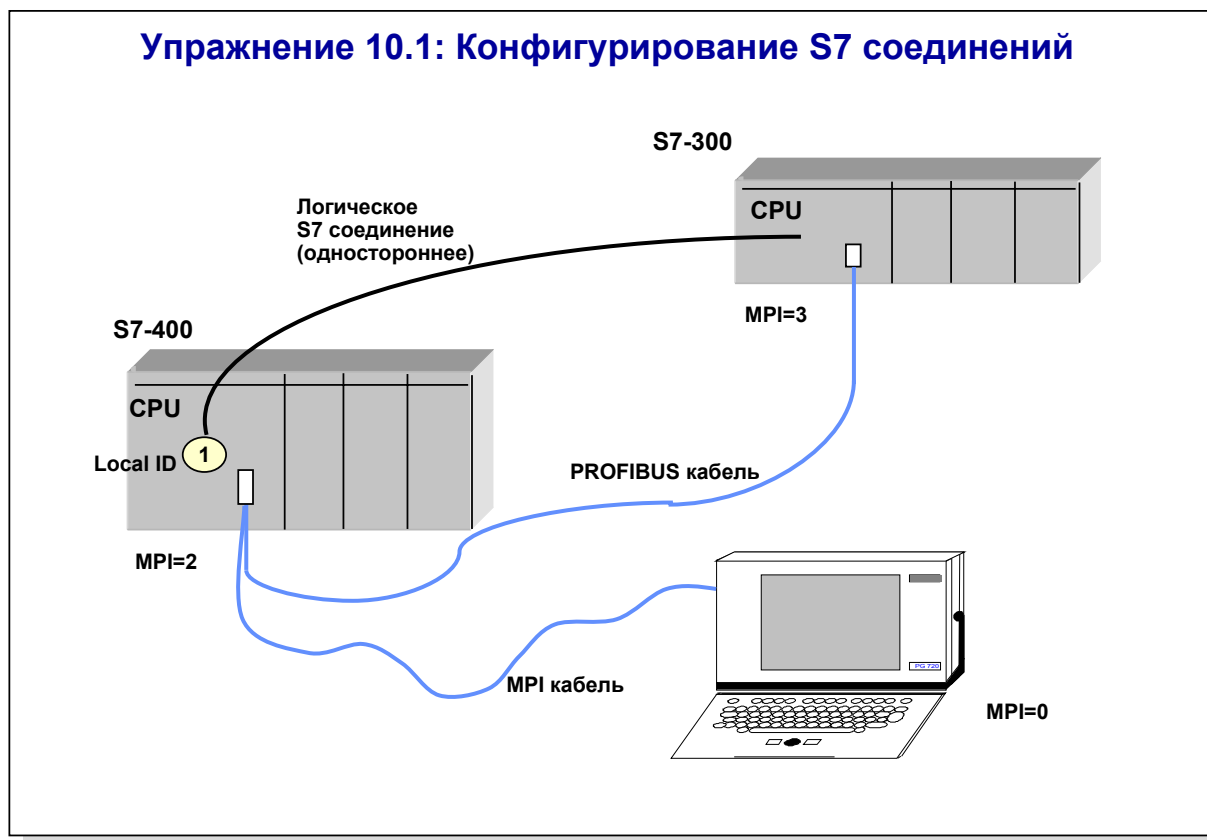
Information and Training Center
Knowledge for Automation

Краткий обзор

С помощью SFC62 "CONTROL" Вы можете сделать запрос состояния соединения и локального экземпляра SFB. После вызова системной функции со значением 1 на входе сигнала управления EN_R делается запрос текущего состояния соединения и SFB, выбранного с помощью I_DB.

Параметр	Вид	Тип	Назначение
EN_R	INPUT	BOOL	Параметр управления для запуска функции
I_DB	INPUT	BLOCK_DB (I,Q;M,D,L, constant)	Номер экземпляра DB
OFFSET	INPUT	WORD (I,Q;M,D,L, constant)	Смещение для мультиэкземпляров, 1-ый номер байта экземпляра DB (Если не мультиэкземпляр= 0)
RET_VAL	OUTPUT	INT (I,Q;M,D,L)	8000H ошибка для SFC62
ERROR	OUTPUT	BOOL (I,Q;M,D,L)	RLO = 1 ошибка в течение выполнения SF C62
STATUS	OUTPUT	WORD (I,Q;M,D,L)	Показ ошибки для SFC 62
I_TYP	OUTPUT	BYTE (I,Q;M,D,L)	Идентификатор типа коммуникационного SFB
I_STATE	OUTPUT	BYTE (I,Q;M,D,L)	Идентификатор текущего графа состояния коммуникационного SFB
I_CONN	OUTPUT	BOOL (I,Q;M,D,L)	Состояние соответствующего соединения 0 = соединение разорвано 1 = соединение имеется в наличии
I_STATUS	OUTPUT	WORD (I,Q;M,D,L)	Параметр STATUS опрашиваемого SFB

Упражнение 10.1: Конфигурирование S7 соединений



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.32



Information and Training Center
Knowledge for Automation

Задача

Создание сети из двух CPU S7-400 и S7-300 и конфигурирование S7-соединения.

Что делать

1. Создать новый проект "SFB-Comm".
2. Создать в вашем проекте две HW станции для S7-400 и S7-300.
3. В *HW Config* определить различные MPI-адреса для двух CPU и "сеть" из двух CPU с общим объектом "MPI Network" в вашем проекте.
3. Затем загрузите данные конфигурации в индивидуальные CPU, пользуясь инструментом *HW Config*.
4. Соединить станции через MPI кабелем и проверить результат, используя функцию PG: "Accessible Nodes".
5. Конфигурировать S7 Соединение между двумя CPU загрузить скомпилированную таблицу соединений в S7-400-CPU.
6. Используя опций меню *PLC -> Module Information*, проверить, было ли соединение фактически зарезервировано в S7-400-CPU (Закладка: *Communication -> Reserved Connections*)
7. Провести полный рестарт S7-400.
8. Проверить, было ли установлено зарезервированное соединение. Для этого, прочтите Online информацию о состоянии S7-400 CPU, используя опцию меню *PLC -> Module Information*. Затем проверьте закладку *Communication* - было ли установлено зарезервированное соединение.

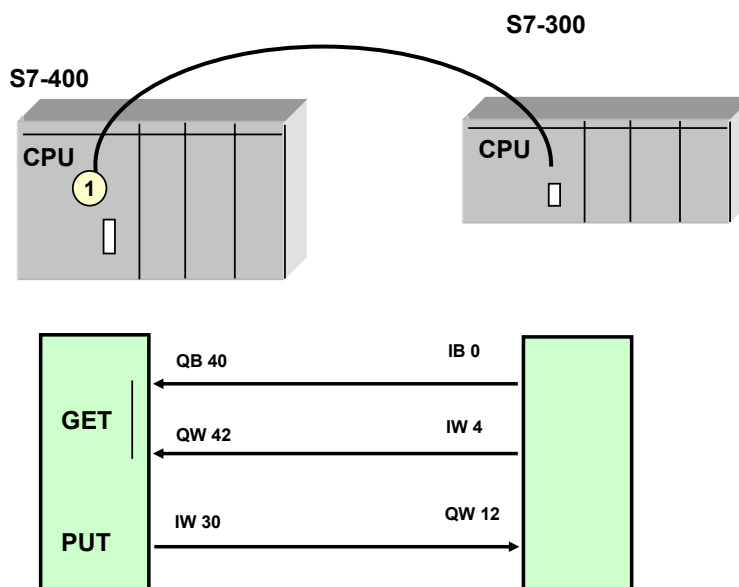
Обратите внимание S7-300 не имеет никаких данных конфигурации и Online-данных, которые бы дали бы информацию относительно зарезервированных и фактически используемых связей.

Упражнение 10.2: Коммуникации с SFB GET/PUT

Программа в S7-400

OB 1

```
CALL SFB14,DB14
REQ=I 0.0
ID:=W#16#1
.
CALL SFB 15,DB15
REQ=I 0.1
ID:=W#16#1
.
.
```



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.33



Information and Training Center
Knowledge for Automation

Задача

Для S7-400 создайте OB1 со следующими функциональными возможностями:

- через вход I28.0, IB0 и IW 4 из S7-300 нужно записать в QB40 или QW42 в S7-400.
- через вход I28.1, IW30 из S7-400 нужно записать в QW12 в S7-300.

Что делать

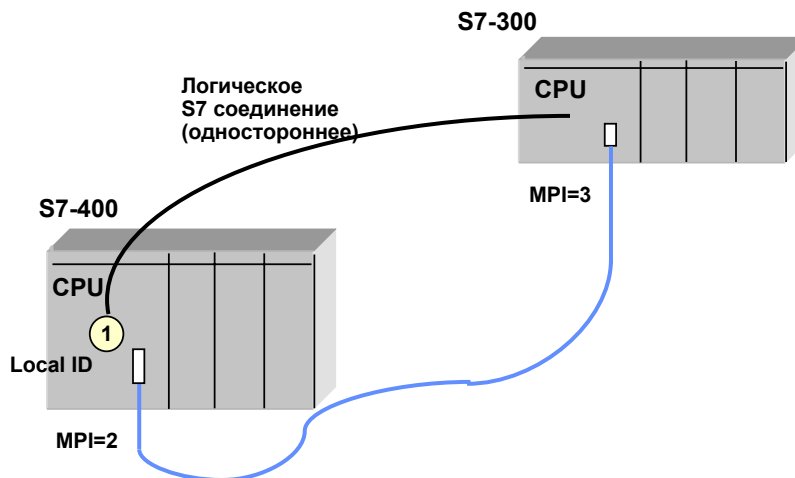
1. Создайте папку S7 Program с именем SFB_GET_PUT.
2. Редактируйте OB1. Создайте сеть (network) "SFB_GET", в которой вызовите SFB, "GET" (переключатель I28. 0).
При вызове "GET" читается IB0 из S7-300 и записывается в QB40 S7-400. Также читайте содержание IW4, и выведите это к QW42 S7-400.
3. Создайте сеть "SFB_PUT" и вызовите SFB "PUT" (переключатель I28.1).
При вызове "PUT" передают IW2 из S7-400 в QW12 S7-300.
4. Передать выходной параметр STATUS (импульс) SFB на цифровой дисплей (QW38) S7-400.
5. Загрузить OB1 в S7-400 CPU и проверьте вашу программу.

Упражнение 10.3: Коммуникации с SFB START/STOP

Программа в S7-400

OB 1

```
CALL SFB20,DB20
REQ= I 28.0
ID:=W#16#1
PI_NAME:= P#M100.0 Byte 9
-----
CALL SFB 19,DB19
REQ=I 28.1
ID:=W#16#1
PI_NAME:= P#M100.0 Byte 9
.
.
.
```



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_10E.34



Information and Training Center
Knowledge for Automation

Задача

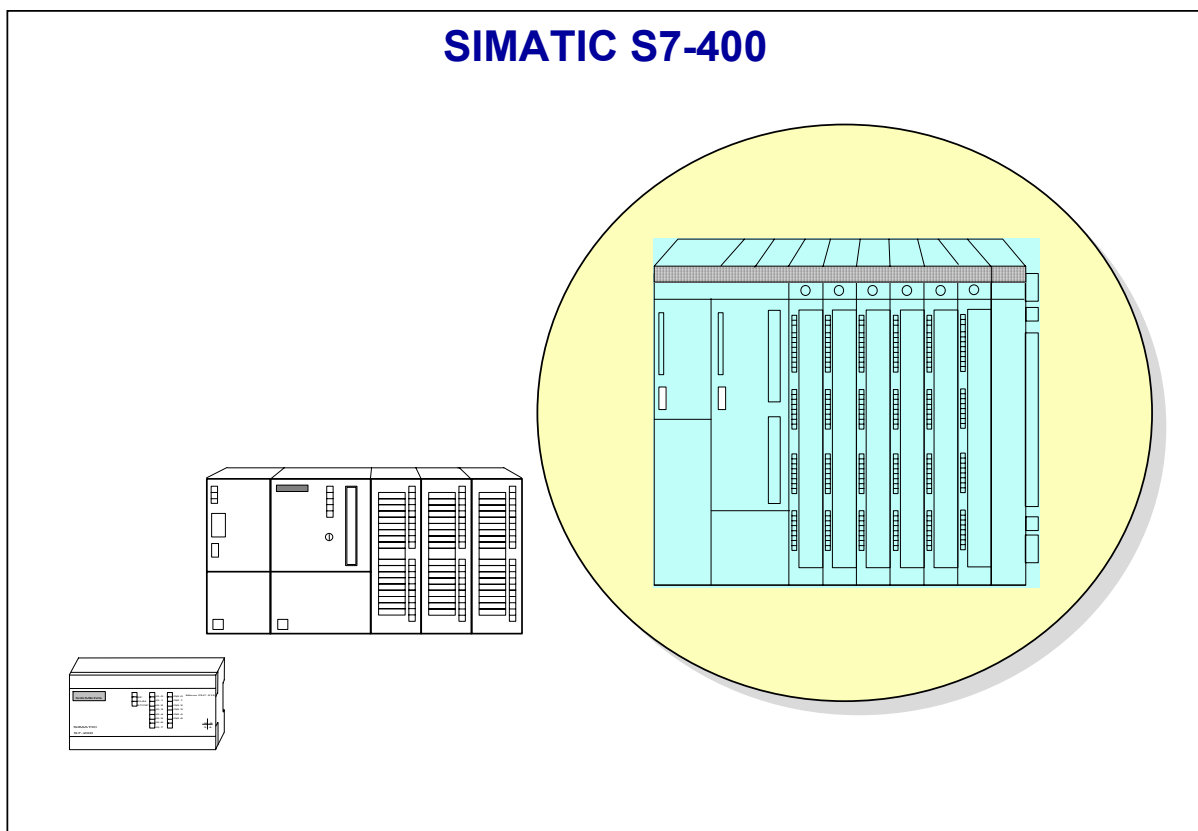
Для S7-400 создайте OB1 со следующими функциональными возможностями:

- удаленный партнер (S7-300) может быть "остановлен" через вход I28.0
- удаленный партнер может быть "запущен" через вход I28.1.

Что делать

1. Создайте папку S7 Program с именем "SFB_START_STOP"
2. Редактируйте OB1. Создайте сеть "P_PROGRAM", в которой Вы сохраняете строку "P_PROGRAM" в меркерах от MB100 до MB109.
3. Создайте сеть "SFB_STOP", в которой Вы вызываете SFB "STOP" (переключатель I28.2).
4. Генерируйте сеть "SFB_START", в которой Вы вызываете SFB "START" (переключатель I28.3).
5. Передайте выходной параметр STATUS (импульс) SFB на цифровой дисплей (QW38) S7-400
6. Загрузите OB1 в S7-400-CPU и проверьте вашу программу.

SIMATIC S7-400



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.1Information and Training Center
Knowledge for Automation

Содержание

Стр

SIMATIC S7-400: Обзор	2
S7-400 обзор модулей	3
Стойки для установки модулей (корзины - racks) S7 - 400	4
Симметричный и асимметричный мультикомпьютинг	5
Централизованная конфигурация	6
Параметры модулей: логический адрес, часть изображения процесса (Part Process Image)	7
Назначение параметров модулей: аналоговые модули	8
Конфигурация мультикомпьютинга	9
SFC 35 для синхронизации в мультикомпьютерном режиме	10
Централизованное расширение 1	11
Централизованное расширение 2	12
Распределенное расширение	13
Распределенное расширение между S7 и S5	14
Расширение централизованной конфигурации	15
Модули CPU	16
Технические данные CPU S7-400 (1)	17
Технические данные CPU S7-400(2)	18
Системная архитектура	19
Параметры CPU : характеристики запуска	20
Параметры CPU: прерывания	21
Параметры CPU: локальные данные	22
Параметры CPU: концепция защиты	23
Организация программы: полный рестарт и рестарт	24
Режим "Force" в S7-400	26
Активизация точек останова	27
Выполнение программы с точками останова	28
Доступ к периферийным выходам	29
CP 441 для соединений точка к точке	30
CP 443-5: соединение для PROFIBUS	31
IM 467: интерфейс PROFIBUS-DP мастер	32
CP 443-1: соединение с Industrial Ethernet	33
CP 443-1 IT: подключение к Internet	34

SIMATIC S7-400: Обзор

Расширяемость системы

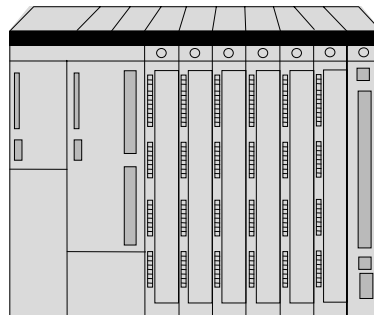
- Высокая плотность монтажа
- Дипломированная эффективность CPU
- Мультикомпьютинг - многопроцессорная обработка
- Может быть подключено до 21 корзины расширения
- Широкий диапазон модулей (SM, FM, CP)
- Гибкие средства работы с сетями

Эффективность

- Высокое быстродействие процессора (до 80 нс на двоичную команду)
- Пользовательская память до 20 Мбайт
- Мощные средства связи

Универсальность

- Специальные функции
- Специальные средства встраивания S5



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.2



Information and Training Center
Knowledge for Automation

SIMATIC S7-400

PLC SIMATIC S7-400 соответствует среднему и верхнему уровням управления. Модульный, без вентилятора, обладающий высокой расширяемостью и прочностью, большими возможностями по связи и высокой эффективностью - все это делает данный PLC подходящим для требования разнообразных проектов.

Расширяемость

Специальные характеристики S7-400:

- Простой модуль, выполненный по вибростойкой технологии. Все модули имеют могут работать без вентилятора и имеют высокую плотность монтажа. По сравнению с S5 пространство монтажа уменьшено на 54 %, пространство на подключение входа - выхода уменьшено на 45 %.
- S7-400 предлагает масштабируемость благодаря спектру доступных модулей CPU, а также симметричную и асимметричную возможность обработки данных в многопроцессорной системе.
- широкое разнообразие модулей, то есть для каждой прикладной программы существуют подходящие CPU, сигнальные модули, функциональные модули и модули связи.
- Расширяемый максимум 21стойкой расширения и дополнительными распределенными модулями через PROFIBUS-DP
- Возможности работы в сети через MPI, PROFIBUS и Industrial Ethernet делают S7-400 удобным для задач эффективного управления процессом.

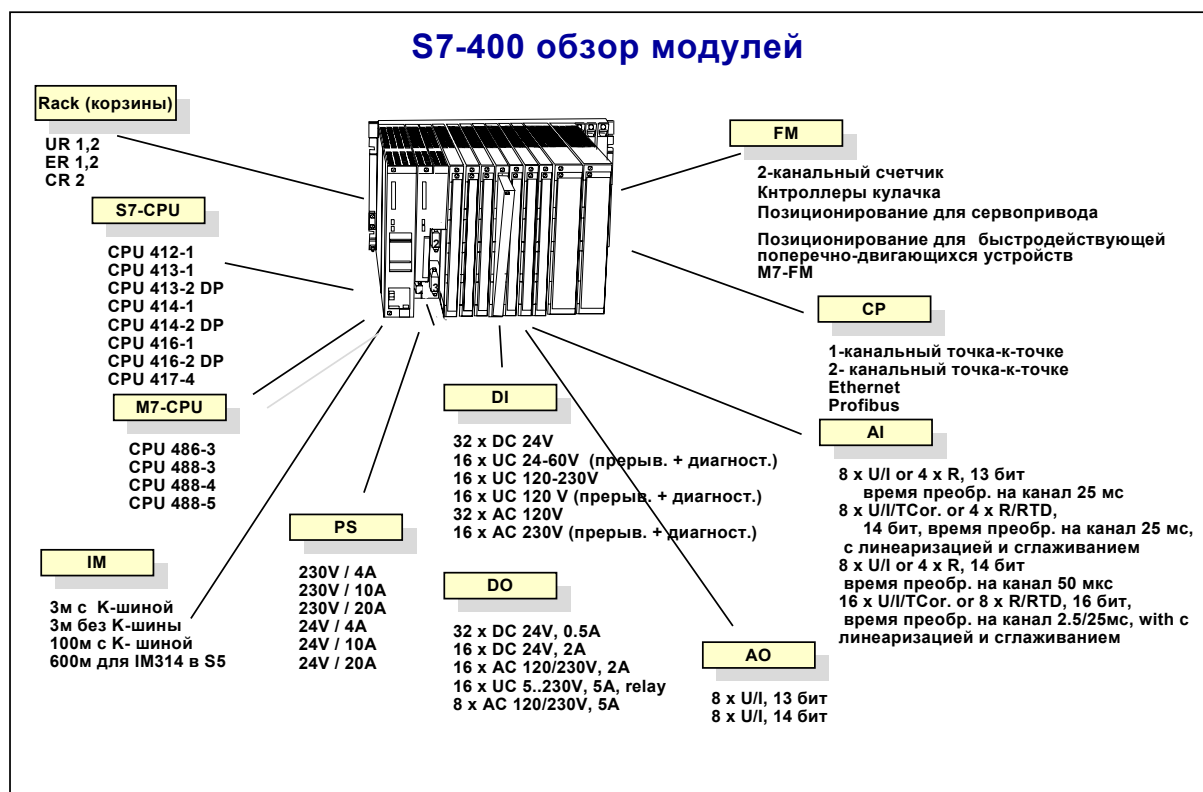
Эффективность

- Высокая скорость обработки (80 ns на команду) и пользовательская память до 1.6 Мб делают возможной реализацию сложных задач автоматизации.
- Высокая эффективность связи по шине (10.5 Мбод) гарантирует быструю связь с высокой пропускной способностью данных.

Универсальность

- Универсальность благодаря специальным функциям типа: перезапуск, удаление и вставка модуля в режиме RUN и т.д.
- Кроме того, S7-400 предлагает специальные возможности перехода от S5 до S7 типа:
 - использование S5 IPS или WFS в S7 центральных стойках
 - подключение S5-модулей расширения к центральной стойке S7.

S7-400 обзор модулей



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.3Information and Training Center
Knowledge for Automation

Стойки

Следующие стойки доступны для S7-400:

- UR1/UR2 разработана как универсальная стойка и может использоваться как центральная стойка или как стойка расширения. Она содержит слоты одиночной ширины - 18/9 штук, а также Р- и К- шины.
- ER1/ER2 - стойки расширения без К-шины.
- CR2 - сегментированная центральная стойка для симметрической обработки данных в многопроцессорной системе.

S7 CPU:

S7-400 CPU совместимы сверху с STEP 7-программами пользователя. Они существуют в двух версиях: одиночная ширина и двойная ширина с интегрированным интерфейсом Мастера DP. Максимум 64 DP-управляемые станции может быть адресован через интегрированный интерфейс DP. Максимальная скорость в бодах - 12 Мбод.

FM

FM для позиционирования, управления с обратной связью(замкнутого контура управления) и счета заменяют спектр S5 IP . Кроме того, M7 FM может быть вставлен как свободно программируемый на языке С модуль для управления процессом.

IM

Стойки расширения SIMATIC S7 и SIMATIC S5 могут быть связаны с S7-400 центральной стойкой через интерфейсные модули.

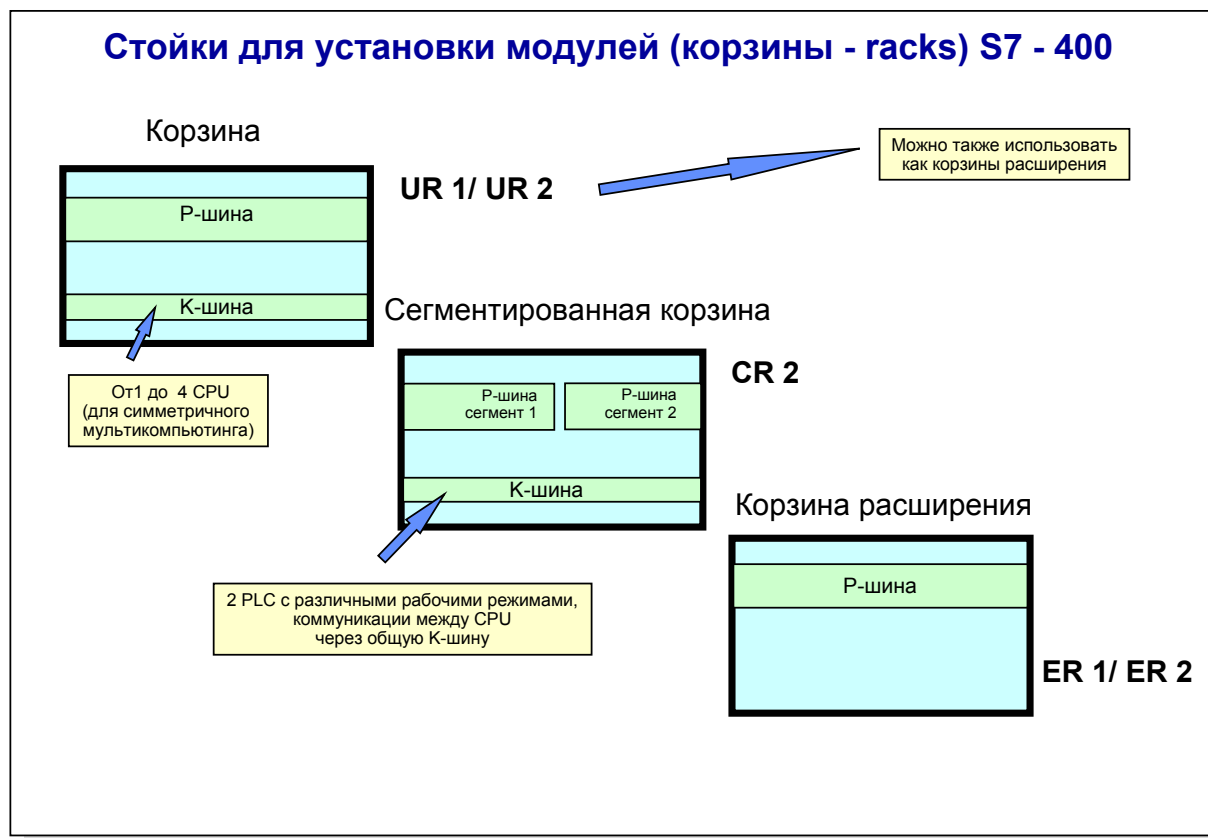
CP

Модули CP делают возможным подключение CPU к следующим сетям:

- Industrial Ethernet (CP 443-1)
- PROFIBUS (CP 443-5)
- Сеть точка-к точке (PtP) (CP441-1 и CP441-2)

Кроме того, каждый CPU имеет интерфейс MPI для подключения к MPI сети. Максимум 32 станции может быть связано MPI сетью.

Стойки для установки модулей (корзины - racks) S7 - 400



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.4



Information and Training Center
Knowledge for Automation

UR 1 / UR 2

UR1/UR2 может использоваться и как центральная стойка и как стойка расширения. Они имеют параллельную I/O шину (Р-шина) для быстрого обмена сигналами ввода - вывода (1.5 мкс/ байтом) и критичного по времени доступа к сигнальным модулям.

Кроме того, UR1 (18 слотов) / UR2 (9 слотов) имеют последовательную, мощную коммуникационную шину (К шину) для быстродействующего обмена данными (10.5 Мбод) между узлами К-шины (S7/M7 CPU, FM, CP).

Разделяя Р-шину и К-шину, каждой задаче назначается собственная система шин. Управление и связь имеют собственные отдельные скорости передачи данных, таким образом обеспечивая гладкое и бесконфликтное управление и действие связи.

CR2

Сегментированная стойка CR 2 - шина ввода - вывода, разделенная в два сегмента с 10 и 8 слотами. Один CPU может использоваться в каждом сегменте. Каждый CPU - мастер в соответствующей Р-шине и может обращаться только к собственным сигнальным модулям.

Переходы рабочего режима не синхронизированы, то есть CPU могут быть в различных рабочих режимах. Оба CPU могут связываться через непрерывную К-шину.

ER 1 / ER 2

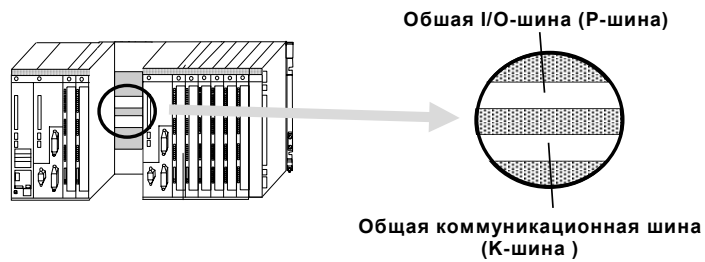
ER1 (18 слотов) /ER2 (9 слотов) не имеют К-шины и линий прерываний, нет питания 24V для модулей и питания от батарей.

Нет правил для слотов

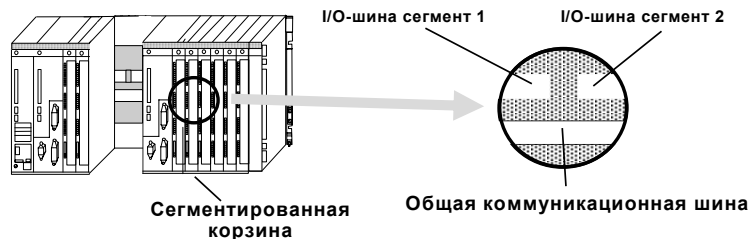
Исключение: PS стоит самым левым, а приемный IM в ER - самым правым.

Симметричный и асимметричный мультикомпьютинг

• Симметричный мультикомпьютинг



• асимметричный мультикомпьютинг



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.5



Information and Training Center
Knowledge for Automation

Мультикомпьютинг Обработка данных в многопроцессорной системе (мультикомпьютинг) делает систему PLC масштабируемой, то есть это делает систему более эффективной и увеличивает ее ресурсы, то есть память, память меркеров, таймеры, счетчики и т.д. Сложная задача, например, может быть разделена между отдельными CPU.

Симметричный При симметричной обработке данных в многопроцессорной системе все CPU (максимально 4 CPU) совместно используют общие P- и K-шины. В частности, существует одно общее адресное пространство ввода - вывода, в котором отображены адреса всех сигнальных модулей.

Каждый вставленный модуль должен однако быть назначен при конфигурировании CPU. CPU затем выполняет "Функции Мастера" для этого модуля, типа:

- получение прерываний от модуля
- назначение параметров модулю
- доступ к модулю через L PBxx, T Wxx и т.д.

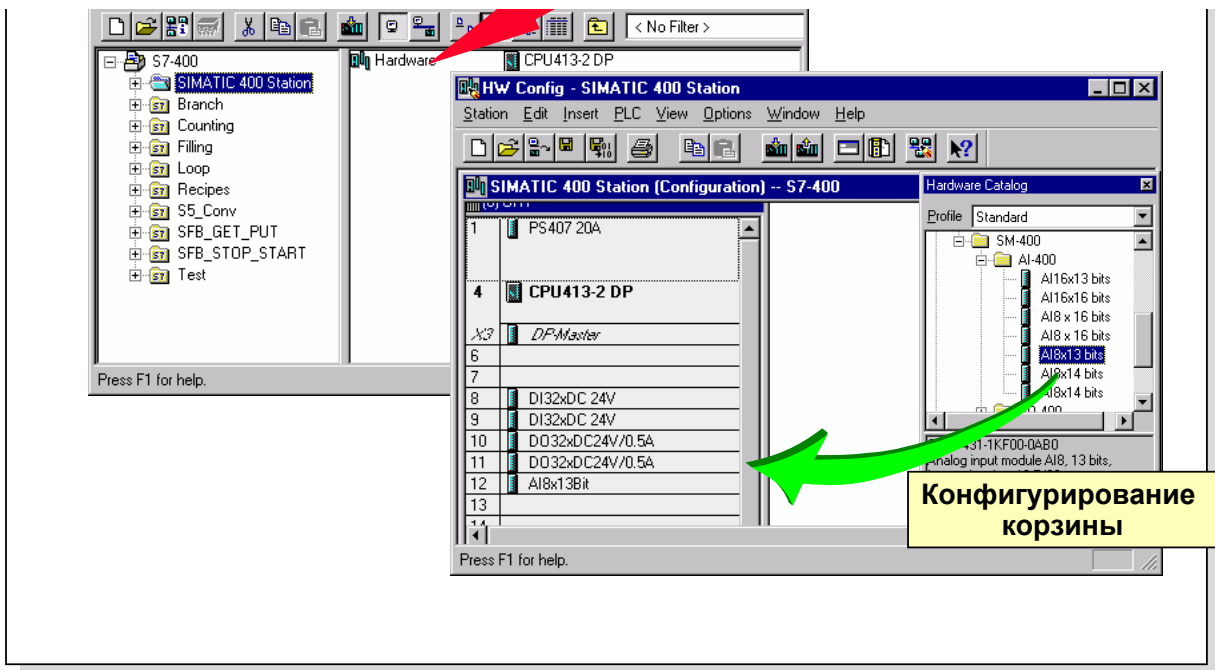
Переходы рабочего режима синхронизированы, то есть все CPU имеют один и тот же рабочий режим. Снаружи кажется, что станция представляет один большой PLC.

Асимметричный Асимметричная обработка данных в многопроцессорной системе достигается с помощью CR2. Сегментированная стойка содержит два отдельных сегмента P-шины.

Один CPU установлен на сегмент шины ввода - вывода. Модули ввода - вывода локально назначены этому CPU. CPU работают независимо друг от друга без синхронизации переходов рабочего режима. Каждый CPU имеет собственное адресное пространство ввода - вывода.

Общая шина связи делает связь между двумя модулями возможной без использования дополнительных аппаратных средств. Снаружи кажется, что это соответствует двум индивидуальным контроллерам, которые связаны через K-шину. Дальнейшие преимущества:

- space saving in the control cabinet пространство, сохраняющееся в управляющем кабинете
- экономия денег, так как требуется только одна стойка и один источник питания.



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.6



Information and Training Center
Knowledge for Automation

Конфигурирование централизованной станции

Для централизованной конфигурации упорядочьте модули рядом с CPU на центральной стойке и продвигайтесь к стойками расширения.

Создание конфигурации



Чтобы создать конфигурацию станции поступайте следующим образом:

1. Прежде всего кликните мышью на желаемую аппаратную станцию.
2. Выберите меню *Edit -> Object*, откройте или дважды щелкните на символ *Hardware* в правом окне. Окно выбранной станции открыто.
3. Щелчок на символе Каталога отображает HW Catalog с текущими компонентами. Из каталога перетащите методом "drag and drop" стойку (Rack) и модули в окно станции или в таблицу конфигурации соответствующей стойки.

Обзор каталога

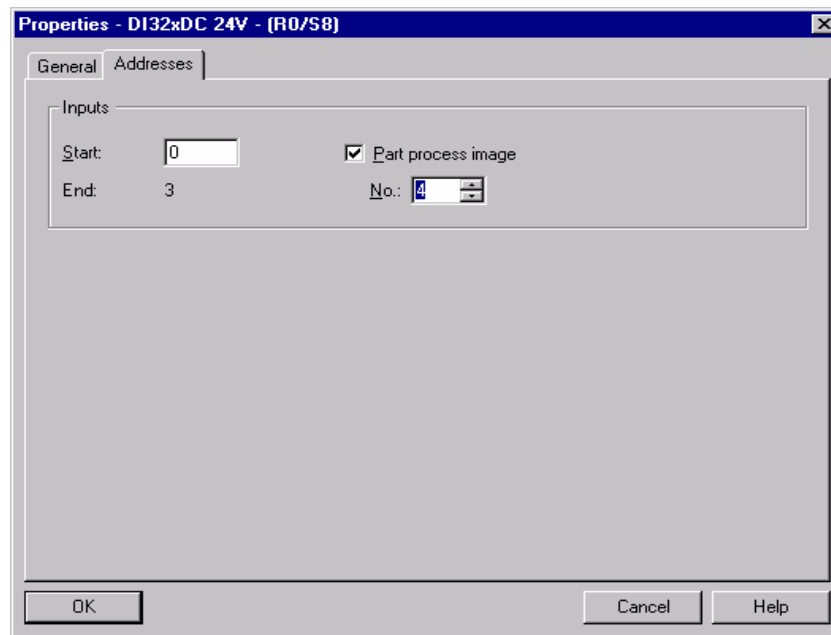
Щелчок на знак "+" открывает связанную подструктуру, щелчок на знак "-" - закрывает. После выбора модуля его наиболее важные технические данные появляются в строке состояния окна каталога.

Выбор стойки

Когда Вы открыли окно станции и аппаратный каталог, Вы можете продолжать следующим образом:

1. В зависимости от типа станции выберите раздел S7-400.
2. Прежде всего откройте подраздел RACK и перетащите стойку в окно станции слева. Пустая таблица отображается для каждой стойки. Стойки синхронно представляются таблицами конфигурации в STEP 7. Эти таблицы конфигурации имеют ряд установочных линий с номерами, равными номеру модулей, которые могут быть установлены в стойку.
3. Затем, используя метод "drag and drop", копируйте желаемые модули налево в пустые слоты в таблице.

Параметры модулей: логический адрес, часть изображения процесса (Part Process Image)



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.7



Information and Training Center
Knowledge for Automation

Общий

Система S7 -400 имеет заданную по умолчанию адресацию для модулей ввода - вывода. Эти значения по умолчанию действительны, пока другие параметры не загружены в CPU.

Адресация по умолчанию

Значения адресов модулей по умолчанию зависят от:

- номера стойки. Номер установлен на приемном IM переключателем (1 .. 21), центральная стойка всегда имеет номер 0

- слота модуля в стойке. Заданный по умолчанию адрес модуля рассчитывается из этих двух значений следующим образом:

Начальный адрес цифрового модуля = [(номер стойки) x 18 + номер слота - 1] x 4

Начальный адрес аналогового модуля = [(номер стойки) x 18 + номер слота - 1] x 64 + 512

Part Process Image

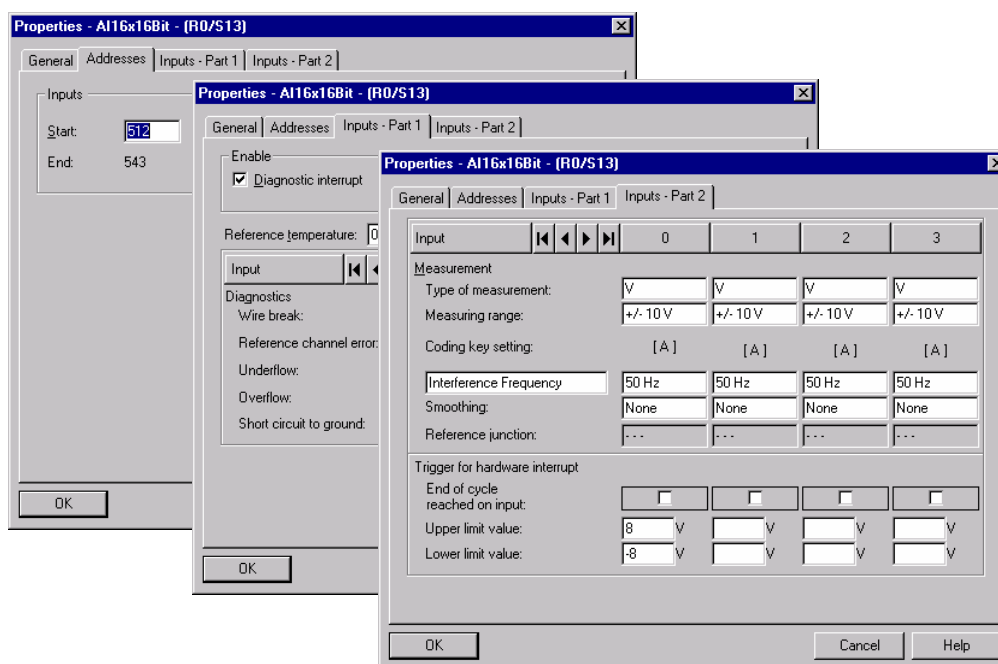
Помимо полного изображения процесса, Вы можете назначать параметры, определяющие до 8 частей изображений процесса. Части изображения процесса могут модифицироваться в программе пользователя с помощью системных функций (SFC 26/27).

Таким образом, пользователь имеет возможность использовать концепцию изображения процесса, которая была первоначально разработана только для класса приоритета OB1, для других классов приоритета, например OB35 для алгоритмов управления.

Внутри OB35 Вы можете затем поступать следующим образом:

1. Считывание текущих входных значений с помощью SFC26 в таблицу, связанную с частью изображения входного процесса.
2. Вызов алгоритма управления. Алгоритм управления пишет новые значения в часть соответствующей таблицы изображения выходного процесса.
3. Вывод части таблицы изображения выходного процесса с помощью SFC27 на выходы.

Назначение параметров модулей: аналоговые модули



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.8



Information and Training Center
Knowledge for Automation

Назначение параметров модуля

Всем модули с назначаемыми параметрами, например аналоговые модули, параметры могут назначаться с помощью инструмента HW Config. Для аналоговых модулей имеется обычно отдельная закладка для назначения параметров. Таким образом, например для аналогового модуля S7-400 могут быть установлены на индивидуальной закладке следующие параметры:

Addresses

- Начальный адрес модуля
- Номер части изображения процесса
- ОВ аппаратного прерывания

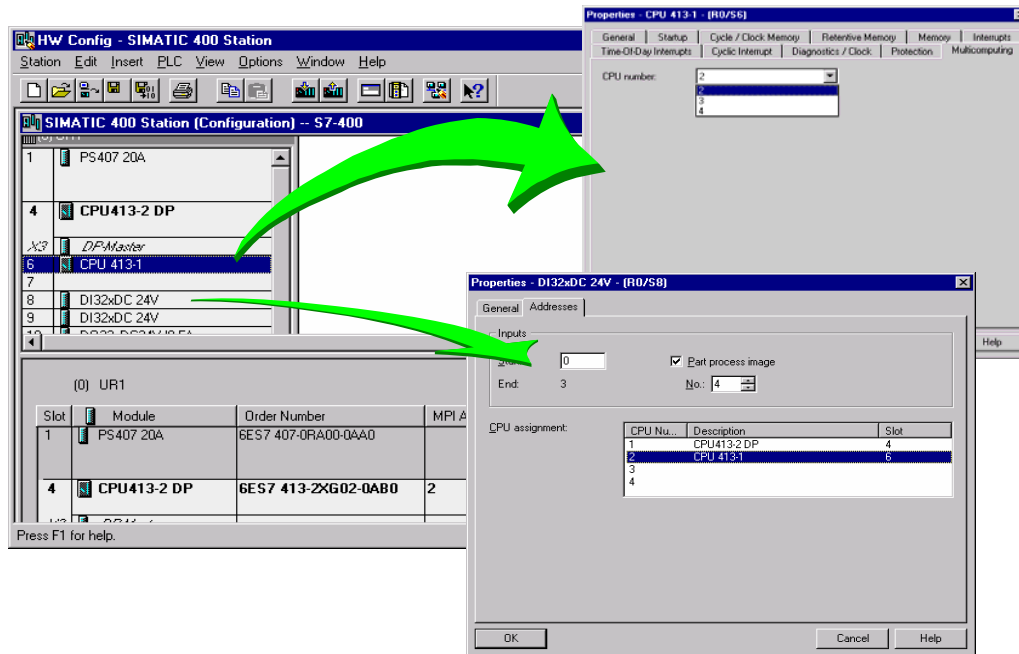
Inputs - Part 1

- Разрешение для аппаратного и диагностического прерываний
- Разрешение специфического для канала диагностики:
 - Проверка обрыва провода (Wire break check)
 - Ошибка канала ссылки (Reference channel error)
 - Выход за нижнюю границу (Underflow)
 - Переполнение (Overflow)
 - Короткое замыкание (Short circuit to ground)

Inputs - Part 2

- Тип измерения (Type of measurement)
- Диапазон измерений (Measuring range)
- Подавление частоты помехи (Interference frequency suppression)
- Сглаживание (Smoothing)
- Верхняя и нижняя предельная величина для аппаратного прерывания (Upper and lower limit value for hardware interrupt)

Конфигурация мультимикомпьютинга



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.9



Information and Training Center
Knowledge for Automation

Краткий обзор

Обработка данных в многопроцессорной системе (мультимикомпьютинг) - синхронный режим отдельных CPU S7-400 (от 2 до 4) в центральной стойке. CPU вместе запускают, если они имеют один и тот же режим запуска (полный рестарт или рестарт) и они также вместе входят в режим STOP.

Установка

мультимикомпьютинга

Мультимикомпьютинг организуется вставкой отдельных CPU, способных к обработке данных в многопроцессорной системе. Стойка и CPU должны быть способными к обработке данных в многопроцессорной системе. Это может быть определено в информационном тексте в "Hardware Catalog".

Общая область адресов разделена между CPU, участвующими в обработке данных в многопроцессорной системе, то есть адрес модуля всегда исключительно связывается с одним CPU.

Процедура

Мультимикомпьютинг конфигурируется следующим образом:

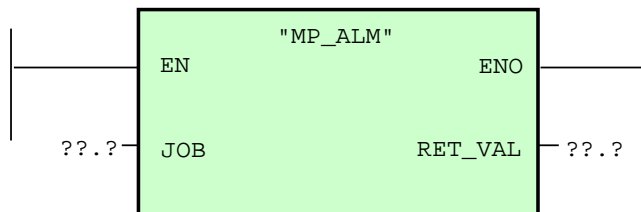
1. Установите все CPU, требуемые для мультимикомпьютинга.
2. Дважды щелкните по CPU и установите номер CPU на закладке "Multicomputing".
3. Чтобы назначить модуль отдельному CPU, приступайте следующим образом:
 - Установите модули в стойке.
 - Дважды щелкают по модулю и выбирают закладку "Addresses"
 - В поле "CPU No." выбирают номер нужного CPU.

В модулях с прерываниями назначенный CPU отображается как целевой CPU на закладках "Inputs" или "Outputs".

Через команду View -> Filter -> CPUx Modules, Вы можете выделять в таблице модули, которые назначены специфическому CPU.

Конфигурация станции может быть загружена только во все CPU. Загрузка только в один CPU не возможна. Таким образом избегают несовместимых конфигураций.

SFC 35 для синхронизации в мультикомпьютерном режиме



Параметр	Объявление	Тип данных	Область памяти	Описание
JOB	INPUT	BYTE	I, Q, M, D, L, Const.	Идентификатор задания (допустимые значения: от 1 до 15)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.10Information and Training Center
Knowledge for Automation

Описание

Вызов SFC 35 "MP_ALM" запускает прерывание обработки данных в многопроцессорной системе. Это ведет к синхронизированному началу OB60 на всех связанных CPU.

В однопроцессорной конфигурации и в сегментированных стойках, OB60 стартует только на CPU, который вызвал SFC 35.

Во входном параметре JOB Вы можете идентифицировать причину для прерывания обработки данных в многопроцессорной системе. Этот идентификатор работы - передается всем связанным CPU и может быть оценен в стартовой информации OB 60.

Вы можете вызывать SFC 35 "MP_ALM" из любого места в вашей программе. Так как обращение имеет смысл только в режиме RUN, прерывание обработки данных в многопроцессорной системе подавлено, когда вызван режим STARTUP. Это сообщается Вам в возвращаемом значении.

Код ошибки

Если происходит ошибка во время вызова функции, возвращаемое значение содержит код ошибки:

W#16#0000: Нет ошибок.

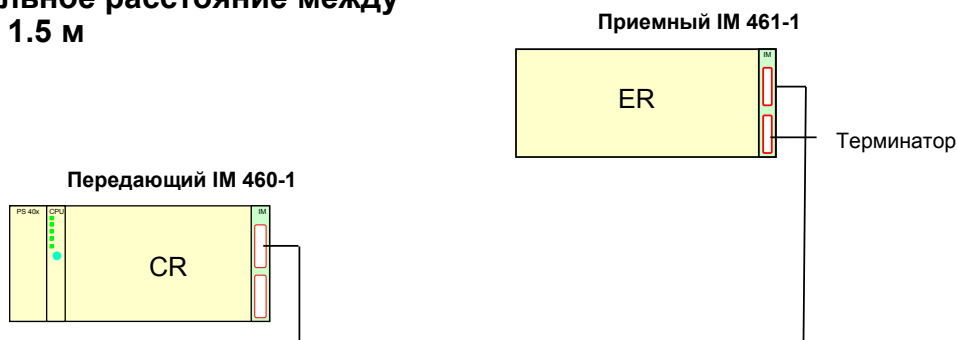
W#16#8090: Входной параметр JOB содержит недопустимое значение.

W#16#80A0: Обработка предыдущего мультикомпьютерного прерывания в OB 60 еще не была завершена на локальном CPU или на другом CPU.

W#16#80A1: Неправильный рабочий режим (STARTUP вместо RUN).

Централизованное расширение 1

- Р-шина и источник питания соединены, но не через К-шину
- 1 ER в цепочке
- Максимальное расстояние между CR и ER: 1.5 м



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.11



Information and Training Center
Knowledge for Automation

Централизованная конфигурация 1

Интерфейсные модули IM 460-1/IM 461-1 предназначены для централизованного подключения стоек расширения к центральной стойке. Централизованный тип расширения 1 имеет следующие характеристики:

- Максимум 2 стойки расширения могут быть связаны (1 на интерфейс).
- Максимальное расстояние между центральной стойкой и стойкой расширения - 1.5 м.
- Максимум 2 передающих IM 460-1 может быть вставлено на центральную стойку.
- Передающий интерфейсный модуль IM 460-1 подключает через стойку расширения только Р-шине (но не К-шину). Кроме того, модули в стойке расширения через соединительный кабель обеспечены напряжением 5V (максимально 5А на слот).

По этой причине, никаких дополнительных источников питания не нужно вставлять в стойку расширения.

- Незанятый интерфейс в передающем IM 460-1 не должен быть нагружен на терминатор; незанятый интерфейс в приемном IM 461-1 должен быть нагружен на терминатор.
- Переключатель, которым выбирается номер стойки расширения, размещен на приемном IM 461-1.

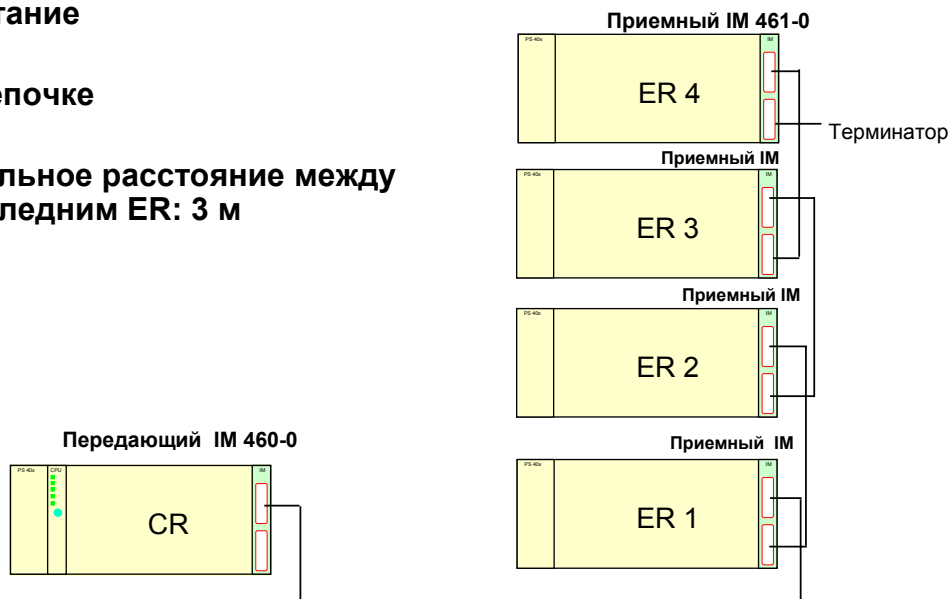
Обратите внимание

Максимум 21 стойка расширения может быть связана с одной центральной стойкой.

Принимающий IM должен всегда вставляться в самый дальний правый слот в стойке расширения.

Централизованное расширение 2

- Соединены Р-шина и К-шина, но не питание
- 4 ER в цепочке
- Максимальное расстояние между CR и последним ER: 3 м



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.12



Information and Training Center
Knowledge for Automation

Централизованная конфигурация 2

Интерфейсные модули IM 460-0/IM 461-0 предназначены для централизованного подключения стоек расширения к центральной стойке S7. Централизованный тип расширения 2 имеет следующие характеристики:

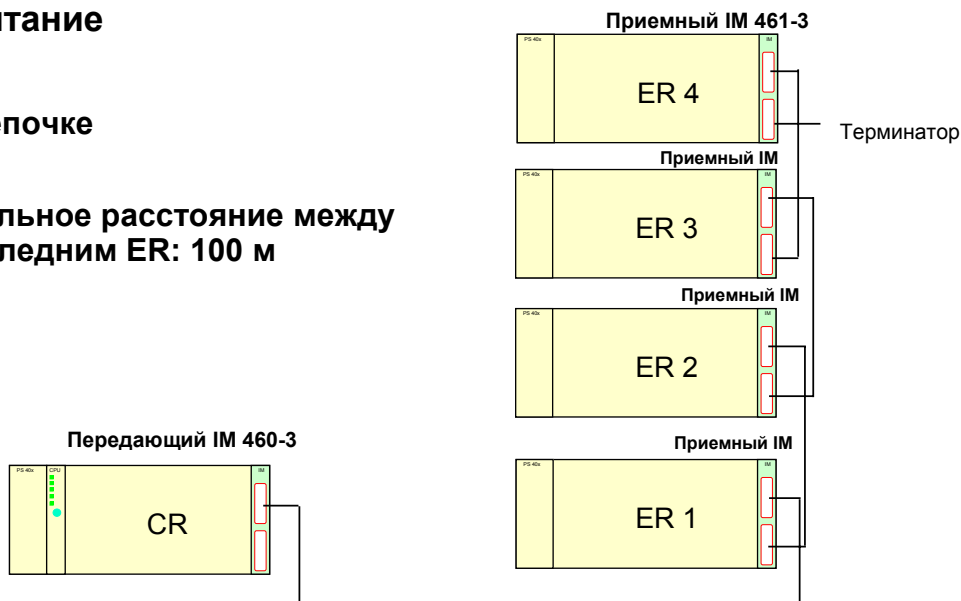
- Максимум 8 стоек расширения может быть связано (4 на интерфейс).
- Максимальное расстояние между центральной стойкой и самой дальней стойкой расширения - 3 м.
- Максимум 6 передающих IM 460-0 может быть вставлено в центральную стойку.
- Передающий интерфейсный модуль IM 460-0 подключает Р- шину и К- шину к стойке расширения. Модули в стойке расширения не обеспечиваются напряжением питания через соединительный кабель. По этой причине каждая стойка расширения должна иметь собственный источник питания.
- Незанятый интерфейс в передающем IM 460-0 не должен быть нагружен на терминатор; незанятый интерфейс в приемном IM 461-0 должен быть нагружен на терминатор.
- Переключатель, которым выбирается номер стойки расширения, размещен на приемном IM 461-0.

Обратите внимание

Максимум 21 стойка расширения может быть связана с одной центральной стойкой. К-шина подключена только к первым 6-и стойкам расширения, то есть интеллектуальные модули типа FM и CP следовательно могут эксплуатироваться только в первых 6-и ER. Принимающий IM должен всегда вставляться в самый дальний правый слот в стойке расширения.

Распределенное расширение

- Соединены Р-шина и К-шина, но не питание
- 4 ER в цепочке
- Максимальное расстояние между CR и последним ER: 100 м



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.13



Information and Training Center
Knowledge for Automation

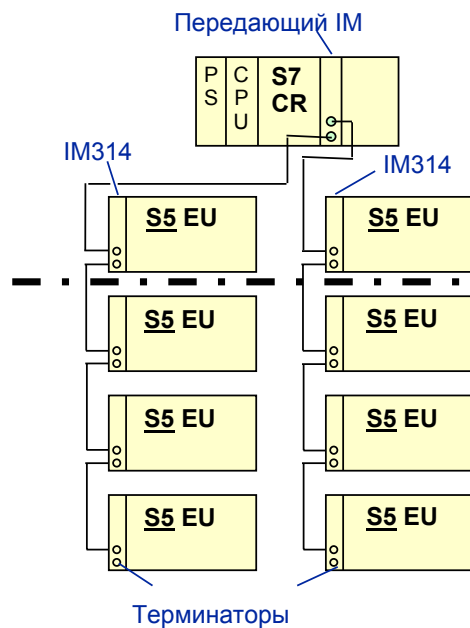
Распределенная конфигурация

Интерфейсные модули IM 460-3/IM 461-3 предназначены для распределенного подключения стоек расширения к центральной стойке S7. Распределенное расширение имеет следующие характеристики:

- Максимум 8 стоек расширения может быть связано (4 на интерфейс).
- Максимальное расстояние между центральной стойкой и самым дальней стойкой расширением - 100 м.
- Максимум 6 передающих IM 460-3 может быть вставлено в центральную стойку.
- Передающий интерфейсный модуль IM 460-3 подключает к стойке расширения Р-шину и К-шину
Модули в стойке расширения не обеспечиваются напряжением питания через соединительный кабель. По этой причине каждая стойка расширения должна иметь собственный источник питания.
- Незанятый интерфейс в передающем IM 460-3 не должен быть нагружен на терминатор; незанятый интерфейс в приемном IM 461-3 должен быть нагружен на терминатор.
- Переключатель, которым выбирается номер стойки расширения, размещен на приемном IM 461-3.

Обратите внимание Максимум 21 стойка расширения может быть связана с одной центральной стойкой. К-шина только связана с первыми 6-ю стойками расширения, то есть интеллектуальные модули типа FM и CP могут, следовательно, эксплуатироваться только в первых 6-и ER. Приемный IM должен всегда вставляться в самый дальний правый слот в стойке расширения.

Распределенное расширение между S7 и S5



- Макс. 4 S5 модулей расширения в цепочке
- Макс. 4 передающих IM в центральной корзине
- Макс. расстояние от CR до последнего EU в цепочке: 600m
- Соединение через параллельную S5-шину
- Доступные S5-модули для расширения:
EU 183 U, EU 185 U,
ER 701-2, ER 701-3
- Другие S5 EU
- Макс. 32 S5 EU на S7-400 CR

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.14



Information and Training Center
Knowledge for Automation

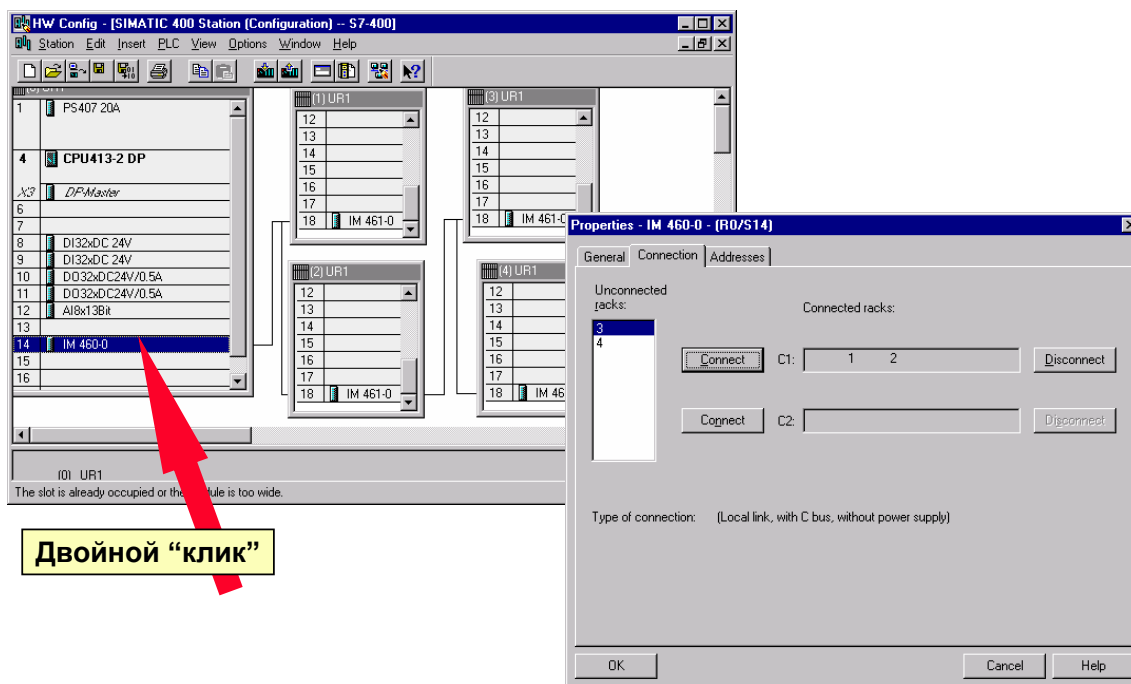
Распределенная конфигурация с S5 EU

Интерфейсный модуль IM463-2 дает возможность модулям расширения S5 быть подключенными к центральной стойке S7-400. Следующие правила применяются при соединении со стойками расширения S5:

- Максимум 4 IM463-2 можно подключить к центральной стойке.
- Максимум 4 модуля расширения на цепочку
- Максимальное расстояние между CR и последним EU - 600 м
- Старшие S5-системы могут быть связаны с центральной стойкой S7-400, если первый S5 EU - EU-183U/EU-185U для S5-135U/-155U или ER-701-2/ER-701-3 в случае S5-115U.

Дальнейшее расширение EU может быть проведено в соответствии с правилами S5.

Расширение централизованной конфигурации



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.15



Information and Training Center
Knowledge for Automation

Расширение конфигурации

Если Вы хотите присоединить стойки расширения, поступайте следующим образом:

1. Из аппаратного каталога выбирают желаемую стойку расширения.
2. Перетащите стойки одну за другой в окно станции методом Drag&Drop.
3. Вставьте желаемые модули в стойки.

Важно: Приемные интерфейсные модули должны быть вставлены во все стойки расширения перед подключением к передающему интерфейсному модулю в центральной стойке.

4. Только для S7-400: Чтобы установить подключение между передающим IM и приемными IM стоек расширения, поступайте следующим образом:

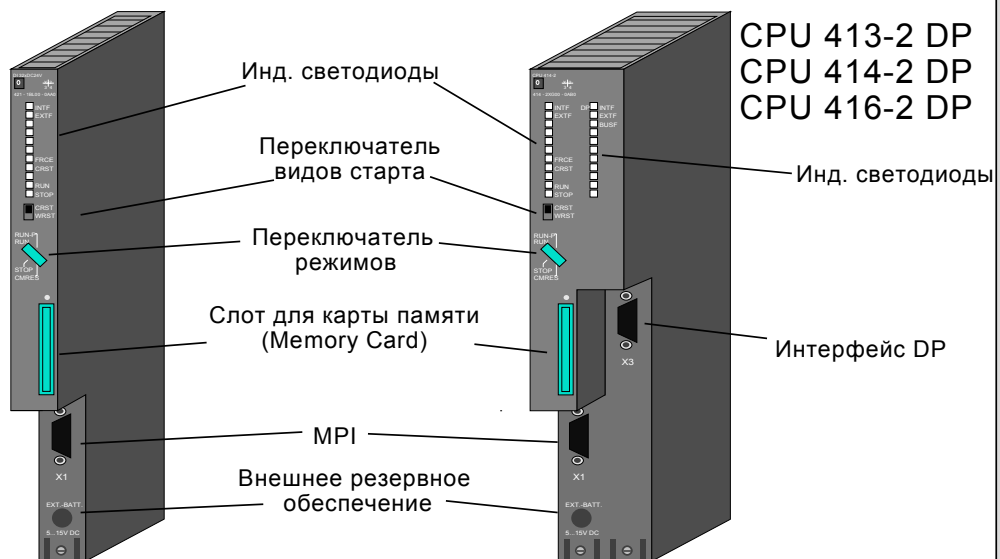
- Дважды щелкните на передающем IM
- Выберите закладку "Connection". Все стойки, которые не связаны, отображены в списке.
- Выбирайте каждую стойку по очереди и, используя кнопку "Connect", подключите их к желаемому интерфейсу передающего IM (C1 и C2). Линии подключения между стойками отображаются в окне станции.

Особенности CR2

При работе с центральной стойкой CR2, Вы должны сначала создать подключение между пустым (кроме приемного IM) стойками с передающими IM соответствующего CPU, прежде, чем Вы сможете вставлять модули в стойку расширения.

Модули CPU

CPU 412-1
CPU 413-1
CPU 414-1
CPU 416-1



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.16



Information and Training Center
Knowledge for Automation

Секция батарей

Обеспечивает внешнее напряжение постоянного тока 5V - 15V, чтобы сохранять RAM, например при замене модуля питания. RAM может сохраняться внутренней батареей модуля питания или через подключение "EXT-BATT." Разъем разработан для разъема 2.5 мм.

МПИ интерфейс

Интерфейс МПИ используется, чтобы подключить устройства программирования и/или системы интерфейса оператора.

Интегрир. PROFIBUS-DP интерфейс

CPU 413-2/414-2/416-2 имеют интегрированный PROFIBUS-DP мастера, с помощью которого подключаются модули распределенного ввода-вывода, типа ET200M, ET200U (B/C), S7-300, и т.д.

Карты памяти

RAM или платы FLASH-EPROM могут быть вставлены в CPU S7-400 как внешняя загрузочная память согласно индивидуальным требованиям.

- платы RAM объемом в 64КБ, 256КБ, 1МБ и 2МБ сохраняются с помощью батареи в блоке питания.
- платы FLASH-EPROM объемом в 64КБ, 256КБ, 1МБ, 2МБ, 4МБ, 8МБ, 16МБ.

Рабочие режимы

MRES = Сброс памяти

STOP = Режим STOP, то есть нет обработки программы

RUN = Программа выполняется, доступ из PG возможен только для чтения.

RUN-P = Программа выполняется, доступ из PG возможен для чтения и записи.

Переключатель режимов запуска

CRST = Полный рестарт CPU (Холодный рестарт)

WRST = Рестарт CPU (Теплый рестарт)

Технические данные CPU S7-400 (1)

CPU	412-1	413-1	413-2 DP	414-1	414-2 DP	416-1	416-2 DP	417-4
Выполнение двоичной инструкции	200 ns	200 ns	200 ns	100 ns	100 ns	80 ns	80 ns	100 ns
Загрузка/Передача (word)	200 ns	200 ns	200 ns	100 ns	100 ns	80 ns	80 ns	100 ns
16-бит фиксир. точка (+/-)	200 ns	200 ns	200 ns	100 ns	100 ns	80 ns	80 ns	100 ns
IEEE плав. точка (+/-)	1,2 µs	1,2 µs	1,2 µs	0,6 µs	0,6 µs	0,48 µs	0,48 µs	0,48 µs
Пользоват. память								
Рабочая память	48 KB	72 KB	72 KB	128 KB	128/384 KB	512 KB	0.8/1.6 MB	4...20 MB
Интегрир. загруз. память	8 KB	8 KB	8 KB	8 KB	8 KB	16 KB	16 KB	256 KB
Внешняя загруз. память	15 MB	15 MB	15 MB	15 MB	15 MB	15 MB	15 MB	64 MB
Адреса								
Обл. меркеров	4096	4096	4096	8192	8192	16384	16384	16384
Часы раб. времени	8	8	8	8	8	8	8	8
Таймеры	256	256	256	256	256	512	512	512
Счетчики	256	256	256	256	256	512	512	512
Типы блоков/номера								
FB	256	256	256	512	512	2048	2048	6144
FC	256	256	256	1024	1024	2048	2048	6144
DB	511	511	511	1023	1023	4095	4095	8192
Размер области отображения (таблица входов/выходов)	128 byte каждая	128 byte каждая	128 byte каждая	256 byte каждая	256 byte каждая	512 byte каждая	512 bytes each	1024 bytes each
Макс. пространство адресов I/O	0.5 KB ¹⁾ каждое	1 KB ¹⁾ каждое	1 KB ¹⁾ каждое	2 KB ¹⁾ каждое	4 KB ¹⁾ каждое	4 KB ¹⁾ каждое	8 KB ¹⁾ each	16 KB ¹⁾ each
Интегрированные интерфейсы	MPI	MPI	MPI, DP	MPI	MPI, DP	MPI	MPI, DP	MPI, 4 x DP

¹⁾ 1 Byte = 8 цифровых входов/выходов
2 Byte = 1 аналоговый вход/выход

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.17



Information and Training Center
Knowledge for Automation

Типы CPU

Для каждого класса задач существует CPU с соответствующим быстродействием, размером рабочей памяти и числом блоков.

Программирование Программы пользователя пишутся в соответствии с IEC 1131-3. В S7 были интегрированы следующие новые характеристики:

- Команды для аналоговой обработки
 - арифметика с фиксированной запятой и с плавающей запятой (32разрядная)
 - квадрат и квадратный корень
 - логарифмическая функция
 - тригонометрические функции
- Новые типы данных для проблемно-ориентированного программирования, например:
 - ARRAY (массив)
 - STRUCT (структура)
 - POINTER (указатель)
- Концепция объектно-ориентированных блоков, например:
 - FB с экземпляром блока данных
 - модель мультиэкземпляров
- Интегрированные системные блоки, например для коммуникаций, и т.д.

Обработка входов - выходов

Логические адреса модулей ввода - вывода размещаются в линейном адресном пространстве подходящего размера. Адреса управляемых станций, соединенных с интегрированным интерфейсом DP также находятся в этом линейном адресном пространстве. Таким образом, распределенный ввод - вывод может быть адресован в программе пользователя тем же самым способом, что централизованный ввод - вывод. Чтобы назначить адреса для центрального и распределенного ввода - вывода, используется STEP 7.

Технические данные CPU S7-400 (2)

CPU	412-1	413-1	413-2 DP	414-1	414-2 DP	416-1	416-2 DP	417-4
Организационные блоки	№ОВ	№ОВ	№ОВ	№ОВ	№ОВ	№ОВ	№ОВ	№ОВ
Свободный цикл	1	1	1	1	1	1	1	1
Прер. по дате и врем.	10,11	10,11	10,11	10-13	10-13	10-17	10-17	10-17
Прер. с задержкой	20,21	20,21	20,21	20-23	20-23	20-23	20-23	20-23
Циклич. прерыв.	32,35	32,35	32,35	32-35	32-35	30-38	30-38	30-38
Аппаратн. прерыв.	40,41	40,41	40,41	40-43	40-43	40-47	40-47	40-47
Мультикомп. прерыв.	60	60	60	60	60	60	60	60
Фоновый	90	90	90	90	90	90	90	90
Запуск	100-102	100-101	100-102	100-102	100-101	100-102	100-102	100-102
Синхронные ошибки	80-87	80-87	80-87	80-87	80-87	80-87	80-87	80-87
Асинхронные ошибки	121,122	121,122	121,122	121,122	121,122	121,122	121,122	121,122
Локальные данные	4 KB	4 KB	4 KB	8 KB	8 KB	16 KB	16 KB	24 KB
Макс. длина блока	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB	64 KB
Глубина вложения блоков на класс приоритета	16	16	16	16	16	16	16	24
Программно-управл. коммуникации: Макс. число. соедин.	8	16	16	32	32	64	64	64
Глобальн. данные коммуникации через MPI: GD циклов на CPU	8	8	8	8	8	16	16	16
Передав. GD пакетов на GD цикл	1	1	1	1	1	1	1	1
Приемн. GD пакетов на GD цикл	2	2	2	2	2	2	2	2
Макс. пользов. данных на пакет	54 bytes	54 bytes	54 bytes	54 bytes	54 bytes	54 bytes	54 bytes	54 bytes

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.18Information and Training Center
Knowledge for Automation**Связь**

S7-400 предлагает несколько различных средств связи.

1. Интегрированный многоточечный интерфейс (MPI-интерфейс), через который могут соединяться как активные узлы PG / PC, HMI-системы, системы M7-300/400 и также системы S7-300/400.
2. Интегрированный интерфейс PROFIBUS-DP связывает CPU 413-2/414-2/416-2/417-4 с распределенной системой ввода - вывода (например, ET200).
3. Коммуникационные процессоры типа CP443 для коммуникаций по шинам PROFIBUS и Industrial Ethernet.
4. Коммуникационные процессоры типа CP441 для связи точка к точке (PtP) с S7, S5 или внешними (не фирмы Siemens) контроллерам и системам.

S7-функции

Коммуникационные S7-функции разделены на два типа:

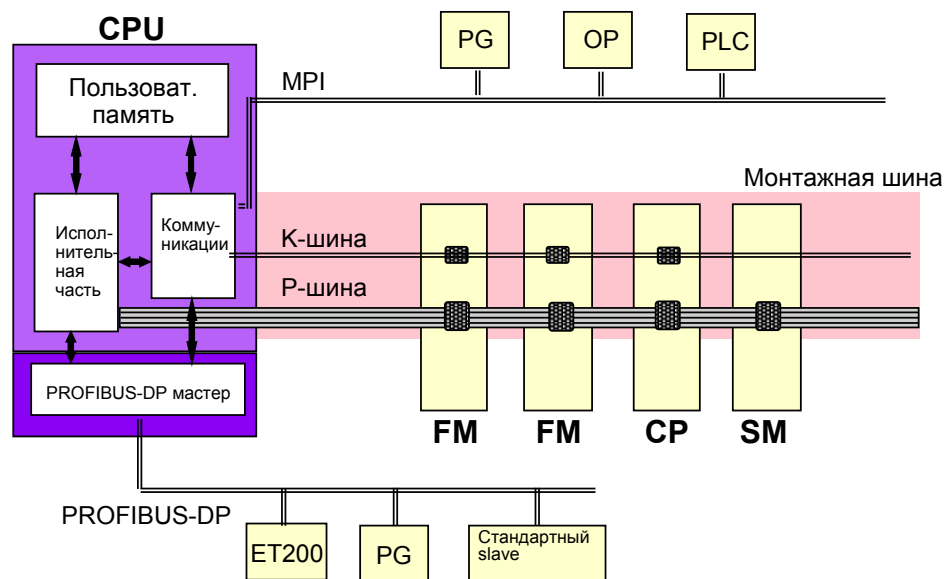
S7 базовые коммуникации: Между партнерами по связи (S7-300/400) можно передавать небольшие объемы данных (до 76 байтов) через MPI или внутри станции (к интеллектуальным пассивным узлам - через PROFIBUS-DP).

Коммуникационные SFC интегрированы в операционную систему. Они не требуют конфигурирования связей, распределения коммуникационных ресурсов и адресации партнера по связи происходит непосредственно при вызове SFC.

S7 расширенные коммуникации: Можно передавать большие объемы данных (до 64 Кбайт) независимо от сети (MPI, Profibus или Industrial Ethernet).

Соответствующие SFB интегрированы в операционную систему для S7-400 (не для S7-300, S7-300 - только сервер) и требуют конфигурированного соединения при вызове SFB. Конфигурирование соединений происходит через таблицу соединений, когда система запускается и соответствующие ресурсы назначены статически.

Системная архитектура



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.19



Information and Training Center
Knowledge for Automation

Конфигурация CPU S7-400 CPU разделены на два функциональных блока:

- Процессорная часть
- Коммуникационная часть

Р-шина

Процессорная часть CPU осуществляет доступ к сигнальным модулям через Р-шину. Она оптимизирована для обмена 4-мя байтами данных сразу. Входная Р-шина системы S7-400 имеет следующие характеристики:

- ширина 8 битов
- параллельная
- время доступа 1.5 μ s

К-шина

К-шина (коммуникационная шина) осуществляет асинхронный обмен данными с интеллектуальными, способными к коммуникациям модулями, например FM или CP. Она оптимизирована для обмена большими количествами данных.

К-шина разработана как много-мастерная шина; это - логическое расширение интерфейса MPI. Она имеет следующие характеристики:

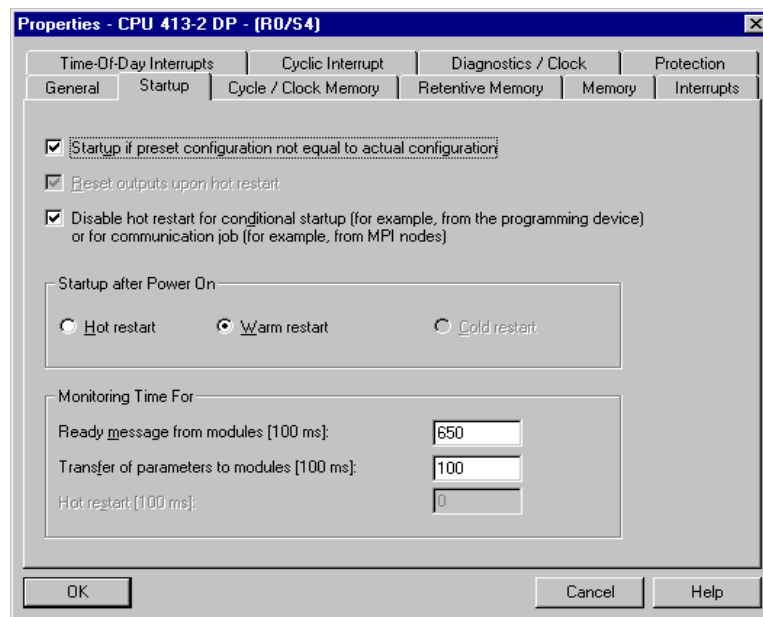
- последовательная
- скорость в бодах: 10.5 Мбайт / с
- максимально 127 узлов (теоретически)

MPI

Связь через интерфейс MPI имеет следующие характеристики:

- последовательная
- скорость в бодах: 187.5 Мбайт / с
- максимально 32 узла

Параметры CPU : характеристики запуска



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.20



Information and Training Center
Knowledge for Automation

Startup if present configuration not equal to actual configuration

Дезактивируя это поле, Вы можете сделать так, чтобы CPU переходил в STOP после запуска, если фактическая конфигурация не совпадает с установленной конфигурацией (согласно конфигурации).

Hardware Test

Активизируя эту функцию, внутренняя RAM CPU проверяется во время запуска. Продолжительность теста зависит от размера памяти.

Delete PIQ..

При рестарте S7-400, изображение процесса удаляется по умолчанию после того, как остаток цикла обработан. Если это не желательно, Вы можете отменить выбор.

Disable hot restart..

Ограничение полным рестартом при ручном пуске S7-400.

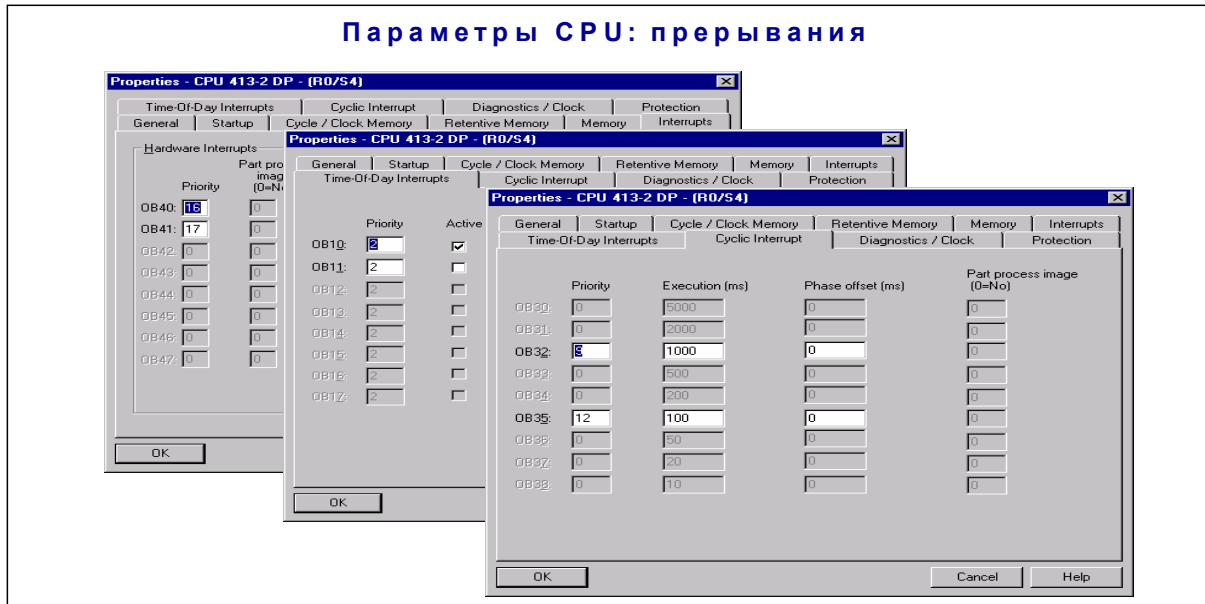
Startup after Power On В S7-400 Вы имеете выбор видов запуска:

- Полный рестарт (Warm restart) (стирание нереманентных областей и начало обработки программы с первой команды в блоке OB1).
- Рестарт (Hot restart)(все области памяти остаются, и программа продолжается с места прерывания).
- Холодный рестарт (Cold restart)(стирание реманентных и нереманентных областей и программы. Обработка начинается с первой команды в блоке OB1).

Monitoring Time For Следующие времена могут быть определены:

- максимальное время ожидания, пока все модули не послали квитанции CPU.
- максимальное время, в течение которого модуль должен подтвердить передачу параметров.
- для S7-400, максимальное время после сбоя питания, в течение которого рестарт может еще выполняться (0 означает всегда).

Параметры CPU: прерывания



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PROZ_11E.21



Information and Training Center
Knowledge for Automation

Приоритеты

В S7-400 Вы можете изменять приоритеты блоков, обрабатывающих прерывания и таким образом, устанавливать последовательность, в которой они обрабатываются, когда события, вызывающие прерывания происходят одновременно. Приоритеты - от 1 до 28 - и прерывание с самым высоким приоритетом обрабатывается первым.

Hardware Interrupts

В этом блоке параметров Вы устанавливаете приоритеты организационных блоков для аппаратных прерываний. Вы можете устанавливать приоритеты 0 и от 2 до 24 (0 отменяет выбор).

Time-of-Day Interrupt

Вы можете использовать эти прерывания, чтобы установить время запуска для выполнения программы однократно или циклически начиная с указанного момента времени (каждую минуту, каждый час, ежедневно, еженедельно, ежемесячно, ежегодно).

Cyclic Interrupt

Циклическое прерывание может использоваться, чтобы выполнить часть программы через фиксированные интервалы времени. Вы можете выбирать интервалы от 1 до 60000 ms. Это дает возможность Вам, например, выполнять задачи управления с обратной связью (замкнутый контур управления), которые должны быть обработаны в фиксированных интервалах (интервал дискретизации).

На S7-400 имеется восемь различных циклических прерываний с различными интервалами. Чтобы предотвратить одновременный запуск циклических прерываний, Вы можете устанавливать "Phase Offset" ("Фазовый сдвиг") так, чтобы вызов циклических прерываний не был одновременным.

Time-Delay Interrupts

Прерывание с задержкой - одноразовый вызов организационного блока, который активизируется с запаздыванием, например после того, как получен сигнал процесса.

Прерывания с задержкой обрабатываются в программе пользователя при помощи SFC 32 - 34.

- SFC32 "SRT_DINT" = запуск прерывания с задержкой.
- SFC33 "CAN_DINT" = отмена прерывания с задержкой.
- SFC34 "QRY_DINT" = опрос состояние прерывания с задержкой.

Параметры CPU: локальные данные

Properties - CPU 413-2 DP - (R0/S4)

Time-Of-Day Interrupts | Cyclic Interrupt | Diagnostics / Clock | Protection
 General | Startup | Cycle / Clock Memory | Retentive Memory | Memory | Interrupts

Local Data (Priority Classes)

1	512	7	0	13	0	19	0	25	256
2	256	8	0	14	0	20	0	26	256
3	256	9	256	15	0	21	0	27	256
4	256	10	0	16	256	22	0	28	256
5	0	11	0	17	256	23	0	29	256
6	0	12	256	18	0	24	256		

Occupied 3840 Bytes of max. 2048

Communication Resources
 Max. number of communication jobs: 150

OK Cancel Help

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
 File: PRO2_11E.22



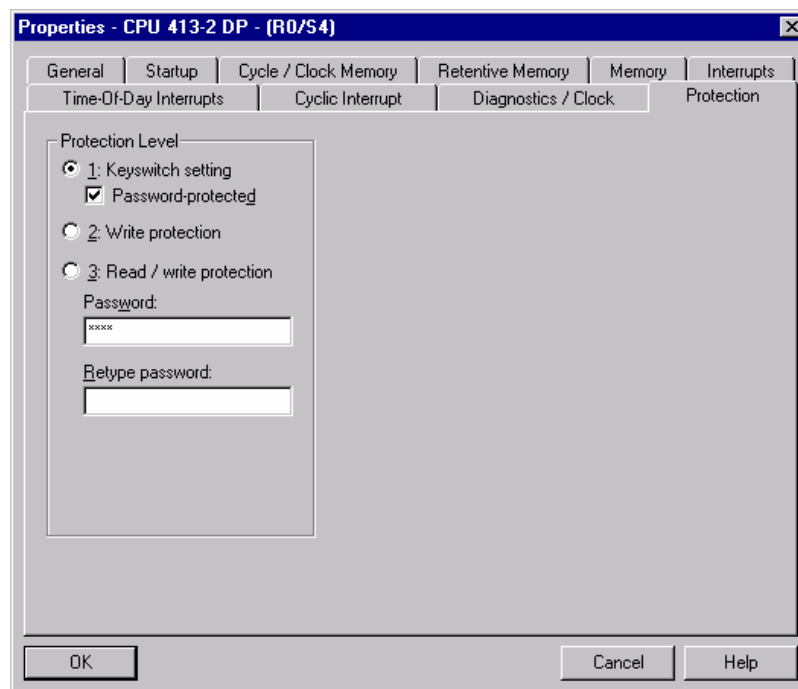
Information and Training Center
 Knowledge for Automation

Локальные данные Вышеупомянутая страница позволяет пользователю определять требования к локальным данным для каждого класса приоритета (ОВ). Если локальная область данных класса приоритета превышена, CPU переходит в режим STOP. Если Вы хотите адресовать локальные данные символически, STL/LAD/FBD-редактор гарантирует правильную адресацию и администрирование.

Размер L-стека Количество локальных данных зависит от CPU.

- CPU 412 - 4 Кбайта локальных данных
- CPU 413 - 4 Кбайта локальных данных
- CPU 414 - 8 Кбайт локальных данных
- CPU 416 - 16 Кбайт локальных данных
- CPU 417 - 32 Кбайта локальных данных

Параметры CPU: концепция защиты



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.23



Information and Training Center
Knowledge for Automation

Функция

В этом диалоге Вы можете выбирать один из трех уровней защиты, чтобы защитить CPU от несанкционированного доступа.

Предварительно установленные характеристики

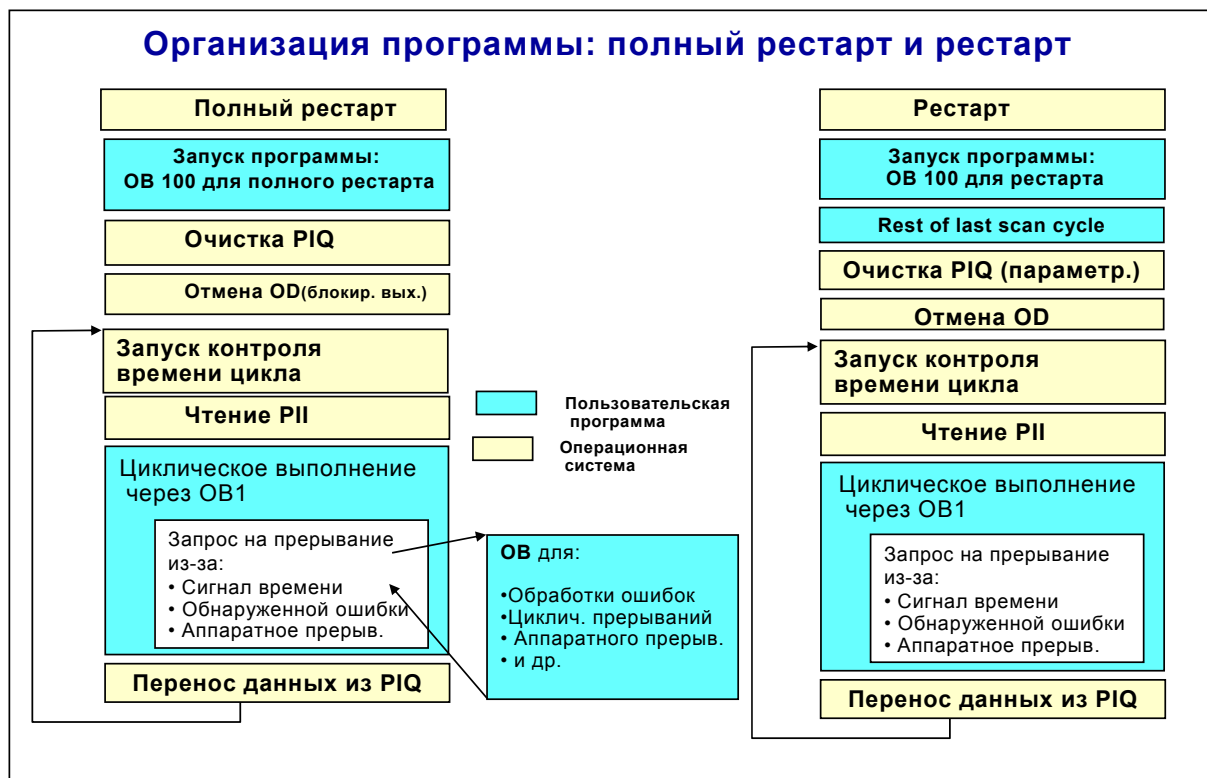
Уровень защиты 1 (нет пароля): позиции переключателя режимов CPU определяют защиту:

- Переключатель расположен в положениях RUN-P или STOP: никакие ограничений.
- Переключатель расположен в положении RUN: доступ возможен только для чтения !

Параметризация уровней защиты

Если Вы имеете параметризовали уровень защиты паролем:

- Чтение и доступ для записи возможен для владельца пароля, независимо от позиции переключателя и независимо от параметризованного уровня защиты.
- Применимы следующие ограничения для "невладельца пароля":
 - Уровень защиты 1: соответствует предварительно установленным характеристикам
 - Уровень защиты 2: возможный доступ только для чтения, независимо от позиции переключателя
 - Уровень защиты 3: ни чтение ни запись не возможны, независимо от позиции переключателя.



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.24Information and Training Center
Knowledge for Automation**Полный рестарт
(Complete Restart)**

При полном рестарте нереманентные области памяти: меркеры, счетчики и таймеры сбрасываются, после этого вызывается OB100.

**Холодный рестарт
(Cold Restart)**

Действия, выполняемые в течение холодного рестарта подобны действиям для полного рестарта, со следующими отличиями:

- вместо OB100 вызывается OB102
- блоки данных, сгенерированные с помощью SFC во времени выполнения, удаляются, другие блоки данных инициализируются значениями из загрузочной памяти.
- изображение процесса и все таймеры, счетчики и меркеры сброшены, независимо от того, являются ли они реманентными или нет.

**Перезапуск
(Restart)**

При рестарте, после того, как блок OB101 был выполнен, выполнение OB1 продолжается в месте, где он был прерван, то есть остаток цикла завершается с реманентными меркерами, таймерами и счетчиками.

OD

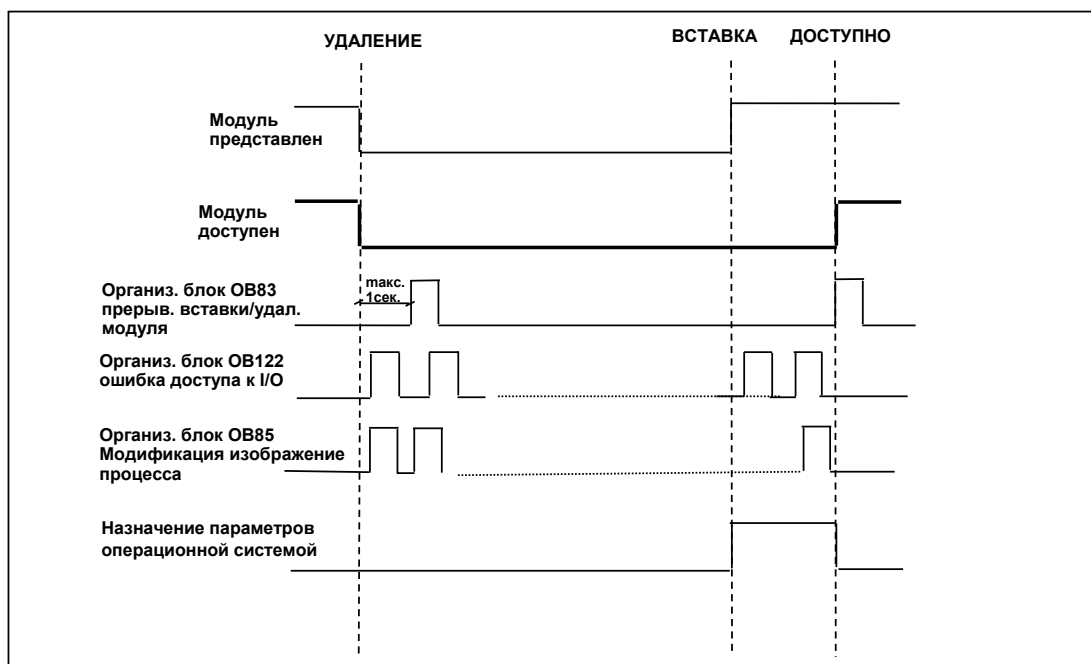
Output disable - Выход запрещен.

Действия

Операционная система выполняют следующие действия при запуске:

- Очищает стеки (C/CR)
 - Сбрасывает нереманентные меркеры, таймеры и счетчики (CR)
 - Сбрасывает все меркеры, таймеры и счетчики (C)
 - Очищает изображение процесса в таблице выходов PIQ (C/CR), если параметризовано (R)
 - Сбрасывает область памяти выходов (C, CR), если параметризовано (R)
 - Очищает (запрещает) прерывания (C/CR/R) через OD
 - Модифицирует список состояния системы (C/CR/R)
 - Передает конфигурацию в модули (C/CR/R)
- (CR = Полный рестарт, C = Холодный рестарт, R = Рестарт).

Прерывание вставки/удаление-модуля в S7- 400



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.25Information and Training Center
Knowledge for Automation

Прерывание удаления и вставки OB83

В S7-400 Вы можете удалять и вставлять модули в то время как он находится в режимах RUN или STOP.

Исключения - CPU, источники питания, S5-модули и IM.

После удаления модуля в RUN, операционная система CPU может вызывать один из следующих организационных блоков - в зависимости от местоположения:

- OB85 -модификация изображения процесса
- OB122 - ошибка доступа к I/O
- OB83- событие удаления/вставки.

Пользователь должен учесть, что OB83 вызывается приблизительно только через 1с., в то время как другие OB, как правило, активизируются намного скорее.

После вставки модуля это проверяется CPU и - если никаких ошибок типа не существует - назначаются параметры. После организованного назначения параметров, модуль доступен для использования.

Если ошибка распознана при назначении параметров, вызывается диагностическое прерывание OB82.

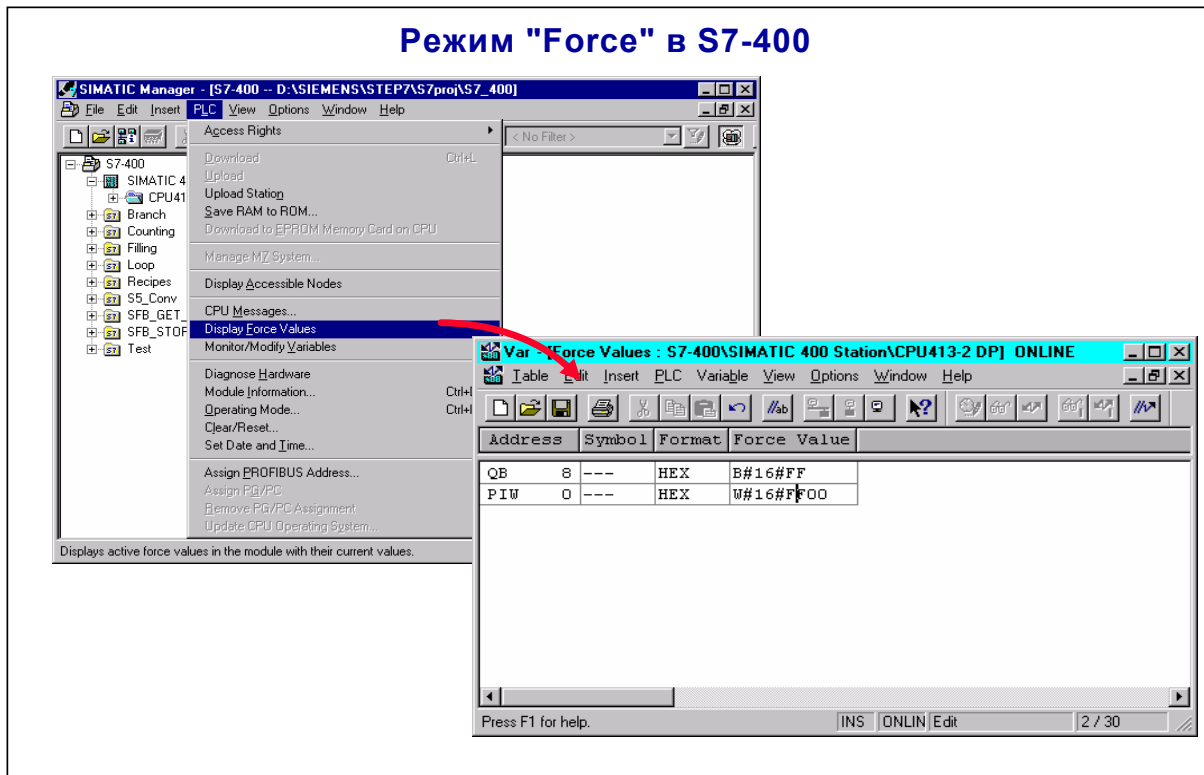
Стартовая информация в OB83

- Следующая информация существует в локальных данных OB83 :
- Удаление / вставка модуля
 - Логический адрес модуля
 - Фактический тип модуля

Заменяющее значение

Вы может использовать системную функцию SFC 44 (RPL_VAL) для замены значения в ACCU1, которое туда должен записать модуль, отсутствующий в данный момент (или вышедший из строя)
Функция вызывается в OB121-122.

Режим "Force" в S7-400



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.26Information and Training Center
Knowledge for Automation

Режим Force

С функцией Forcing Вы можете в S7-400 устанавливать predetermined значения для переменных программы пользователя.

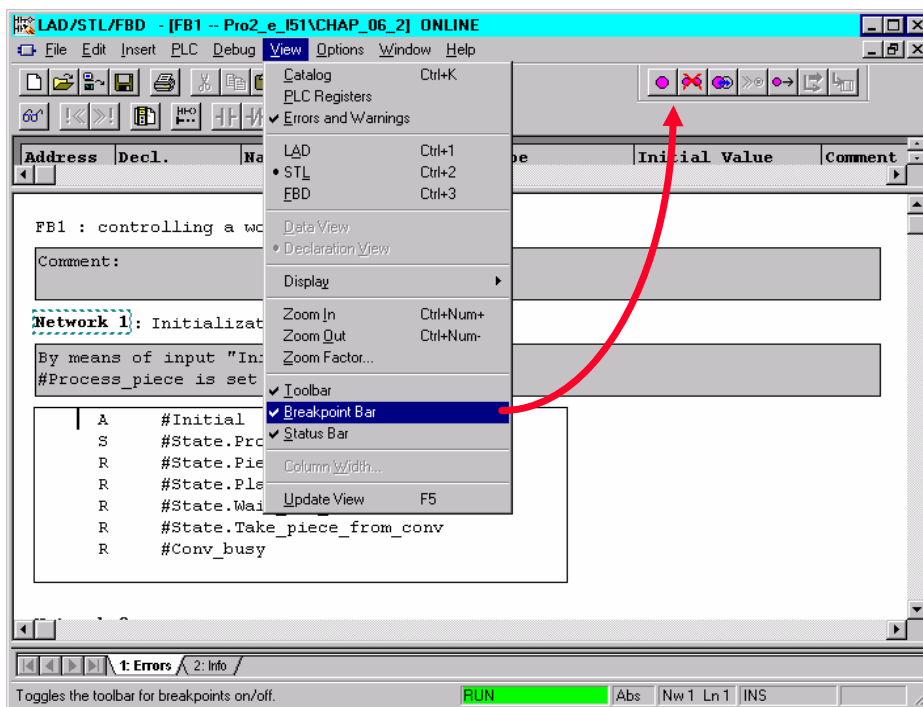
Замечания

- Прежде, чем Вы начинаете функцию "Force", Вы должны удостовериться, что никто еще не выполняет эту функцию одновременно на том же самом CPU.
- Фнкция Force может быть удалена или закончена только командой *Variable -> Stop Forcing*.
- "Forcing" не может быть отменен командой *Edit -> Undo*.
- Заккрытие окна "Force" значений или завершение приложения "Monitor/Modify Variables" не удаляет функцию Force.
- Читайте информация, предоставляемую On-line Help'ом, о различиях между Force и изменением переменных (Modifying Variables).

Выбор функции "Force"

1. В SIMATIC Manager выберите командой меню *PLC -> Display Force Values* CPU для "форсирования", чтобы открыть окно Force Values, в котором отображается текущее состояние выбранного CPU. Только, когда активно окно "Force Values", могут быть выбраны команды меню для "форсирования". Если никакого "форсирования" в настоящее время нет, окно пусто. Если "форсирование" активно, переменные отображаются в полужирном шрифтом с соответствующими "форсированными" значениями.
2. В столбце "Address", введите переменные, которые Вы хотите "форсировать". В столбце "Force Value", введите значения, которые Вы хотите назначать переменным.
3. Начните "форсирование" командой меню *Variable -> Force*. Если функция "Force" в настоящее время не активна, переменным назначаются "форсирующие" значения.
4. Вы можете завершить "форсирование" командой меню *Variable -> Stop Forcing*.

Активизация точек останова



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.27



Information and Training Center
Knowledge for Automation

Контрольные точки С помощью этой тестовой функции можно проверить программу, созданную в STL-представлении, в пошаговом режиме и, таким образом, можно проследить за последовательностью выполняемых инструкций, а также содержимое регистров. В блоке могут быть установлены несколько контрольных точек. Возможное число контрольных точек зависит от типа CPU

- CPU 412, 413: 2 контрольных точки
- CPU 414: 3 контрольных точки
- CPU 416: 4 контрольных точки

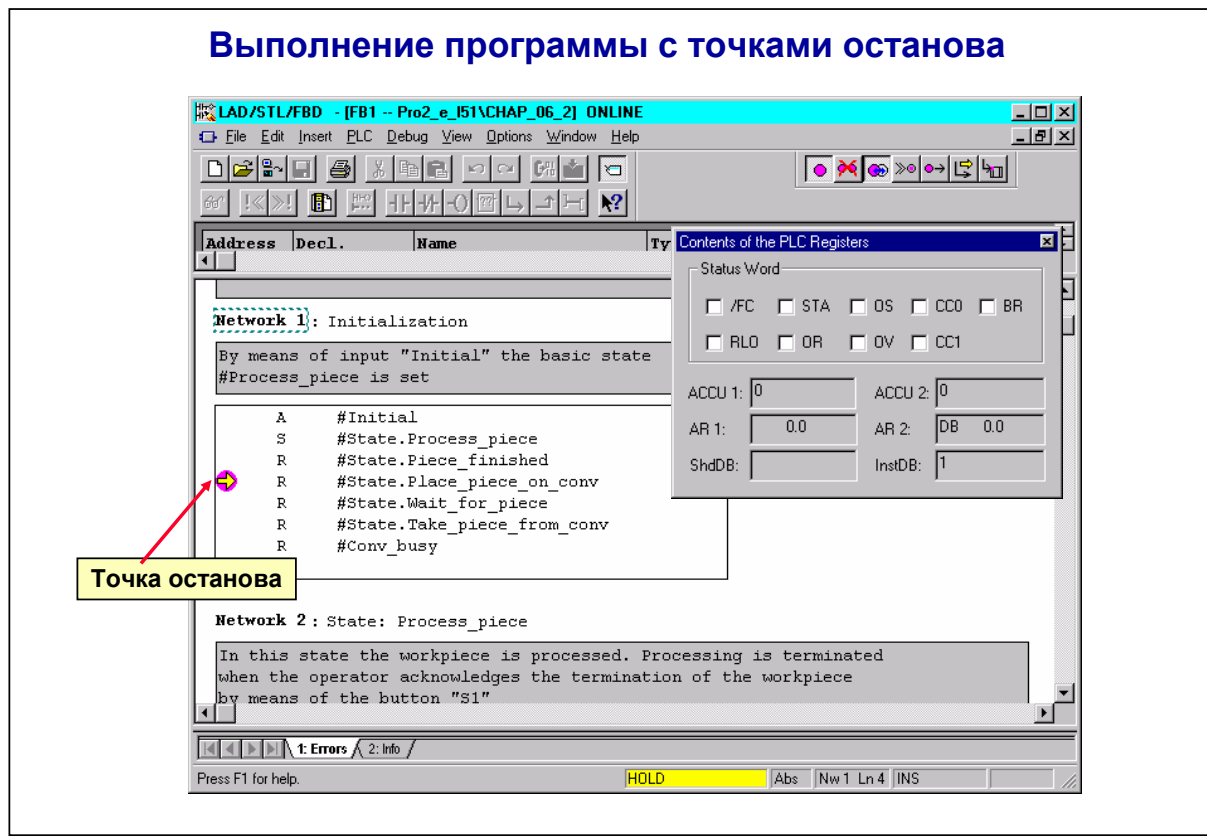
Обратите внимание

- Чтобы выбрать функцию "Breakpoint" блок должен быть открыт в режиме "on-line".
- Функция "Breakpoints" может быть запущена, если только выбрано *Debug -> Operation -> Test Operation*.
- Команды меню *Execute Next Statement* или *Execute Call* требуют свободной контрольной точки для внутреннего выполнения.
- Когда запущен тестовый режим с контрольными точками, CPU переключается из режима RUN в режим HOLD и в то же самое время вспыхивает светодиод STOP, а светодиод RUN мигает.

Функции контрольных точек Функции контрольных точек могут быть выбраны в программном редакторе через пункт меню "Debug" или из панели контрольных точек.

Панель контрольных точек Вы активизирует панель контрольных точек в программном редакторе через пункт меню *View -> Breakpoint Bar*.

Выполнение программы с точками останова



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.28



Information and Training Center
Knowledge for Automation

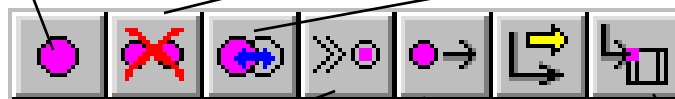
Панель точек останова

Панель контрольных точек предлагает командные кнопки для тестирования в пошаговом режиме.

Set Breakpoint

Delete Breakpoint

Breakpoint Active



Show Next Breakpoint

Continue

Next Statement

Execute Call

Set Breakpoint

С помощью "Set Breakpoint" Вы определяете, в каком месте выполнение программы должно быть приостановлено. Инструкция в контрольной точке не выполняется.

Delete Breakpoint

Все точки останова удаляются.

Breakpoint Active

С помощью "Breakpoint Active" Вы активизируете все точки останова; не только установленные, но и которые будут установлены.

Show Next Breakpoint

С помощью "Show Next Breakpoint" редактор устанавливает курсор на следующую точку останова, без выполнения программы.

Continue

С помощью "Continue" программа выполняется до следующей точки останова.

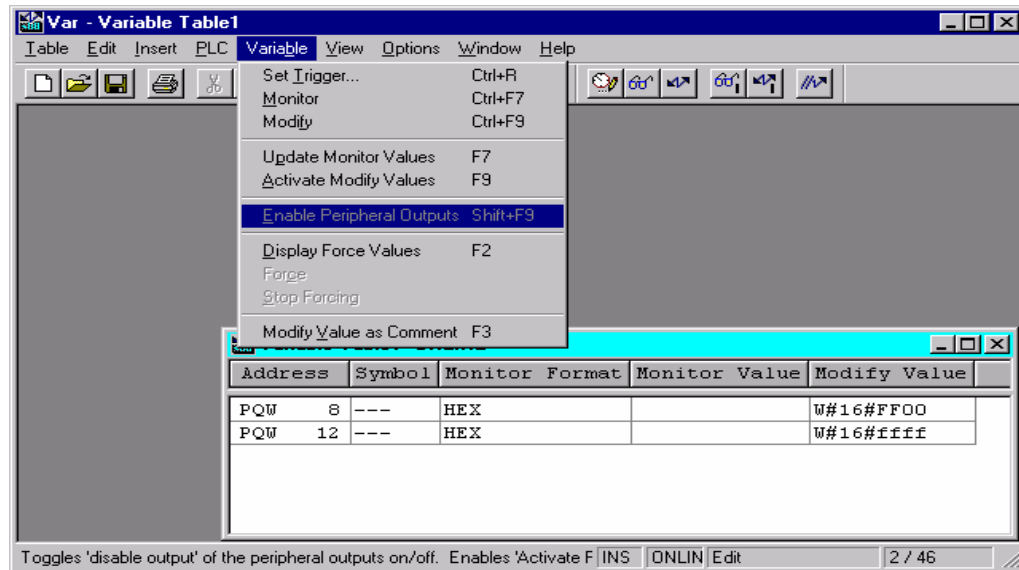
Next Statement

С помощью "Execute Next Statement" Вы выполняете программу в пошаговом режиме. Если встречается вызов блока, происходит переход на следующую команду после команды CALL (вызов выполняется как одна команда).

Execute Call

Отличается от "Execute Next Statement" тем, что при встрече команды CALL происходит переход на первую команду вызываемого блока. (Пошаговый режим с заходом в вызываемые блоки).

Доступ к периферийным выходам



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.29



Information and Training Center
Knowledge for Automation

Введение

Функция "Enable Peripheral Output" позволяет изменять периферийные выходы (PQ) независимо от входов. Это позволяет Вам изменять периферийные выходы, когда CPU находится в режиме STOP.

Вызов

Чтобы изменять периферийные выходы, поступайте следующим образом:

1. Используйте команду меню *Table -> Open*, чтобы открыть таблицу переменных (VAT). Она должна содержать периферийные выходы, которые Вы хотите изменять.
2. Выберите меню *PLC -> Connect*, чтобы установить связь с требуемым CPU, у которого Вы хотите изменять периферийные выходы.
3. Откройте "Operating Mode" с помощью команды меню *PLC -> Operating Mode* и переведите CPU в режим STOP.
4. Введите соответствующие значения для периферийных выходов, которые Вы хотите изменить в колонке "Modify Value".
Примеры: PQW 7 Modify value: 2#0001000011
PQW 2 W#16#0027
PQD 4 DW#16#0001
5. Используйте команду меню *Variable -> Enable Peripheral Output*, чтобы перейти в режим "Enable Peripheral Output".
6. Используйте команду меню *Variable -> Activate Modify Values*, чтобы изменить периферийные выходы. "Enable Peripheral Output" остается активной пока Вы снова не выберите команду меню *Variable -> Enable Peripheral Output*, чтобы выключить эту функцию.
7. Чтобы задать новые значения, начните с шага 4.

Обратите внимание

- Если CPU изменяет состояние, например, переходит из STOP в RUN или в STARTUP выдается сообщение.
- Если CPU находится в режиме RUN и выбрана функция "Enable peripheral outputs", также выдается сообщение.

CP 441 для соединений точка к точке

Интерфейсы:

- CP 441-1: 1 сменный интерфейсный модуль
- CP 441-2: 2 сменных интерфейсных модуля

Светодиоды:

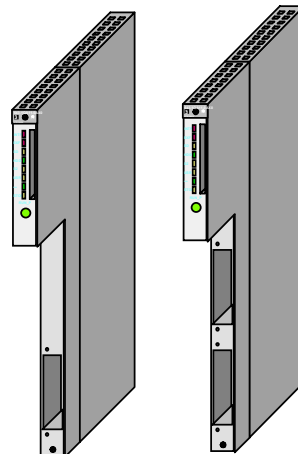
- Передача, прием, ошибка

Скорость передачи в бодах:

- CP 441-1: макс. 38.4 кБод
- CP 441-2: макс. 76.8 кБод

Протоколы:

- Интегрированные стандартные протоколы
- Загружаемые несименсовские протоколы (спец. драйверы) - для CP 441-2
- CP441-1: Дешевый со стандартными функциональными возможностями
- CP441-2: Высокая эффективность для требуемой задачи



CP 441-1

CP 441-2

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.30



Information and Training Center
Knowledge for Automation

Описание

Соединение "точка к точке" является рентабельной и быстродействующей альтернативой системе шин. Можно соединить простые устройства типа считывателей штрихового кода и масштабов, также как PLC.

Протоколы CP441-1 С одним вставным интерфейсом

- 3964 (R) параметризуемый
- ASCII параметризуемый протокол
- Принтер

Протоколы CP441-2 С двумя вставными интерфейсами

- 3964 (R) параметризуемый
- RK 512
- ASCII параметризуемый протокол
- Принтер
- Загружаемые внешние протоколы типа:
 - Modbus master / slave (Modicon)
 - Allen Bradley (DF1 протокол)

- Модули интерфейса**
- TTY, гнезда на 9 штырьков, активный / пассивный модуль интерфейса
 - V. 24 (RS232C) 9-штырьковый коннектор
 - RS 422/485, гнездо на 15 штырьков.

CP 443-5: соединение для PROFIBUS

Формат: шириной в один слот

Протоколы:

- SEND/RCV
- S7-функции
- FMS (только для CP 443-5 Basic)
- DP Master (только для CP 443-5 Extended)

Скорость передачи в бодах:

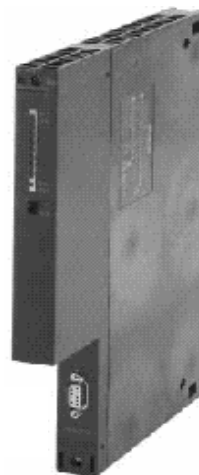
- от 9.6 кБод до 12 МБод

Соединения:

- Электрич. кабель: 9-pin sub-D разъем
- Световод: Шинный терминал

Конфигурирование:

- NCM S7 для PROFIBUS включая FC и FB



**CP 443-5 Basic
CP 443-5 Extended**

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.31



Information and Training Center
Knowledge for Automation

Описание

Коммуникационный процессор CP 443-5 разрешает подключение S7-400 к сети PROFIBUS.

Протоколы

Доступны следующие протоколы :

S7-функции:

Для коммуникаций, основанных на уровне 7 модели ISO/OSI между SIMATIC S7/M7/C7 и PC.

S7-протокол обеспечивает SFB-интерфейс для S7-CPU для связи в пределах семейства SIMATIC S7. Дополнительно функции для программирования, тестирования, обеспечения администрирования объекта и диагностики.

SEND/RCV:

Для связи, основанной на уровне 2 (FDL-уровень) между SIMATIC S7, SIMATIC S5, PC/PGS и устройствами "не - Siemens'a" . SEND/RCV интерфейс обеспечивает простую, надежную связь неструктурированных блоков данных.

SEND/RCV интерфейс осуществляется с помощью

- Обрабатывающих блоков в SIMATIC S5
- Вызываемых функций SIMATIC S7
- Вызовов C- функций на PGS/PC

PROFIBUS FMS (Fieldbus Message Specification)

Для открытой связи между SIMATIC S5, S7, PCS/PGS, полевыми устройствами и изделиями "не - Siemens". (EN 50170, Vol. 2, PROFIBUS). Протокол FMS разрешает объектно-ориентированную связь на уровне 7 из модели ISO/OSI. Типичный размер блока данных - 240 байтов.

PROFIBUS DP (Distributed Peripheral)

Для открытой связи между SIMATIC S5, S7, PCS/PGS или системами "не - Siemens" и полевыми устройствами. (EN 50170, Vol. 2, PROFIBUS).

DP-протокол разработан для быстрого обмена маленькими количествами данных между PLC или PC и полевыми устройствами с временем реакции < 10 ms.

IM 467: интерфейс PROFIBUS-DP мастер

Формат: шириной в один слот

Протоколы:

- DP Master
- S7-функции

Скорость передачи в бодах:

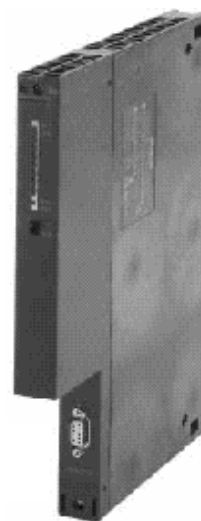
- от 9.6 кБод до 12 МБод

Соединения:

- Электрич. кабель: 9-pin sub-D разъем

Конфигурирование:

- Конфигурирование и программирование доступно через PROFIBUS-DP



IM 467

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.32



Information and Training Center
Knowledge for Automation

Описание

Интерфейсный модуль IM 467 предназначен для системы PLC S7-400. Он делает возможной связь S7-400 по PROFIBUS-DP.

Протоколы

IM 467 предлагает два сервиса для коммуникаций:

PROFIBUS-DP

IM 467 - PROFIBUS-DP Master согласно EN 50 170. Конфигурация полностью совместима с STEP 7. Работает в принципе идентично интегрированному интерфейсу PROFIBUS-DP на модуле CPU.

STEP 7 не вызывает никаких функций в программе пользователя для DP связи.

S7-функции

S7-функции гарантируют оптимальную и простую связь в решениях автоматизации, основанных на SIMATIC S7/M7/C7. Следующие S7-функции разрешены для IM 467:

- функционирование PG через PROFIBUS-DP
- функционирование HMI через PROFIBUS-DP

Связь имеет место без дальнейшей конфигурации IM 467.

S7-функции могут использовать по одному или параллельному PROFIBUS-DP-протоколу. Если они используются параллельными DP-коммуникациями, то это влияет на время цикла шины PROFIBUS-DP.

CP 443-1: соединение с Industrial Ethernet

Формат: шириной в один слот

Протоколы:

- SEND/RCV и S7-функции на стеке ISO Transport (CP 443-1) и стеке TCP/IP (CP 443-1 TCP/IP)
- CP444: MMS/MAP

Соединения:

- S7-функции : макс. 48 соединений
- SEND/RCV: макс. 64 соединений

Функции:

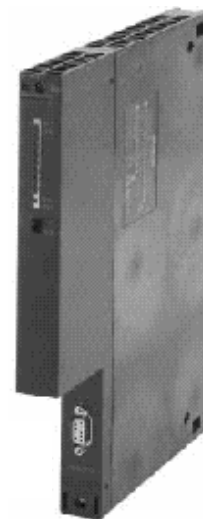
- Возможность использования многих протоколов
- Удаленное программирование через LAN и WAN (только CP443-1 TCP/IP)

Соединения:

- Автоматическое переключение между AUI витой парой

Конфигурирование:

- NCM-S7 для Industrial Ethernet, включая обращение к функциям для SEND/RCV



CP 443-1
CP 443-1 TCP/IP
CP 444

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.33



Information and Training Center
Knowledge for Automation

Описание Коммуникационные процессоры CP 443-1 и CP 443-1 TCP/IP обеспечивают связь S7-400 по сети Industrial Ethernet.

Протоколы Доступны следующие протоколы :

S7-функции

Для коммуникаций, основанных на уровне 7 модели ISO/OSI между SIMATIC S7/M7/C7 и PC.

SEND/RCV

Для связи, основанной на уровне 4 (ISO Transport Stack CP, 443-1 и TCP Transport Stack CP 443-1 TCP/IP) между SIMATIC S7, SIMATIC S5 и PC/PGS.

MMS/MAP

Для открытой связи, основанной на уровне 7 между SIMATIC S7, SIMATIC S5, PCS/PGS и системами "не - Siemens'a".

Эти функциональные возможности обеспечиваются CP 444.

CP 443-1 IT: подключение к Internet

Тот же самый формат и функциональные возможности как у CP 443-1 TCP:

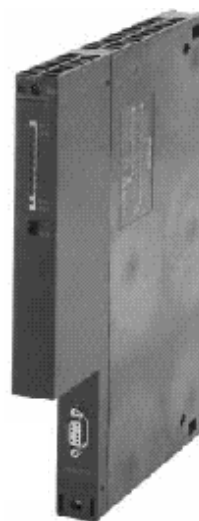
- S-функции
- Send/Receive через RFC 1006 и UDP

Дополнительная Internet-функциональность:
CP 443-1 IT является WWW станцией

- HTML страницы и апплеты для S7-функций
- WWW станция, используемая для управления / текущего контроля малых контроллеров
- нет издержек для конечного пользователя
- независимость от платформы
- знакомые оператору основные принципы Internet

E-mail клиент:

- Использование e-mail для исправления ошибок
- Доступ к моб. телефонам, пейджерам, PC, факсам и т.д.
- Одно обращение, различные системы, к которым обращаются одновременно, то есть не отдельное программное обеспечение для каждой системы



CP 443-1 IT

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_11E.34



Information and Training Center
Knowledge for Automation

Описание

Коммуникационный процессор CP 443-1 IT позволяет Вам соединять S7-400 с Internet.

Протоколы

Доступны следующие протоколы для CP 443-1 IT:

S7-функции

Для связи, основанной на уровне 7 модели ISO/OSI между SIMATIC S7/M7/C7 и PC.

SEND/RCV

Для связи, основанной на уровне 4 (TCP Transport Stack) между SIMATIC S7, SIMATIC S5 и PC/PGS.

Связь через Internet

Internet CP обеспечивает доступ к Internet. Это означает:

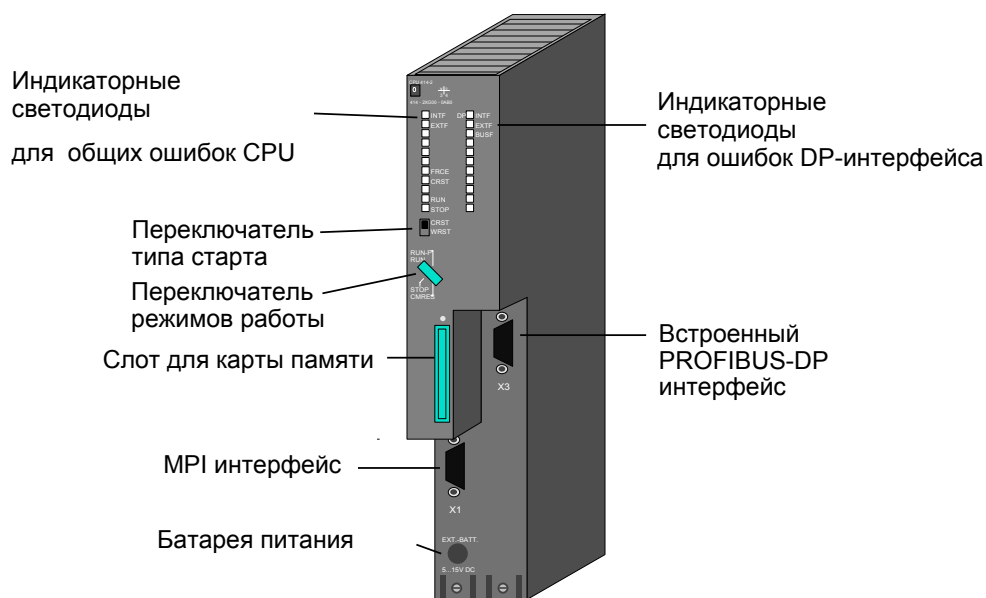
- Пользователи могут регистрироваться в системе отовсюду, используя пароль.
- Вы можете читать процесс и оперативные установки с любым Internet browser (Internet Explorer, Netscape, и т.д.). (Составные HTML-страницы для информационных систем).

Возможно также вмешательство оператора, позволяя обслуживание во всем мире.

Internet CP также позволяет Вам послать все важные данные или информацию состоянии установки в любую точку Земли, например :

- по электронной почте через Internet
- через портативный телефон или факса
- на внешние PC
- к pagers или palmtops с доступом в Internet.

Распределенный ввод - вывод и назначение параметров



SIMATIC S7

Siemens AG 1999. All rights reserved.

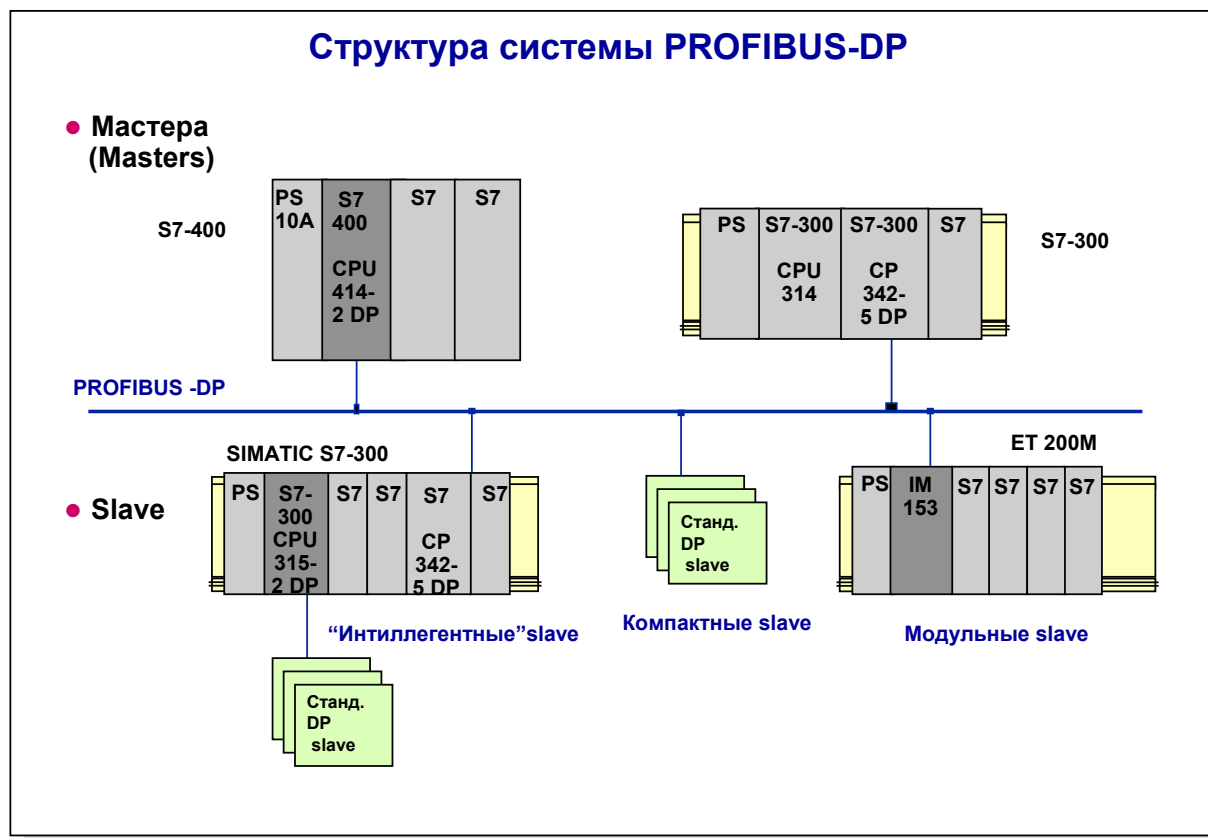
Date: 04.11.2005
File: PRO2_12E.1Information and Training Center
Knowledge for Automation

Содержание

Стр

Структура системы PROFIBUS-DP	2
Методы связи PROFIBUS	3
Время шинного цикла системы PROFIBUS-DP с одним мастером	4
PROFIBUS мастера в SIMATIC S7	5
Доступные DP slave	6
Терминатор PROFIBUS-DP	7
Конфигурирование системы DP Master	8
Конфигурирование компактных и модульных DP slave	9
Конфигурирование интеллектуальных DP slave (на примере CPU 315-2)	10
Вставка интеллектуальных DP-slave в мастер-систему	11
Анализ ошибок в OB 86 при отказе модуля-slave	12
Диагностика slave с помощью SFC 13 (DPNRM_DG)	13
Чтение последовательных данных из стандартного DP slave с помощью SFC 14	14
Запись последовательных данных из стандартного DP-slave с помощью SFC 15	15
Синхронизация DP slave с помощью SFC 11 (DPSYC_FR)	16
Установка (доустановка) PROFIBUS-DP slave	17

Структура системы PROFIBUS-DP



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.2Information and Training Center
Knowledge for Automation

Краткий обзор

Полевое оборудование, для автоматизации технических процессов, типа датчиков, приводов, преобразователей и двигателей, все чаще использует систему полевых шин для обмена информацией с блоками управления более высокого уровня.

PROFIBUS - система полевых шин, которая может использоваться всем оборудованием автоматизации, типа PLC, PC, HMI-систем, приводов и датчиков для обмена данными.

PROFIBUS-DP

PROFIBUS-DP - протокол, оптимизированный по скорости, был специально разработан для связи между PLC (DP мастера) и устройствами распределенного ввода - вывода (DP slave).

PROFIBUS-DP - дешевая, гибкая замена передачи сигналов по параллельной линии 24V и 20mA .

PROFIBUS-DP основывается на DIN 19245 части 1 и пользовательских расширениях, определенных в DIN 19245 части 3. В ходе европейского процесса стандартизации полевых шин, PROFIBUS-DP был объединен в Европейский стандарт полевых шин EN 50170.

Мастера

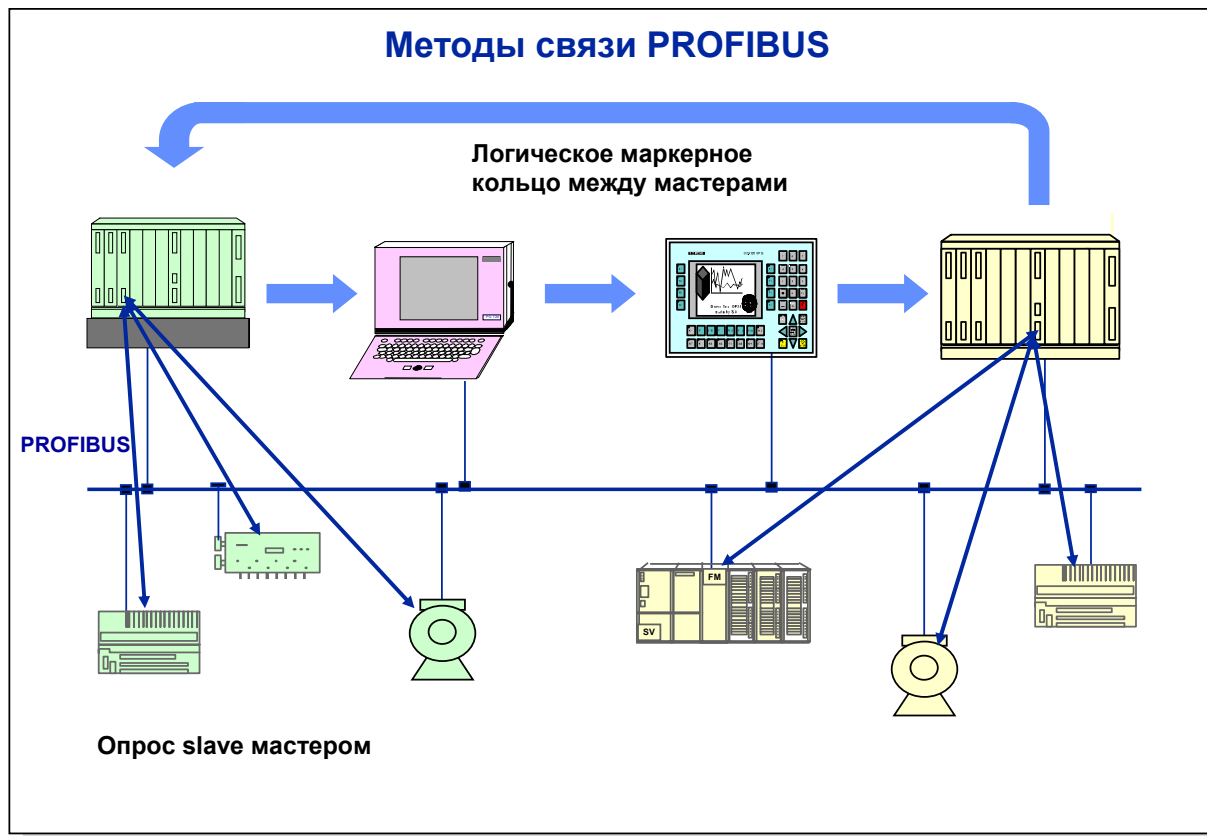
PROFIBUS делает различие между мастерами (masters) и slave.

PROFIBUS мастера отвечают за движение данных по шине. Мастер может посылать сообщения, когда он владеет маркером (token). Владение маркером дает право мастеру на доступ к шине.

Мастера называются также в протоколе PROFIBUS активными узлами.

Slave

PROFIBUS slave - простые устройства ввода - вывода, типа приводов, датчиков, преобразователей и т.д. Они не получают маркер, то есть они могут только подтверждать получение сообщений или посылать сообщения мастеру по запросу. Slave называются также пассивными узлами.



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.3Information and Training Center
Knowledge for Automation

Управление доступом к шине Метод управления доступом к шине определяет, когда узел может посылать данные. Это необходимо, чтобы в любой момент только один узел имел право послать данные.

PROFIBUS-протокол осуществляет два основных требования, положенные в основу метода управления доступом к шине:

- Для связи между сложными станциями равного статуса (мастерами) должно быть обеспечено, что каждая из этих станций имеет достаточную возможность для решения коммуникационных задач в заданном интервале времени.
- Для связи между мастером и slave должен быть обеспечен циклический обмен данными в режиме реального времени.

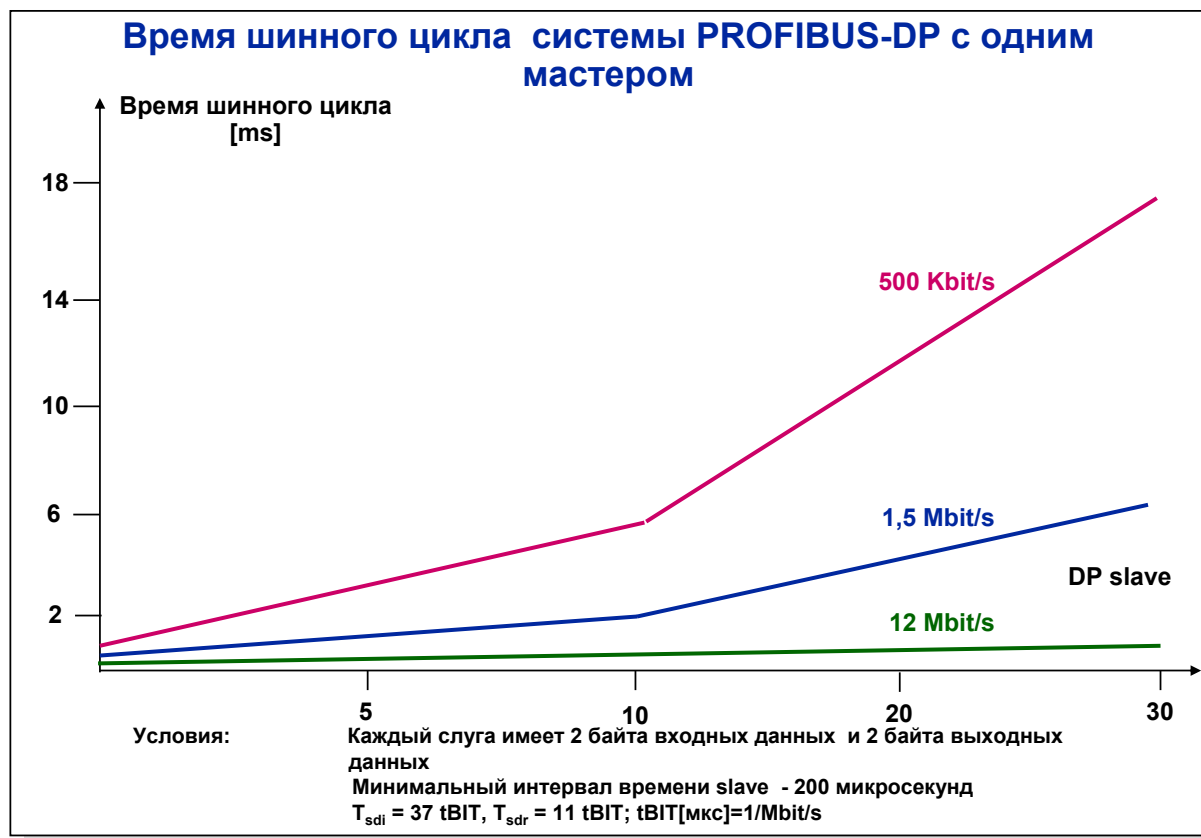
PROFIBUS-метод управления доступом к шине поэтому использует маркер (token) для связи между мастерами и принципом мастера- slave для связи между мастерами и простыми устройствами ввода - вывода (slave).

Метод обмена маркером

Владение маркером гарантирует право на доступ к шине в пределах точно определенного интервала времени.

Маркер - это специальное сообщение, которое посылается от одного мастера другому, чтобы дать каждому мастеру по очереди право на доступ к шине в пределах максимального времени обращения маркера.

Принцип мастер-слуга Принцип мастер-слуга позволяет мастеру, который в настоящее время владеет маркером, опрашивать слуг, назначенных данному мастеру. Мастер может посылать сообщения слугам или принимать сообщения от слуг.



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.4Information and Training Center
Knowledge for Automation**PROFIBUS-DP**

PROFIBUS-DP протокол разработан для быстрого обмена данными на уровне датчиков / привода. На этом уровне центральные блоки управления, типа PLC, связываются с распределенными входами и выходами через быстродействующую последовательную связь. Обмен данными с этими распределенными устройствами главным образом циклический.

Центральный диспетчер (мастер) читает входные данные от slave и пишет выходную информацию slave. Время цикла шины должно быть короче, чем время цикла PLC.

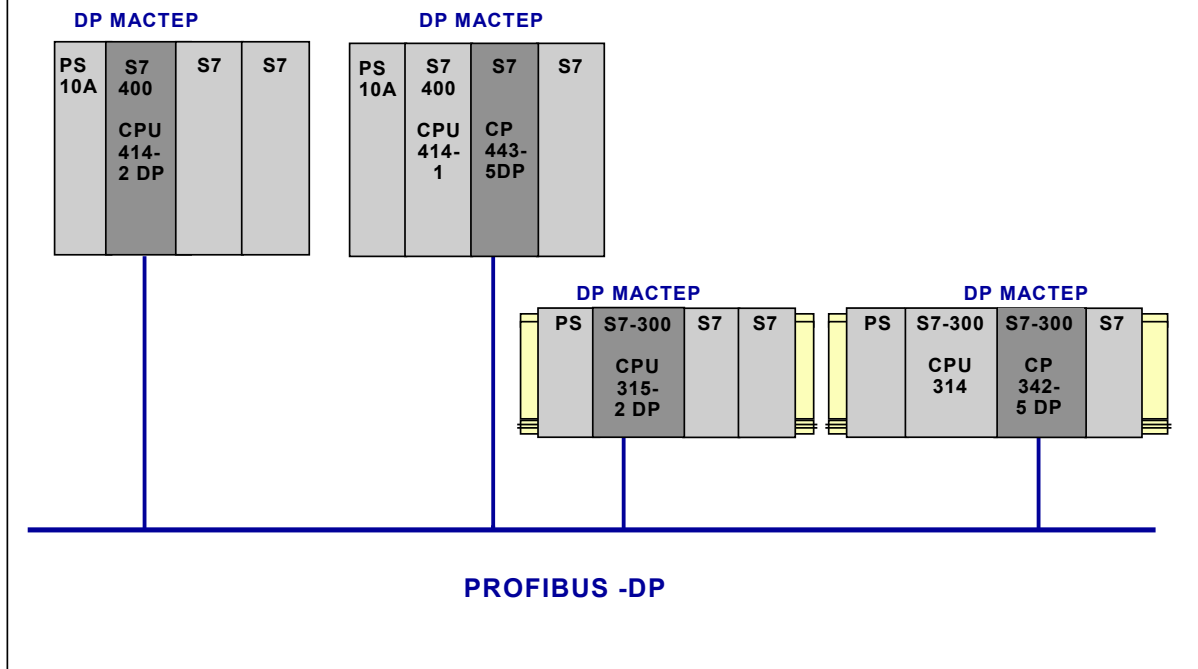
Обратите внимание PROFIBUS-DP протокол *не может* использоваться для обмена информации между мастерами.

Скорость

Для передачи 512 бит входных данных и 512 бит выходных данных разделенных между 32 узлами PROFIBUS-DP требует приблизительно 6 ms при скорости передачи 1.5 Mbit/s и меньше, чем 2 ms при 12 Mbit/s.

Гораздо более высокая скорость этого протокола по сравнению с таким протоколом, как PROFIBUS-FMS - главным образом вследствие того, что входные и выходные данные передаются в одном цикле сообщения, используя уровень 2 модели стека протоколов OSI (Link Layer).

PROFIBUS мастера в SIMATIC S7



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.5



Information and Training Center
Knowledge for Automation

Краткий обзор

SIMATIC S7/M7 продолжает тенденцию к распределенной автоматизации, объединяя распределенный ввод - вывод в системе автоматизации. Технология автоматизации SIMATIC S7/M7 формирует идеальное товарищество с международно признанными полевыми шинами PROFIBUS-DP/PA и распределенными станциями ввода - вывода.

PROFIBUS мастер

PLC S7-300 и S7-400 могут быть связаны с PROFIBUS как мастера или через встроенный в CPU интерфейс PROFIBUS-DP или через коммуникационные процессоры (CP).

CPU со встроенным интерфейсом PROFIBUS-DP позволяют Вам создавать распределенные системы автоматизации со скоростями передачи до 12 МБод.

Интеграция

Полная интеграция системы PLC и распределенных входов - выходов имеет следующие преимущества для пользователя:

- Однородное конфигурирование: Вы конфигурируете и центральные и распределенные входы - выходы с помощью STEP 7. Это означает, что инструмент конфигурирования одинаков для пользователя, независимо от типа решения автоматизации.
- Централизованное и распределенное программирование: Ваша программа для PLC пишется с помощью STEP 7 независимо от типа конфигурации. Это означает, что Вы можете писать программу не принимая во внимание заключительную конфигурацию аппаратных средств целевой машины.
- Одинаковая работа системы, независимо от того имеет ли она централизованную или распределенную конфигурацию периферии: SIMATIC S7/M7 предлагает мощную поддержку системы. Она включает программное обеспечение - параметризацию входов - выходов, широкий диапазон средств обслуживания, диагностики.
- Программирование, испытание и запуск через PROFIBUS-DP: Со STEP 7 Вы можете проверить программу и запустить ее на центральном PLC через полевою шину так легко, как Вы можете это сделать через центральный порт устройства программирования на CPU.

Доступные DP slave



ET 200M



ET 200U

Составные модули, состоящие из интерфейсного модуля и модулей из набора S7-300 (ET 200M) или из набора S5 (ET 200U).



ET 200B



ET 200L

Маленькие, компактные станции ввода - вывода (степень защиты IP 20) с объединенными входными и выходными каналами.



ET 200X



ET 200S

Интерфейсный модуль плюс модули входа /выхода. Степень защиты: ET 200X: IP 65/67, ET 200S: IP 20.



CPU 215



CPU 315-2 DP

Интеллектуальные DP модули: S7-200 и S7-300. Для предварительной обработки данных.



CPU 316-2 DP



CPU 318-2 DP



CP 342-5

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.6



Information and Training Center
Knowledge for Automation

Модульные slave ET 200M

ET 200M состоит из интерфейсного модуля IM153-1, который связан с S7/M7-PROFIBUS мастером. Все S7-300 модули, адресуются через P-шину и могут быть вставлены в ET 200M.

Максимальный размер адресного пространства в ET 200M: 128/128 байты к для входов / выходов с максимальной скоростью 12 МБод.

Компактные slave ET 200L и ET 200B

ET 200L и ET 200B состоят из терминального блока и блока электроники. Имеются блоки электроники с цифровыми и аналоговыми каналами.

ET 200L используется, где требуется немного входов и выходов со скоростью до 1.5 МБод. ET 200B используется, где имеется ограниченная глубина установки. Максимальная скорость - 12 МБод.

Компактный slave ET 200C

Компактный модуль ET 200C с высокой степенью защиты (IP66/IP77).

С максимумом скорости до 12 МБод для цифровых входов / выходов и до 1.5 МБод для аналоговых входов / выходов

Составные slave ET 200X

ET 200X - компактная станция ввода - вывода с высокой степенью защиты IP 65/IP 67 - состоит из основного модуля и модулей расширения (например, модули входа / выхода, мастера AS - интерфейса, пневматические модули, источник питания SITOP).

Составные slave ET 200S

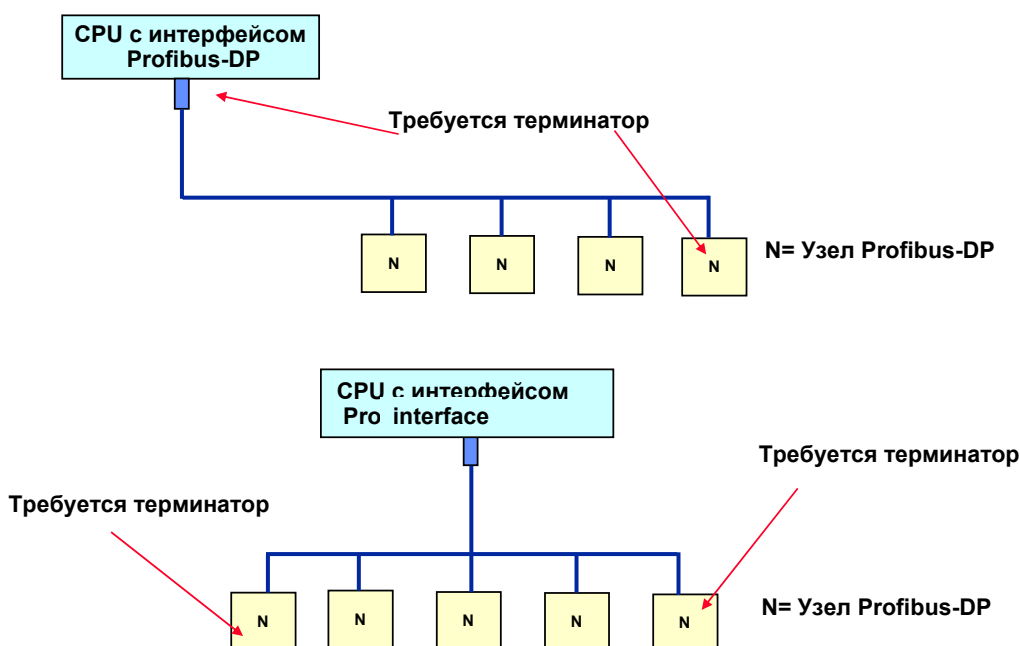
ET 200S - распределенная станция ввода - вывода со степенью защиты IP 20. Высокая модульность позволяет его быстро приспособлять к любому применению .

ET 200S состоит из интерфейсного модуля PROFIBUS-DP, цифровых и аналоговых модулей, функциональных технологических модулей (например, счетчик, позиционер) и load feeders.

Интеллектуальные slave

Например, CPU 315-2, CP 342-5 или S5-95-PROFIBUS с функциональными возможностями слуги.

Терминатор PROFIBUS-DP



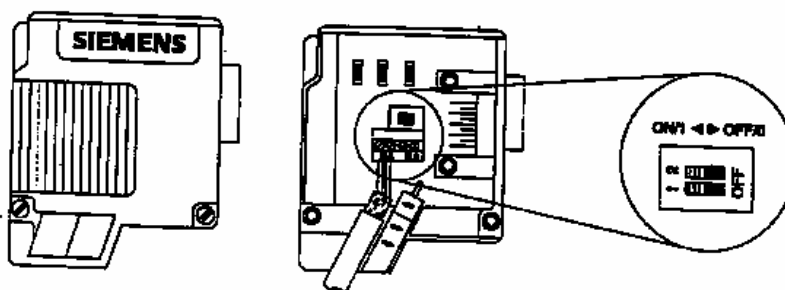
SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.7Information and Training Center
Knowledge for Automation

Установка терминатора

Вы должны включить терминатор на первом и последнем узлах сегмента. Чтобы это сделать, откройте крышку шинного коннектора и установите выключатель в позицию ON (см. слайд)



PROFIBUS правильно закончен, если только фактически включено электропитание узла, в который вставлен терминатор. Если это не так, PROFIBUS может также быть закончен активным RS485 терминатором (6ES7972-0DA00-0AA0). Терминатор тогда получает постоянное электропитание, отдельное от других компонентов ввода - вывода.

Подключение терминаторов к шине позволяет подсоединять и разъединять по желанию узлы (например, ET 200L), без возникновения сбоев.

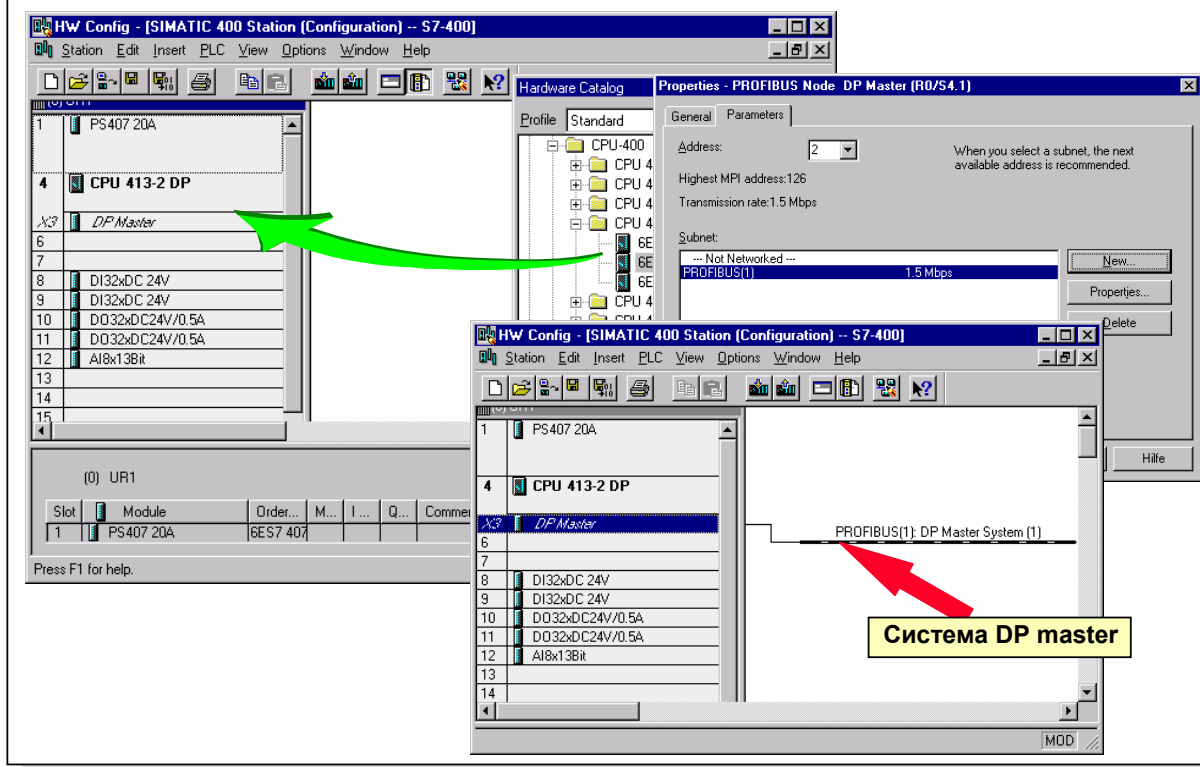
Длина кабеля

Максимальная длина сегмента Profibus зависит от скорости передачи данных:

Скорость	Длина сегмента
От 9.6 до 187.5 КБод	10000 м.
500 КБод	400 м.
1.5 МБод	200 м.
От 3 до 12 МБод	100 м.

Максимальная длина сегмента для MPI - 50 м. В сети может быть до 9 репитеров.

Конфигурирование системы DP Master



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.8



Information and Training Center
Knowledge for Automation

Распределенный ввод - вывод

Все системы, состоящие из DP мастера и DP slave, связанные через шинный кабель и общающиеся через протокол PROFIBUS-DP, называются распределенным вводом - выводом.

DP мастер

В качестве DP мастера Вы можете устанавливать:

- S7-CPU со встроенным DP- интерфейсом (например, CPU 414-2, и т.д.)
- Интерфейсный модуль, который назначен на M7-CPU/M7-FM
- CP для связи с CPU (например. CP 443-5, и т.д.)

При установке DP-мастера

Чтобы создать систему мастера, поступайте следующим образом:

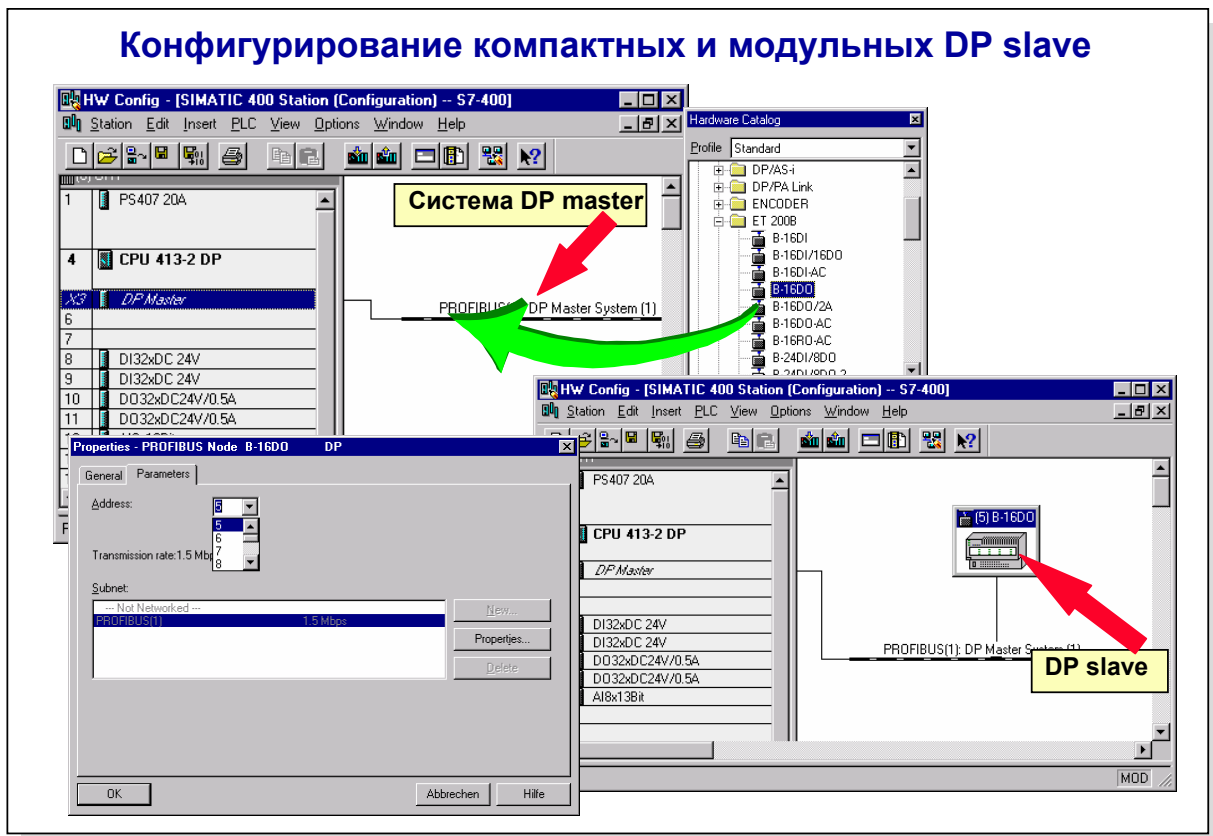
1. Выбрать DP-мастера из окна "Hardware Catalog" .
2. Используя Drag&Drop, перетяните модуль в допустимую линию стойки. Откройте диалоговый бокс "Properties - PROFIBUS Nodes". В этом диалоге Вы можете устанавливать следующие свойства:
 - Создавать новую подсеть PROFIBUS или выбрать существующую.
 - Устанавливать свойства PROFIBUS подсети (скорость передачи и т.д.).
 - Устанавливать адрес мастера PROFIBUS DP .
3. Подтвердить назначения кнопкой "OK"..Появляется следующий символ для DP-мастера:

Этот символ используется как "вешалка" для DP- slave.

Обратите внимание

Вы можете иметь на одной PROFIBUS-DP подсети одного мастера или несколько мастеров (моно и мульти мастерные системы) . При моно-мастерной системе используется только один DP мастер на одной PROFIBUS подсети, в мульти-мастерной системе используются на одной PROFIBUS подсети несколько DP-мастеров с соответствующей каждому из них системой slave.

Конфигурирование компактных и модульных DP slave



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.9



Information and Training Center
Knowledge for Automation


DP slave

- Slave с объединенными цифровыми / аналогом входами и выходами (компактные DP-модули, например, ET200B).
- Интерфейсные модули с назначенными S5 или S7 модулями (модульные DP устройства, например, ET200M).
- S7-200/300 станции с модулями, которые поддерживают функцию "Интеллектуальный слуга" (например, CPU 215-DP, CPU 315-2).

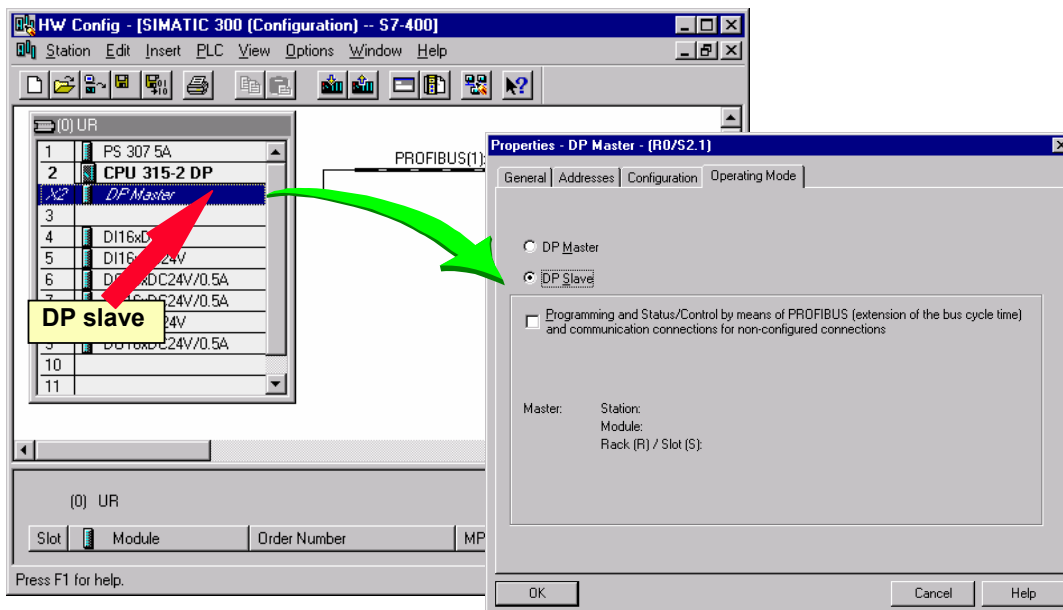
Установка

DP-slave

Чтобы сконфигурировать DP-slave, поступайте следующим образом:

1. Выбрать желаемый компактный DP-slave (например, ET200B) или интерфейсный модуль (Например, IM153 для ET200M) для модульного устройства из "Hardware Catalog".
2. Перетянуть символ на символ мастера .
Открыть диалоговый бокс "Properties - PROFIBUS Nodes". Здесь Вы можете выбрать:
 - Свойства PROFIBUS подсети (скорость и т.д.).
 - Адрес PROFIBUS DP-устройства.
3. Подтвердить назначения с "О.К.". Конфигурационная таблица приложена к символу, который представляет дополнительный ввод - вывод компактного slave или стойки модульного устройства.
4. Вы вставляете нужные модули из "Hardware Catalog" в конфигурационную таблицу.

Конфигурирование интеллектуальных DP slave (на примере CPU 315-2)



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.10Information and Training Center
Knowledge for Automation

Интеллектуальный slave

Главное назначение интеллектуального DP-slave - проводить предварительную обработку входных данных перед их передачей мастеру или выходных данных перед их передачей простым slave.

При работе с интеллектуальным DP- slave, DP-мастер производит доступ не ко входам / выходам интеллектуального DP- slave, а скорее к адресным областям "предварительной обработки" CPU. Программа пользователя предварительной обработки CPU должна заботиться об обмене данными между адресной областью и областью входа / выхода.

Обратите внимание

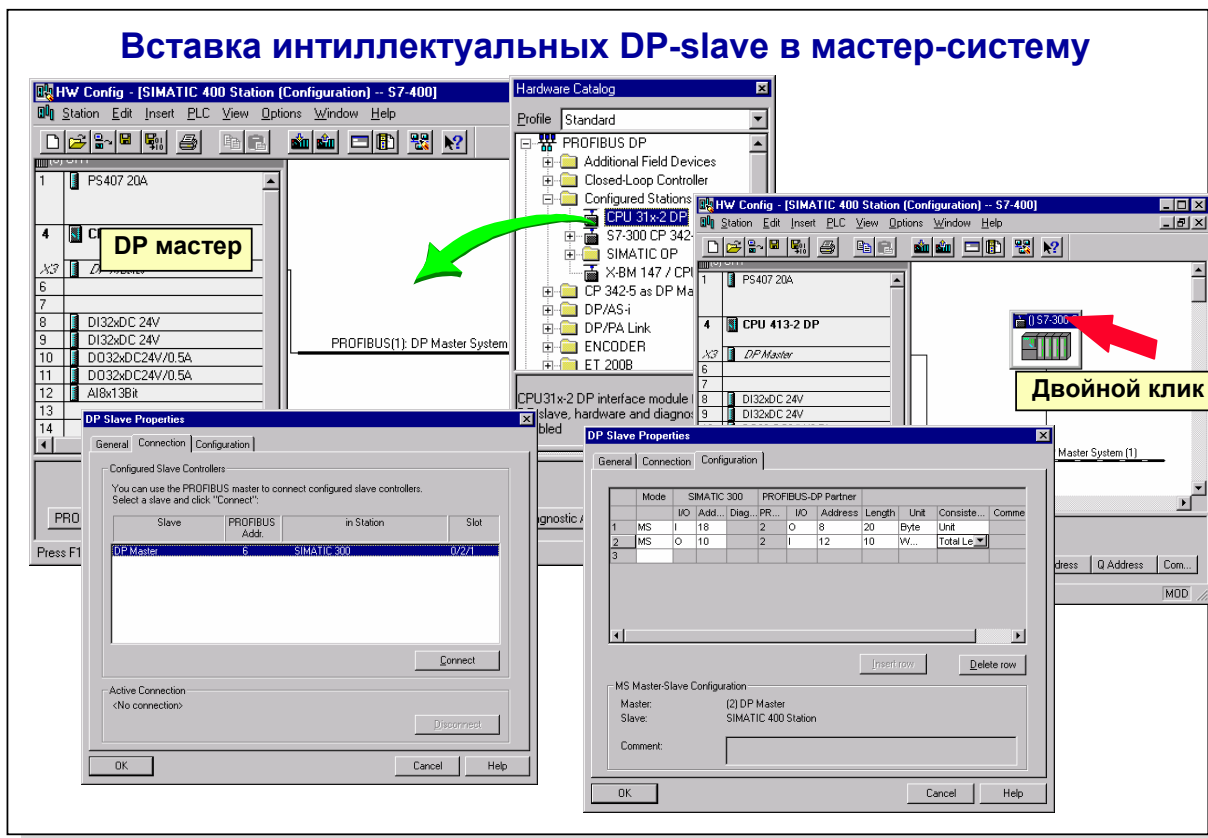
Интеллектуальный DP- slave (например, CPU 315-2 DP) не может одновременно быть сконфигурирован как DP-мастер и как DP- slave. CPU 315-2 DP, сконфигурированный как DP- slave, не может одновременно быть DP-мастером для других интеллектуальных DP- slave.

Конфигурирование DP- slave

Чтобы сконфигурировать CPU 315-2 DP как интеллектуальный DP- slave, сделайте следующее:

1. Вставьте S7-300 станцию в Ваш проект.
2. Откройте HW-Config, выбирая для данной станции,
3. Вставьте CPU 315-2 DP из "Hardware Catalog" в конфигурационную таблицу.
4. Двойным щелчком на линии 2.1 в конфигурационной таблице открыть диалоговый бокс "Properties - DP Master".
5. Установить бокс выбора "Use Controller as Slave" на "Slave Configuration".
6. Определить другие PROFIBUS -параметры CPU 315-2-DP.
7. Подтвердить назначения с "O.K"..

Вставка интеллектуальных DP-slave в мастер-систему



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.11



Information and Training Center
Knowledge for Automation

Вставка интеллектуального DP slave

Чтобы назначить CPU 315-2 DP как интеллектуальный DP slave мастеру, поступайте следующим образом:

1. Вставьте в Ваш проект станцию, способную быть DP мастером (например, S7-400).
2. Откройте HW Config для данной станции.
3. Вставьте DP-мастера (например, CPU 414-2 DP) из "Hardware Catalog" в конфигурационную таблицу.
4. Двойным щелчком на линии "DP Master" в конфигурационной таблице открыть диалоговое окно "Properties - DP Master".
5. Определить все PROFIBUS параметры DP-мастера, и сохранить их, нажав кнопку "OK".
6. Используя Drag&Drop, тяните CPU 315-2 DP из "Hardware Catalog" к системе мастера.
7. Двойным щелчком по DP slave (по названию DP slave, то есть "CPU 315-2 DP") и выбрать таблицу "Connection".
Появляется список всех сконфигурированных интеллектуальных DP-slave.
8. Выбрать нужного интеллектуального DP-slave, и щелкнуть на кнопке "Connect".
9. Выбрать таблицу "Slave Configuration", и назначить мастеру и slave области, через которые они будут обмениваться данными.
10. Подтвердить назначения кнопкой "O.K".

Обратите внимание

Для правильной работы DP-мастера и интеллектуального DP-slave, должны быть загружены в соответствующие CPU OB ошибки (OB 85, OB 86, и т.д.).

Анализ ошибок в OB 86 при отказе модуля-slave

The screenshot shows the SIMATIC Manager interface with the title bar 'LAD/STL/FBD - [OB86 -- S7-400\SIMATIC 400 Station\CPU 413-2 DP]'. Below the menu bar is a toolbar. The main window is divided into two panes. The top pane displays a table of OB86 parameters:

Address	Decl.	Name	Type	Initial	Comment
0.0	temp	OB86_EV_CLASS	BYTE		16#38/39 Event clas
1.0	temp	OB86_FLT_ID	BYTE		16#C1/C4/C5, Fault
2.0	temp	OB86_PRIORITY	BYTE		26/28 (Priority of
3.0	temp	OB86_OB_NUMBER	BYTE		86 (Organization b.
4.0	temp	OB86_RESERVED_1	BYTE		Reserved for system
5.0	temp	OB86_RESERVED_2	BYTE		Reserved for system
6.0	temp	OB86_MDL_ADDR	WORD		Depending on fault
8.0	temp	OB86_RACKS_FLTD	ARRAY[0..31]		
*0.1	temp		BOOL		
12.0	temp	OB86 DATE TIME	DATE AND TIME		Date and time OB86

The bottom pane shows a ladder logic network for 'OB86 : "Loss Of Rack Fault"'. The network is titled 'Network 1: Title:'. The logic consists of two normally open contacts in series: '#OB86_FLT_ID' and 'B#16#C4', followed by a coil 'M 0.4'. A comment '//DP-Station failure' is placed to the right of the first contact. The status bar at the bottom indicates 'Press F1 for help.', 'Offline', 'Abs', 'Nw 1 Ln 4', 'INS', and 'MOD'.

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.12Information and Training Center
Knowledge for Automation

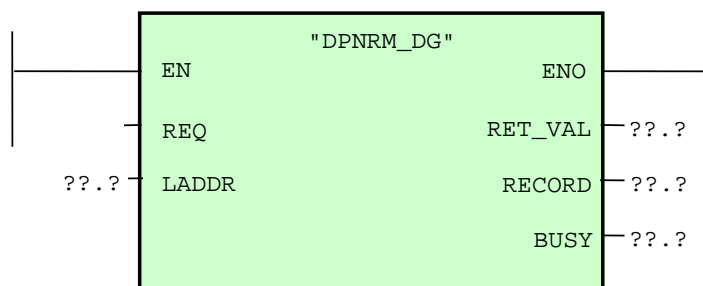
Отказ станции

Операционная система CPU (CPU 315-2DP или S7-400) активизирует OB86, если обнаружен отказ стойки, подсети или распределенной станции ввода - вывода. Это событие идентифицируется, как приходящее.

Если Вы не запрограммировали OB86, и возникает такая ошибка, CPU переходит в режим STOP.

- Переменные в OB86**
- OB86_FLT_ID: B#16#C4 // выход из строя DP-станции.
 - OB86_FLT_ID: B#16#C5 // сбой DP-станции
 - OB86_MDL_ADDR: Логический базовый адрес DP-мастера (диагностический адрес)
 - OB86_RACKS_FLTD: == > замените тип данных на DWORD
Содержание(для OB86_FLT_ID= B#16#C4 или B#16#C5):
 - Биты от 0 до 7: Номер DP-станции (PROFIBUS адрес)
 - Биты от 8 до 15: Идентификатор мастер-системы DP
 - Биты от 16 до 30: Логический базовый адрес DP- slave (диагностический адрес)
 - Бит 31: Идентификатор ввода - вывода

Диагностика slave с помощью SFC 13 (DPNRM_DG)



Параметры	Объявления	Тип данных	Обл. памяти	Описание
REQ	INPUT	BOOL	I, Q, M, D, L, Const.	REQ = 1: Запрос на чтение
LADDR	INPUT	WORD	I, Q, M, D, L, Const.	Диагностический адрес DP slave
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Если происходит ошибка во время обработки функции, возвращаемое значение содержит код ошибки. Если ошибки нет, RET_VAL содержит длину фактически переданных данных.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Область для чтения диагностических данных. Допустимый тип данных BYTE. Минимальная длина данных - 6 байт.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY = 1: Чтение еще не закончено.

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.13Information and Training Center
Knowledge for Automation

Диагностика slave

С помощью SFC 13 "DPNRM_DG" Вы читаете диагностические данные DP slave в форме, предусмотренной в EN 50 170.

Если в течение передачи не происходит ошибки, читаемые данные выводятся в область, назначенную в выходе RECORD (OUT2).

Вы запускаете функцию SFC 13, назначая 1 входному параметру REQ (IN0).

Структура диагностической информации

Основная структура диагностических данных slave показывается в следующей таблице

Для дальнейшей информации пожалуйста см. руководства для DP-slave (например, номера ошибок в NCM-S7 руководстве).

Основная структура диагностической информации

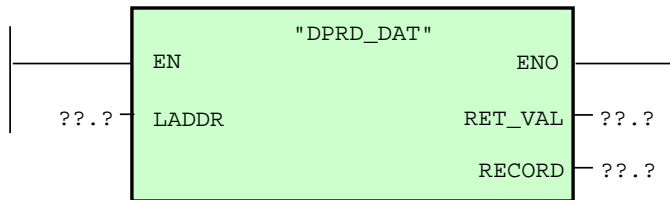
Байт	Значение
0	Статус станции 1
1	Статус станции 2
2	Статус станции 3
3	Номер мастер-станции
4	Идентификатор изготовителя (младший байт)
5	Идентификатор изготовителя (старший байт)
6 ...	Специфическая для slave диагностика

Обратите внимание

В случае стандартных slave, для которых число стандартных диагностических данных является большим, чем 240 байтов и не больше, чем 244 байта, первые 240 байтов записываются в область предназначения, и устанавливается бит переполнения (OV).

Чтение последовательных данных из стандартного DP slave с помощью SFC 14

- SFC 14 "DPRD_DAT" применяется, чтобы прочитать больше, чем четыре последовательных байта данных (консистентных данных).



Параметр	Объявл.	Тип данных	Обл. памяти	Описание
LADDR	INPUT	WORD	I, Q, M, D, L, Const.	Начальный адрес области входов модуля, из которого данные должны читаться.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Если происходит ошибка во время обработки функции, параметр содержит код ошибки.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Область предназначена для чтения данных пользователя. Она должна иметь точно ту же длину, что и область, которую Вы сконфигурировали для выбранного модуля в STEP7. Допустимый тип данных BYTE.

SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.14



Information and Training Center
Knowledge for Automation

Функция

С помощью SFC 14 "DPRD_DAT" Вы читаете последовательные данные из стандартного DP slave.

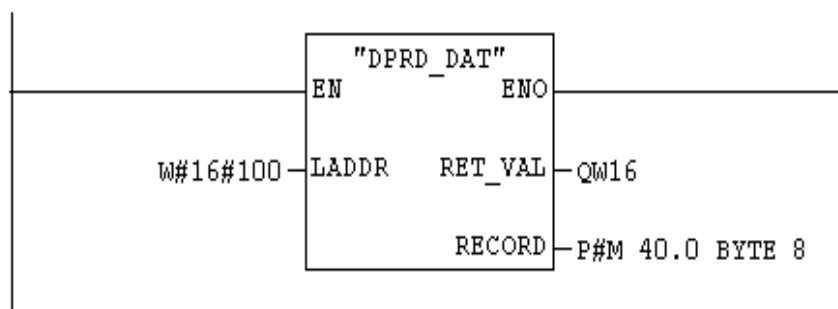
Длина данных должна составлять три или больше, чем четыре байта. Максимальная длина зависит от CPU. Вы найдете эти данные в технических спецификациях для вашего CPU. Если не происходит ошибки в течение передачи, читаемые данные вводятся в область, указанную в параметре RECORD

Назначенная там область должна иметь ту же самую длину, что и область, сконфигурированная Вами для данного модуля STEP 7.

Если стандартный DP slave имеет модульную конструкцию или имеет несколько DP идентификаторов, Вы можете получить в одном вызове SFC 14 доступ только к данным одного модуля/DП-идентификатора одновременно по назначенному адресу.

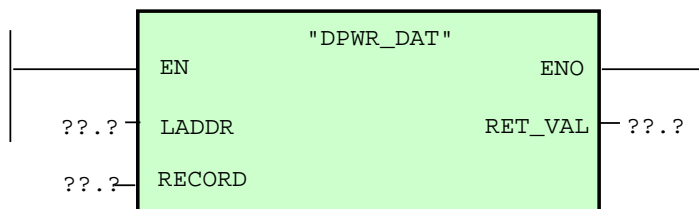
Пример

Network 1: Read 4 analog values from starting address 256



Запись последовательных данных из стандартного DP-slave с помощью SFC 15

- SFC 15 "DPWR_DAT" применяется, чтобы записать больше, чем четыре последовательных байта данных (консистентных данных).



Параметр	Объявл.	Тип данных	Обл. памяти	Описание
LADDR	INPUT	WORD	I, Q, M, D, L, Const.	Начальный адрес области входов модуля, в который данные должны писаться.
RECORD	INPUT	ANY	I, Q, M, D, L	Исходная область для данных пользователя, которые будут записаны. Она должна иметь точно ту же длину, что и область, которую Вы сконфигурировали для выбранного модуля в STEP7.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Допустимый тип данных BYTE. Если происходит ошибка во время обработки функции, параметр содержит код ошибки.

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.15Information and Training Center
Knowledge for Automation

Функция

С помощью SFC 15 "DPWR_DAT" Вы записываете последовательные данные из стандартного DP slave.

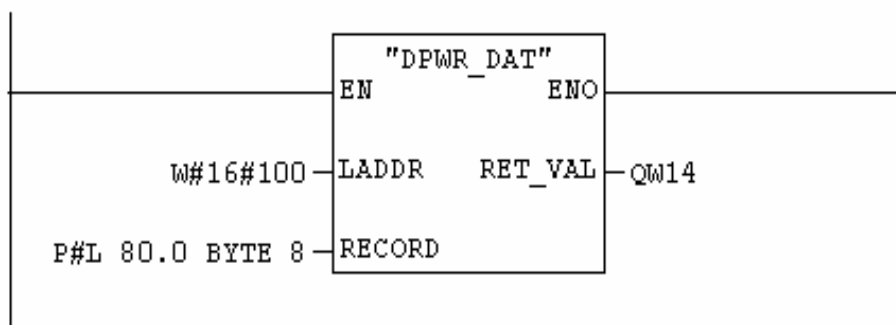
Длина данных должна составлять три или больше, чем четыре байта. Максимальная длина зависит от CPU. Вы найдете эти данные в технических спецификациях для вашего CPU. Если не происходит ошибки в течение передачи, читаемые данные вводятся в область, указанную в параметре RECORD

Назначенная там область должна иметь ту же самую длину, что и область, сконфигурированная Вами для данного модуля STEP 7.

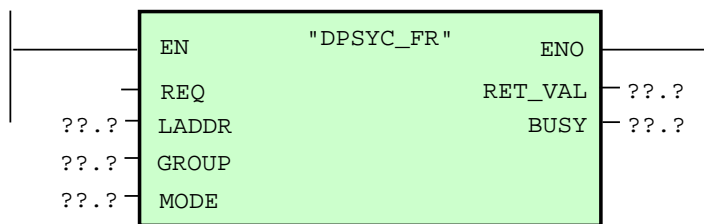
Если стандартный DP slave имеет модульную конструкцию или имеет несколько DP идентификаторов, Вы можете получить в одном вызове SFC 15 доступ только к данным одного модуля/DП-идентификатора одновременно по назначенному адресу.

Пример

Network 2: Output analog values to address 256



Синхронизация DP slave с помощью SFC 11 (DPSYC_FR)



Параметр	Объявл.	Тип данных	Обл. памяти	Описание
REQ	INPUT	BOOL	I, Q, M, D, L, Const.	Переключаемый уровень управл. параметра REQ=1: Триггер для SYNC/FREEZE работы
LADDR	INPUT	WORD	I, Q, M, D, L, Const.	Логический адрес DP-мастера
GROUP	INPUT	BYTE	I, Q, M, D, L, Const.	Выбор группы, Bit 0 = 1: Выбрана 1 группа Bit 1 = 1: Выбрана 2 группа... Bit 7 = 1: Выбрана 8 группа Вы можете выбирать несколько групп для одной работы.
MODE	INPUT	BYTE	I, Q, M, D, L, Const.	Идентификатор работы (закодированный согласно EN 50 170 V 3) Bit 0, 1, 6, 7: зарезервированы (значение 0) Bit 2 = 1: UNFREEZE выполняется ("НЕЗАМОРАЖИВАНИЕ") Bit 3 = 1: FREEZE выполняется ("ЗАМОРАЖИВАНИЕ") Bit 4 = 1: UNSYNC выполняется Bit 5 = 1: SYNC выполняется ("СИНХРОНИЗАЦИЯ")
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Код ошибки. Вы должны оценить RET_VAL после каждого выполнения блока.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY=1: Работа еще не закончена.

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.16Information and Training Center
Knowledge for Automation

Описание

С помощью SFC 11 "DPSYC_FR", Вы может синхронизировать одну или большее количество групп DP slave. Чтобы сделать это, Вы должны посылать одну из следующих команд управления или их комбинации к синхронизируемым группам:

- SYNC (одновременный выход и замораживание состояний выходов DPslave)
- UNSYNC (отменяет команду командуют SYNC)
- FREEZE (замораживание состояний входа DPslave и чтение замороженных входов)
- UNFREEZE (отмена команды FREEZE)

Предпосылки

Прежде, чем Вы посылаете вышеупомянутые команды управления, Вы, должны разделить DP slave на SYNC (СИНХРОНИЗИРУЕМЫЕ) или FREEZE (ЗАМОРАЖИВАЕМЫЕ) группы с помощью STEP 7.

Какой эффект от SYNC?

С помощью управляющей команды SYNC, DP slave названных групп переключаются в Sync-режим, то есть DP мастер передает текущие выходные данные и заставляет DP slave замораживать выходы. Когда они получают дальнейшие выходные сообщения, DP slave просто сохраняют выходные данные во внутреннем буфере; состояние выходов остается неизменным в настоящее время.

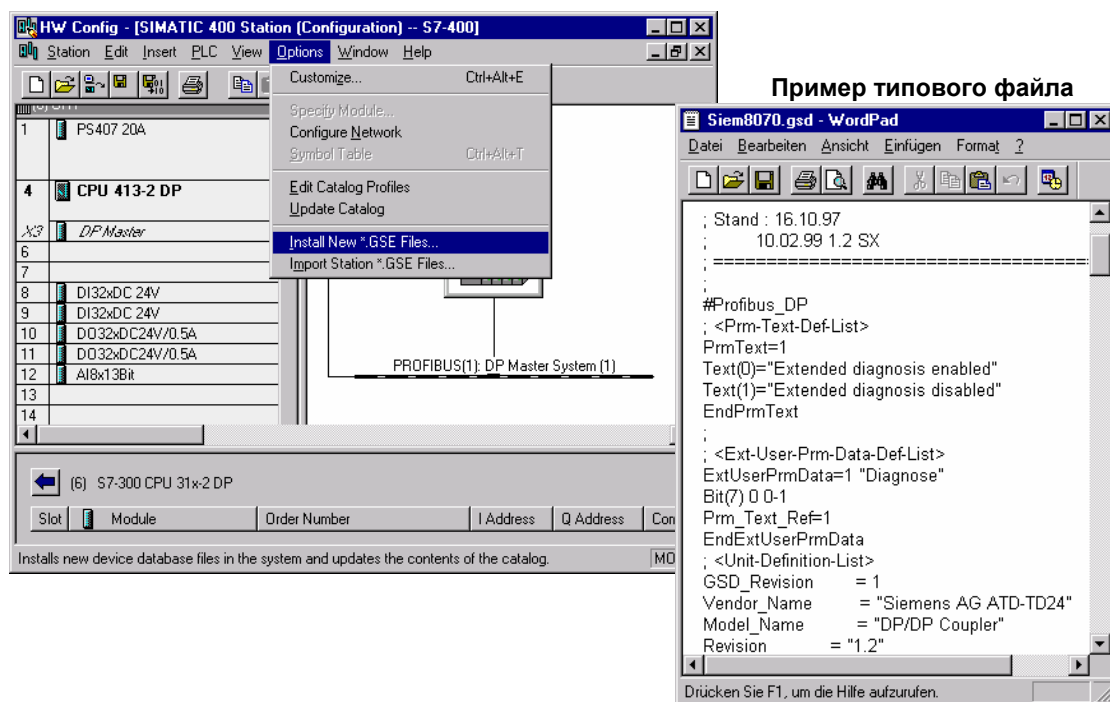
После каждой команды SYNC, DP slave выбранных групп размещают выходные данные из их буферов одновременно в периферийные выходы процесса (одновременный выход с сигналом управления).

Какой эффект от FREEZE?

С помощью команды управления FREEZE, DP slave переключаются в режим замораживания. Каждая команда FREEZE от DP мастера заставляет одновременно все выбранные DP slave сохранять текущее состояние входов. После этого DP мастер передает сохраненные данные в область входов CPU.

Входы или выходы циклически обновляются снова только когда послана команда UNSYNC или UNFREEZE.

Установка (доустановка) PROFIBUS-DP slave



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_12E.17



Information and Training Center
Knowledge for Automation

Типы файлов

STEP 7 необходим файл базы данных устройства (GSD) или файл типа для каждого DP slave для того, чтобы Вы могли выбирать slave из Hardware Catalog в HW-Config.

GSD файл содержит все свойства DP slave в соответствии со стандартом PROFIBUS. Файлы типа должны быть в соответствии со спецификациями Siemens.

Имеется файл типа для каждого типа DP slave SIEMENS AG.

DP slave от других изготовителей также снабжены GSD-файлом или файлом типа.

Внедрение DP slave

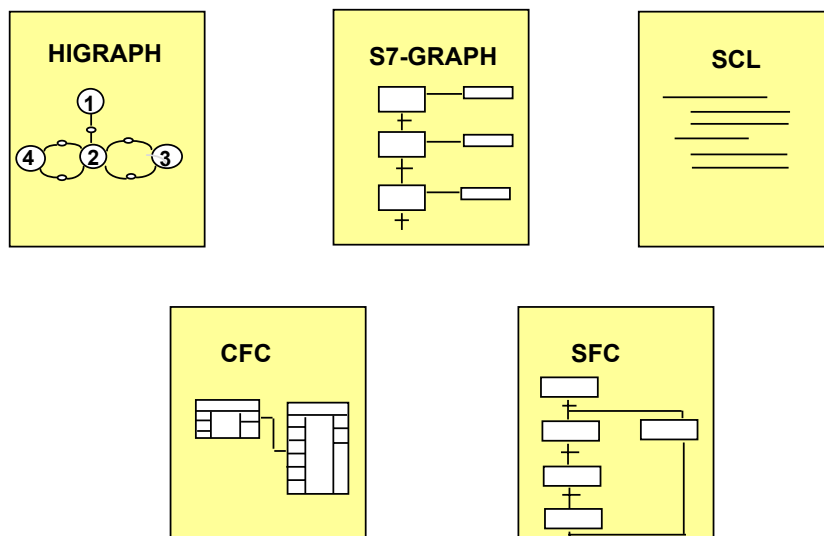
Вы можете вставить нового DP slave в Hardware Catalog следующим образом:

1. Выбрать меню *Options* -> *Install new GSD*.
2. В окне диалога, которое появляется при этом, открыть диск / директорию, содержащие нужный GSD файл.

Slave введен в окно "Hardware Catalog" (профиль каталога только "Standard" !) в разделе "PROFIBUS-Additional Field Devices" и доступен для конфигурирования.

Когда DP slave установлены или импортированы таким образом, существующие GSD файлы, и символы не удалены полностью, но сохранены в резервном справочнике \\ Step7\\S7data\\Gsd\\Bkp[No.]. No.- последовательный номер, который дается автоматически STEP 7.

Инжинеринговые пакеты для S7/M7



SIMATIC S7

Siemens AG 1999. All rights reserved.

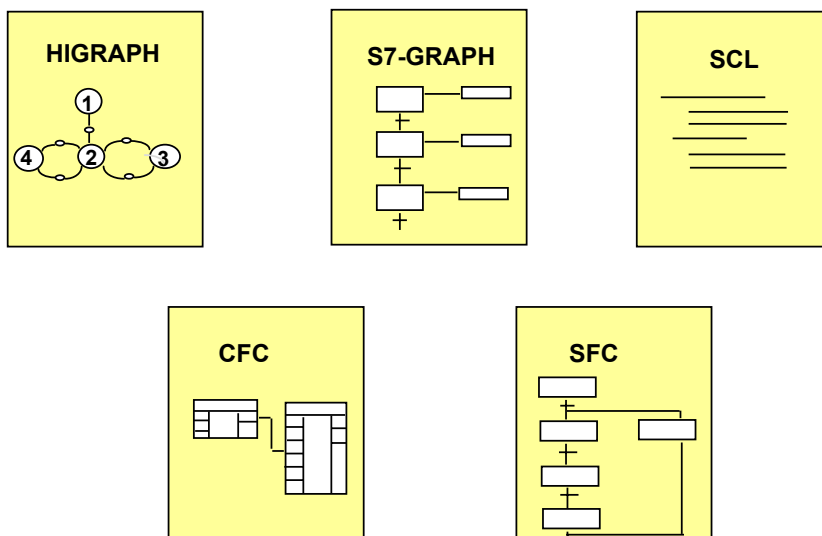
Date: 04.11.2005
File: PRO2_13D.1Information and Training Center
Knowledge for Automation

Содержание

Стр

The S7-GRAPH Software Package	3
Program Structure of a Sequential Control System	4
Creating a Sequencer FB	5
Sequencer Views	6
Elements of a Sequencer	7
Programming Actions	8
Standard Actions in a Step	9
Actions Dependent on an Interlock	10
Actions Triggered by an Event	11
Checking Conditions in Transitions, Interlocks and Supervisions	12
Permanent Instructions	13
Creating an Executable Block	14
Integrating an FB Call in OB1	15
Activating the Debugging Functions	16
The S7-HiGraph Software Package	17
Principle of the State Diagram Method	18
Elements of a State Diagram	19
Example: State Diagrams for an Elevator Controller	20
Creating a State Diagram	21
The HiGraph User Interface	22
Inserting States and Transitions	23
Programming Actions	24
Programming Transitions	25
Programming Permanent Instructions	26
Programming Graph Groups	27

Инжинеринговые пакеты для S7/M7



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.2Information and Training Center
Knowledge for Automation

Содержание

Стр

Assigning Actual Parameters	28
Message Exchange between State Diagrams	29
Assigning Actual Values for Messages	30
Saving and Compiling	31
Debug Functions in S7-HiGraph	32
Programming in the S7- SCL High-Level Language	33
Structure of an SCL Source File	34
The Declaration Part of a Block	35
The Statement Part of a Block	36
Expressions, Operators and Operands in S7-SCL	37
Statements in S7-SCL	38
Value Assignments in S7-SCL	39
The IF Statement in S7-SCL	40
The WHILE Statement in S7-SCL	41
Calling Function Blocks	42
The "OK" Flag for Error Evaluation	43
Compiling an SCL Source File	44
Continuous Monitoring	45
Setting and Editing Breakpoints	46
CFC for SIMATIC S7 and SIMATIC M7	47
CFC Chart	48
CFC Objects	49
Configuring CFC Applications instead of Programming	50
Integral Test and Debugging Functions	51
Configuring Sequential Control Systems with S7- SFC	52
Cooperation between CFC/SFC and SCL	53

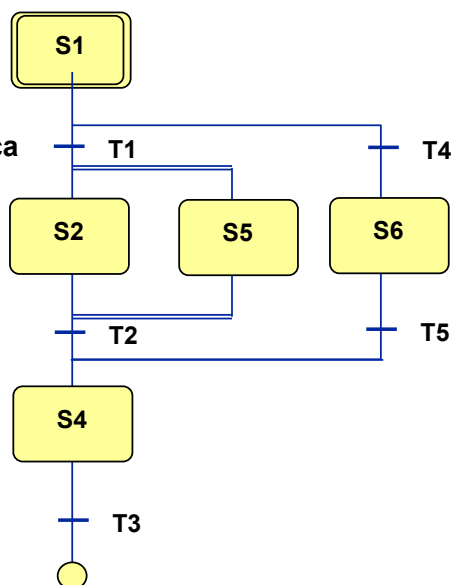
Программный пакет S7- GRAPH

S7-GRAPH: инструмент для программирования последовательностей

- Совместимость с IEC 1131-3
- Разработан для требований промышленности
- Графическое представление процесса через шаги и переходы
- Шаги содержат действия
- Переходы проверяют возможность попасть на следующий шаг

С S7-GRAPH Вы можете оптимизировать следующие стадии в проекте автоматизации :

- Планирование, конфигурирование
- Программирование
- Отладку
- Запуск
- Обслуживание, диагностику



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.3Information and Training Center
Knowledge for Automation

S7-GRAPH

With the S7-GRAPH programming language, you can clearly and quickly configure and program sequential sequences that you wish to control with an S7 PLC system.

The process is split into single steps with their own function scope. The sequence is represented graphically and can be documented with pictures and text.

The actions to be performed are defined in the single steps, the conditions for moving on to the next step are defined by transitions. The definitions of these, as well as interlocks and supervisions are written in a subset of the STEP 7 programming language LAD (Ladder Diagram).

S7-GRAPH for S7-300/400 is compatible with the sequential function chart language defined in the IEC 1131-3 standard.

Functionality

The following functions are offered:

- Several sequencers (max. 8) in the same S7-GRAPH function block
- Free number assignment (1 to 999) to the steps (max. 250 per sequencer complex) and transitions (max. 250)
- Parallel branches and alternative branches (each a max. of 250)
- Jumps (also to other sequencers)
- Starting/stopping of sequencers as well as activating/holding of steps.

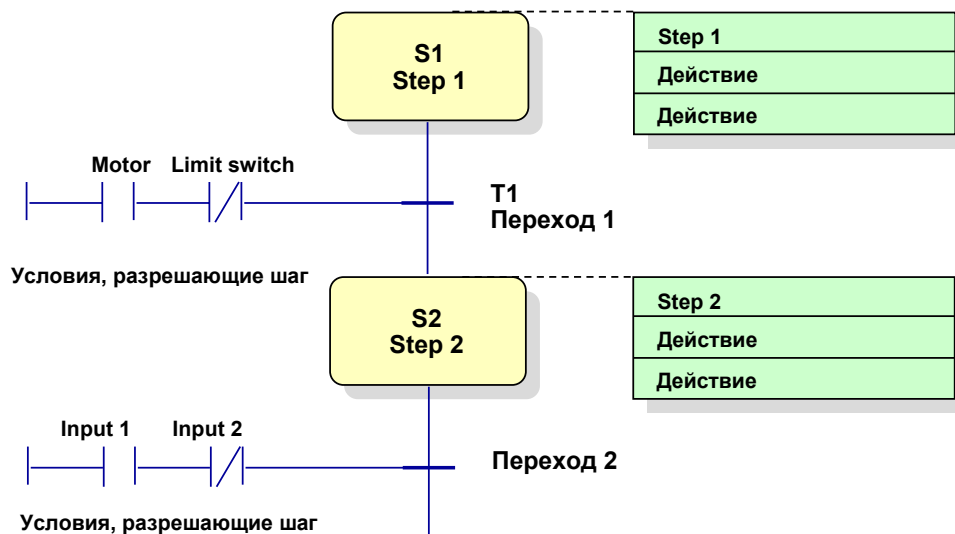
Test Functions

- Display of active steps or faulted steps
- Monitor Status and Modify Variable
- Switching between operating modes: manual, automatic and jogging mode

User Interface

- Overview, Single Page and Single Step representation
- Graphical distinction between interlock conditions (interlock, max. 32 conditions), actions (max. 100 per step) and supervision conditions (supervision, max. 32 conditions)

Структура программы последовательной системы управления



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.4Information and Training Center
Knowledge for Automation

Overview

A sequential control system is a control system that executes step by step. It switches from one step to the next when the step-enabling condition is fulfilled. A characteristic of a sequential control system is therefore the subdivision of the control task into

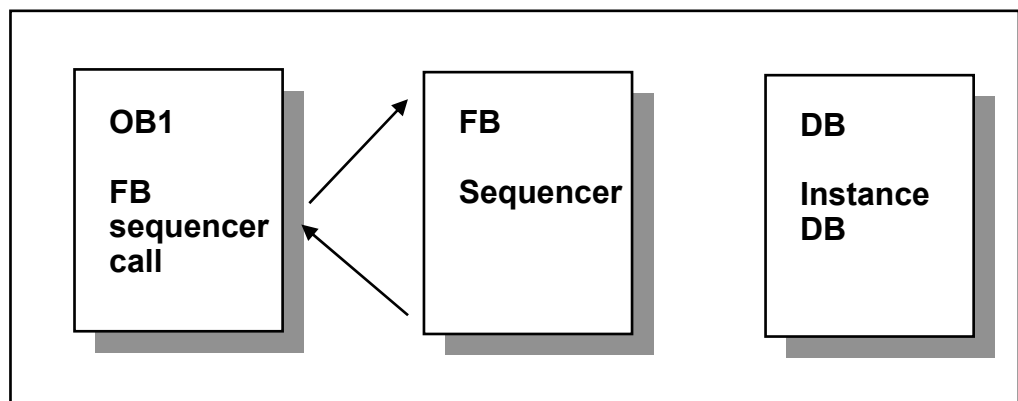
- Steps and
- Transitions (step-enabling conditions)

Sequencer

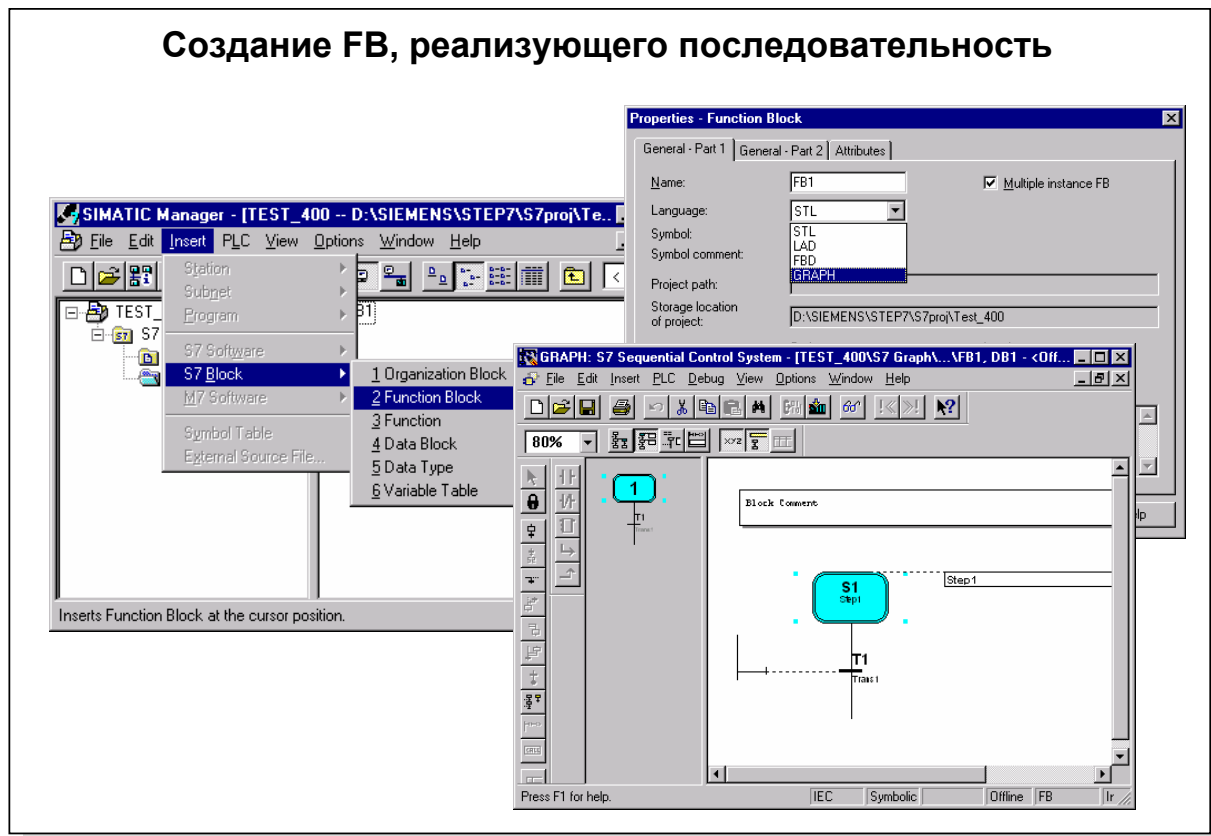
Steps and transitions form a sequencer. The sequencer is stored in an FB. An instance DB, that contains the sequencer data, is assigned to this FB.

At least three blocks are necessary for an executable program:

- the FB, that contains the sequencer(s)
- an instance DB with the sequencer data
- an organization block, function block or function containing the FB call. The parameters and the number of the instance DB are passed in the FB call.



Создание FB, реализующего последовательность



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.5



Information and Training Center
Knowledge for Automation

Overview

You can create S7-GRAPH FBs either with the SIMATIC Manager or with the S7-GRAPH Editor. In both cases, you must first create a project and a corresponding user program.

Creating an FB

To set up an S7-GRAPH FB with the SIMATIC Manager, proceed as follows:

1. Open the user program in which the FB is to be inserted.
2. Select the menu command *Insert -> S7 Block -> Function Block*
3. In the "Properties" dialog box enter the FB's number (e.g. FB1) and enter GRAPH as the programming language. Confirm with "OK"
4. Double-click the inserted block. The S7-GRAPH Editor is called and the block is opened.

S7-GRAPH Editor

In entering and editing a sequencer, you are supported by the S7-GRAPH Editor with context-sensitive functions. The toolbars contain icons which give you quick access to frequently used menu commands.

The following toolbars can be selected via the menu command *View -> Toolbar*:

Standard: Contains functions for file handling (Open, Save, etc.) and for editing (Copy, Insert, etc.).

Sequencer: Contains functions that insert sequencer elements in the program.

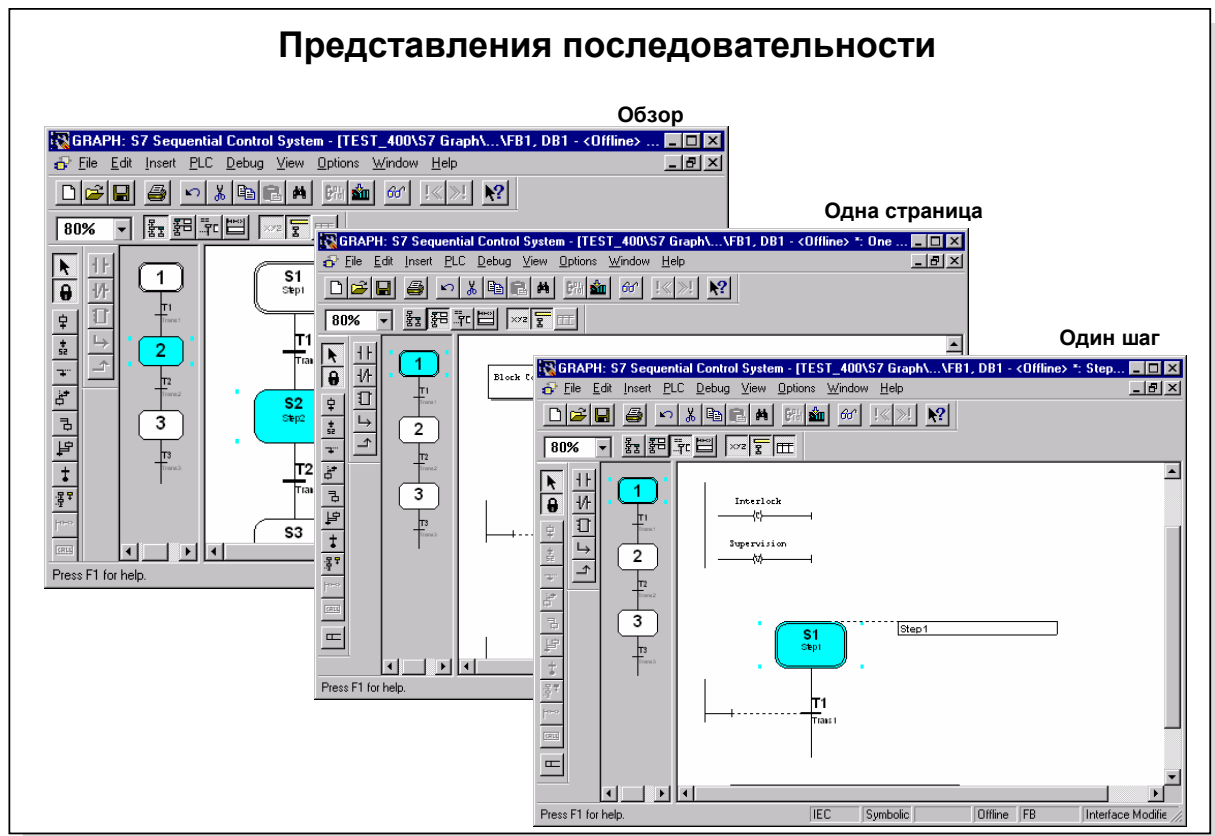
LAD: Contains functions that insert LAD elements in the program.

View: Contains functions for changing the view

Note

You can position the individual toolbars anywhere within the S7-GRAPH Editor. For this, click on the gray area of the toolbar and drag the bar to the desired position with the mouse.

Представления последовательности



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.6



Information and Training Center
Knowledge for Automation

Overview

The S7-GRAPH Editor allows you to edit a sequencer with all its elements, such as steps, transitions, branches, jumps, end-of-sequencer and comments, in different views.

You can select the view by choosing the following menu commands:

View -> Overview/Single Page/Single Step/Permanent Instructions
or from the toolbar.

Overview

This view gives you an overall impression of the sequential control system as a whole. This is especially suitable for configuring the sequential structure.

Several sequencers (max. 8) in one FB appear side by side. The overview contains the block comments and the names and numbers of the steps and transitions.

Single Page

Single page view shows actions, the numbers and names of the steps, the step-enabling conditions for the transitions, the numbers and names of the transitions and the block comments.

Several sequencers in one FB appear one below the other. This view is suitable both for configuring and for programming a sequential control system.

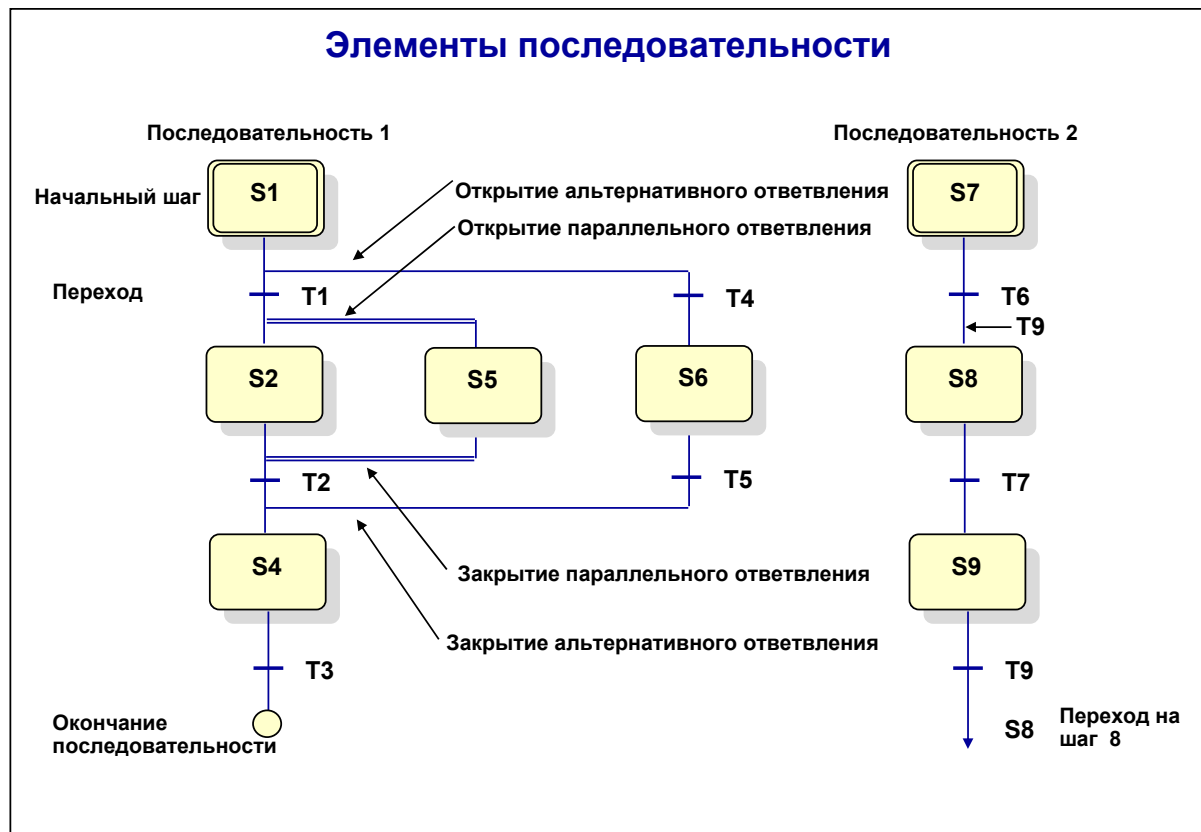
Single Step

This view shows one step with its transitions and all their contents. This view is used for programming not only actions and step-enabling conditions but also supervisions and interlocks. You can also edit step comments in this view.

Permanent Instructions

You use this window for programming "permanent instructions". Permanent instructions are conditions and/or block calls which come before or after the sequencer. They are executed once in every cycle regardless of the status of the sequencer.

Элементы последовательности



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.7Information and Training Center
Knowledge for Automation

Overview

When writing a sequencer, you are supported by functions for graphical programming. You can generate a sequencer structure simply, without requiring extensive programming experience, by arranging the S7-GRAPH structure elements to produce a graphical representation of the sequencer.

Structure of a Sequencer

A sequencer consists of a series of steps and transitions. The sequencer can be linear or branched.

- Within the steps, you formulate the instructions for the plant.
- The transitions contain the conditions for switching from one step to the next.

The following icons represent the elements a sequencer can consist of. You can select these icons from the toolbar



Step/Transition Pair



Jump



Alternative branch, open



End of sequencer



Alternative branch, closed



Parallel branch, open



Parallel branch, closed



Insert new sequencer

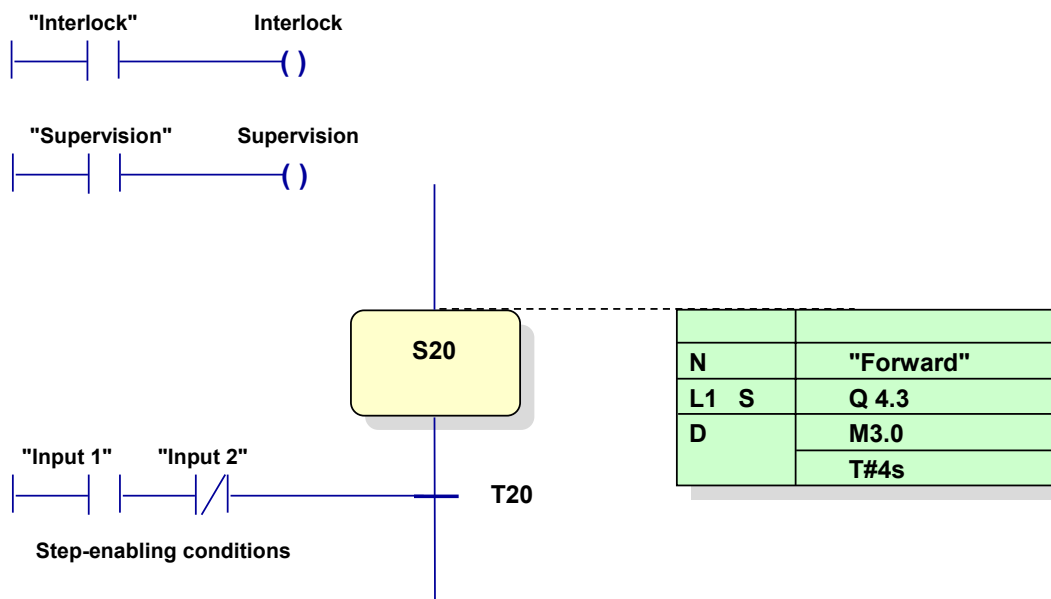
There are also icons for inserting permanent instructions and for changing the insertion mode.

Rules

The sequential control system structure must satisfy the following rules:

- Steps and transitions can only be inserted as pairs.
- Jumps can be added to the end of a branch and jump to a step either in the same sequencer or in a different sequencer.
- An end of sequencer symbol can be added following a transition at the end of a branch and terminates the execution of the branch.

Программирование действий (акций)



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.8Information and Training Center
Knowledge for Automation

Overview

The active parts of a step are the actions that are executed by the controller in the active step. In the steps, instructions are programmed which set outputs or activate or deactivate steps in the sequencer. An action consists of instruction and address and can be attached to a condition and/or combined with an event.

Programming

You program actions in single step or single page view. To do so, proceed as follows:

1. Select the table to the right of the step and press the Tab key.
2. Now program the actions by entering the instructions which are to be executed in the table:
 - Every instruction occupies exactly one line of the table
 - In the left-hand column are the instructions, in the right-hand column the addresses
 - If you use an instruction which requires time information (D or L), S7-GRAPH automatically sets up two lines in the right-hand column. Enter the time information in the lower line.
3. Press the Tab key once again to program a further action. The table is expanded by one line.

If you have used an instruction that is logically combined with a condition (all instructions that contain the letter C), then you must program the condition in single step view as an interlock.

Interlock

The interlock logic is used for conditional enabling of specific actions in a step. If the condition is fulfilled, then all actions conditioned with "C" are executed. Advancing to the next step takes place independent of the interlock condition.

Supervision

The supervision logic is used for recognizing supervision errors and the follow-up reactions (stopping the sequencer and possibly acknowledgement obligation). Advancing to the next step only takes place when the supervision conditions are not fulfilled and the error no longer exists.

Стандартные акции в шаге

Блок акций с простыми инструкциями

Action_block_1	
N	M1.1
S	M1.2
R	M1.3
D	M1.4
	T#1H2M3S
L	M1.5
	T#4MS
CALL	FC1

- N = Назначение Не-сохраняя
- S = Установить (Сохранить)
- D = Задержка время , назначать не-сохраняемую задержку времени T
- L = Ограничение времени, назначение не-сохраняемого на ограниченного времени
- CALL = Вызов блока

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.9



Information and Training Center
Knowledge for Automation

- Overview** These actions are all defined in the IEC 1131-3 standard under 2.6.4.
- D, L, N** The associated actions are reset as soon as the step is completed.
- D, L** The times can be specified as a constant or as a variable.
- S, R** The corresponding actions remain active even after execution of the step.
- CALL** Block call FBi.DBi, FCi, SFBi.DBi, SFCi: calls the specified block. After the processing of the block, the GRAPH program is further executed.
- Note** The addresses in the action block can be either symbolic or absolute.
- Times** The times must be entered in the IEC time format. This is as follows:
T#mDnHoMpSqMS
mD: m Days
nH: n Hours
oM: o Minutes
pS: p Seconds
qMS q Milliseconds
The maximum time is limited to approximately 24 D20H.

Actions Dependent on an Interlock

Action block with conditional instructions

Action_block_2	
NC	M1.1
SC	M1.2
RC	M1.3
DC	M1.4
	T#1H2M3S
LC	M1.5
	T#4MS
CALLC	FB5.DB3

Conditions

- An action identified with a "C" (Condition) is only executed if the interlock condition for the step is true ("C" = 1).
- An interlock error exists if the condition is zero. The action subject to the condition C is then not executed. The step is marked and the error message "Error" is issued.

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.10Information and Training Center
Knowledge for Automation

Special Case

An action in a step which has the qualifier "C" but for which no interlock is programmed is executed unconditionally. The qualifier "C" is ignored.

Examples:

Instruction	Address	Explanation
NC	Q 1.0	As long as the step is active and the condition is fulfilled, the signal at Q 1.0 = 1, otherwise 0.
SC	Q 1.0	As long as the step is active and the condition is fulfilled, Q 1.0 is = 1, and remains set to 1.
RC	Q 1.0	As long as the step is active and the condition is fulfilled, Q 1.0 is = 0 and remains set to 0.
DC	Q 1.0 T#<const>	After the end of the time specified in <const> and as long as the step is active and the condition is fulfilled the signal at Q 1.0 is = 1. If the step is not active the signal is = 0.
LC	Q 1.0 T#<const>	As long as the step is active and the condition is fulfilled, the signal at Q 1.0 is = 1 for the specified time <const>. If the step is not active, the signal is = 0.
CALLC	FB5.DB3	Calls the specified block, when the condition is fulfilled. After the processing of the block, the GRAPH program is further executed.

Actions Triggered by an Event

Action block with event-driven instructions

Action_block_3		
A1	N	M1.1
L1	N	M1.2
L0	N	M1.3
S1	N	M1.4
S0	N	M2.4
V1	N	M2.5
V0	N	M2.6

The action is executed once in the cycle in which the event occurs

- A1 = Acknowledge
- L1 = Interlock error coming
- L0 = Interlock error going
- S1 = Step activated
- S0 = Step deactivated
- V1 = Supervision error coming
- V0 = Supervision error going

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.11



Information and Training Center
Knowledge for Automation

Overview

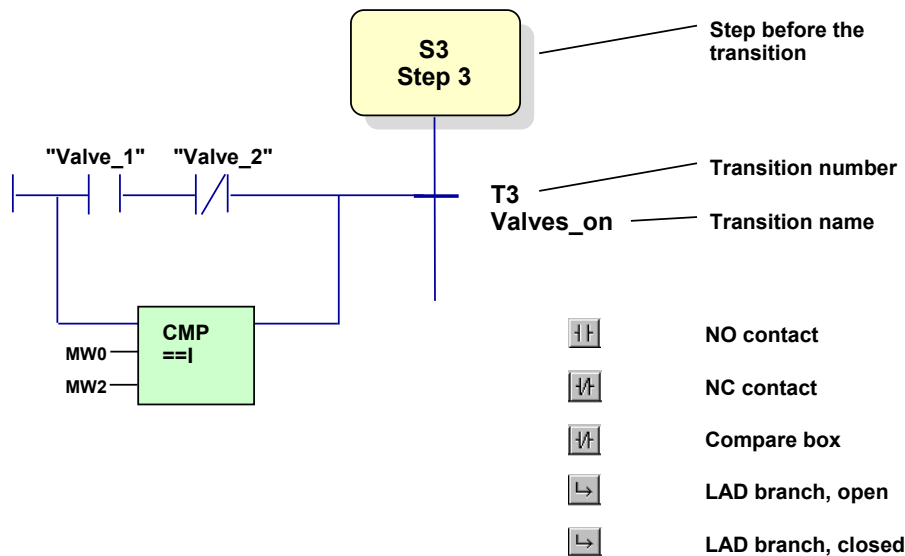
An event can be detected and linked with an action. That way it is possible not only to monitor individual steps but also to monitor and influence the entire sequential control system.

Activating and Deactivating Steps

Other steps of a sequencer can be activated or deactivated with the instructions ON and OFF.

Event	Instruction	Address	Explanation
S1 S0 V1 V0	ON OFF	Si i=Step number	Dependent on the event activate or deactivate step
L1 L0 A1	OFF	S_ALL	Dependent on the event activate or deactivate all steps; except for the step that contains the action

Checking Conditions in Transitions, Interlocks and Supervisions



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.12Information and Training Center
Knowledge for Automation

Overview

You program a condition as a boolean logic operation in the form of Ladder Diagram elements. For every transition, interlock and supervision there is an LAD network in which you can enter LAD elements.

Conditions

S7-GRAPH recognizes the following types of conditions:

Transition: describes the step-enabling conditions by which a controller passes from one step to the following step.

Permanent Condition: Permanent conditions are arranged before or after the sequencer and are evaluated once per cycle.

Interlock: describes conditions that must be fulfilled to allow an action to be executed. An interlock error is signalled if the step interlock is missing.

Supervision: describes conditions that lead to a supervision error being signalled and that prevent the sequencer from switching to the next step.

LAD Elements

The following elements are available for checking conditions: NO contact, NC contact, compare box, LAD branch open and closed.

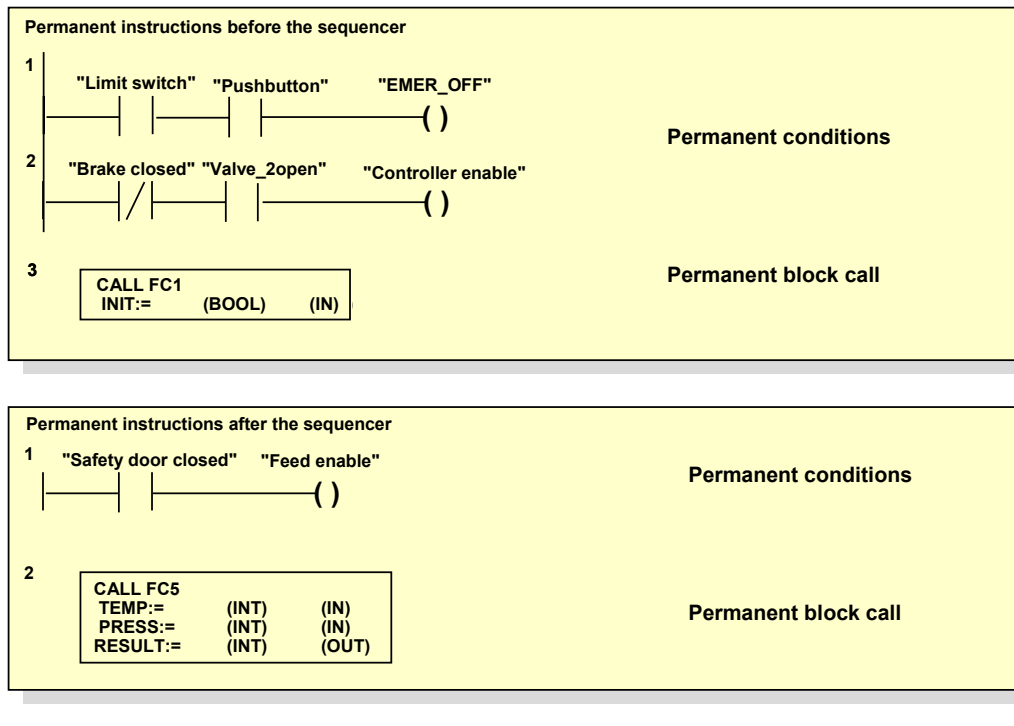
Editing

It is easiest to program the conditions in single step view. You can choose how you want to use the icons in the toolbar for editing a sequencer by selecting the menu options *Insert -> Drag&Drop/Append* or by clicking the relevant icon in the toolbar:

Drag&Drop or Insert Mode ON: First select the icon in the toolbar and then click with the mouse in the place where you want to insert the object.

Append or Insert Mode OFF: First select the element in the network after which you want to insert a new element and then click the desired icon in the toolbar.

Permanent Instructions



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.13Information and Training Center
Knowledge for Automation

Overview

Permanent instructions are conditions or block calls located before or after the sequencer and executed once per cycle.

You can use the permanent instructions, for example, to:

- Program only once conditions that need to be fulfilled at several points in the sequencer.
- Call blocks containing STL, LAD, FBD or SCL instructions from S7-GRAPH.

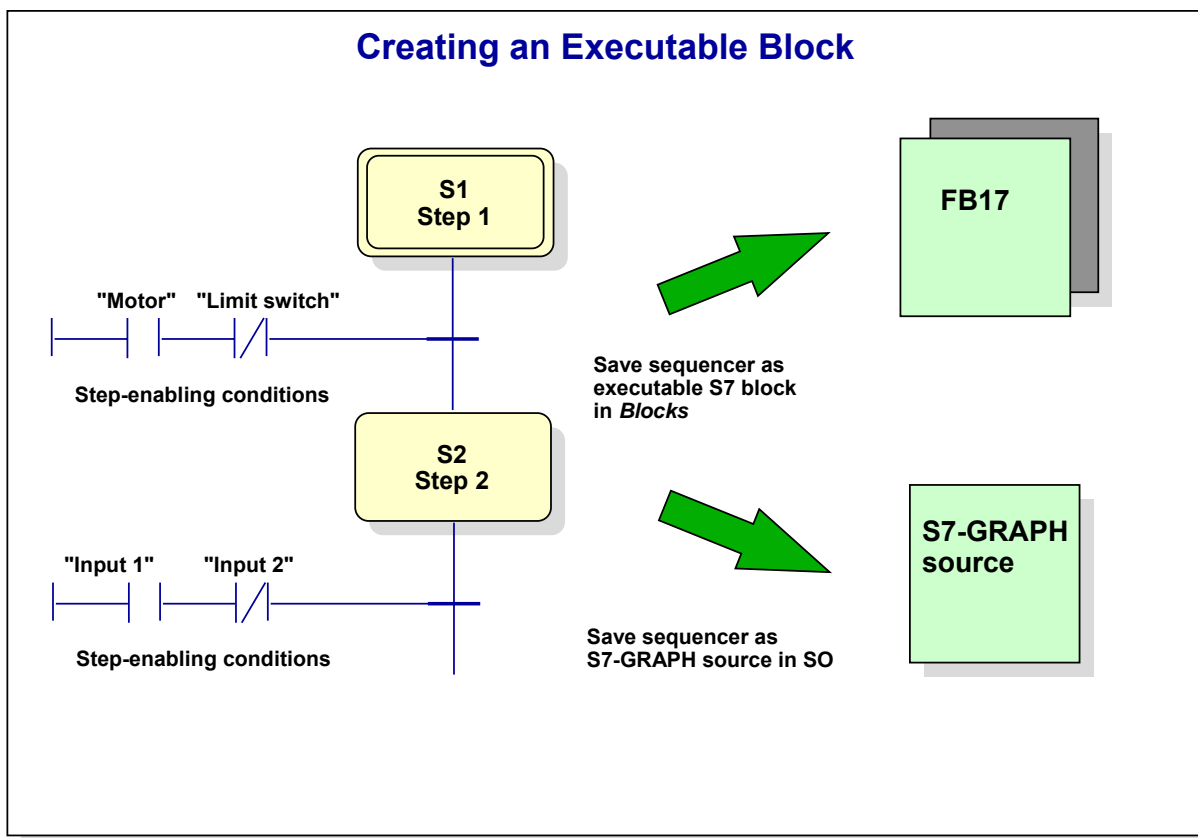
You can have as many permanent instructions in a GRAPH FB as you like.

Permanent Instructions

You program permanent conditions or block calls in *Permanent Instructions* view. Select the Drag&Drop insertion mode with the menu options *Insert -> Drag&Drop* and proceed as follows:

1. Select the menu options *View -> Permanent Instructions*, if the necessary window is not yet displayed.
2. Choose the menu options *Insert -> Permanent Instruction -> Condition/Call* or click with the mouse on the corresponding icon in the toolbar.
The mouse pointer changes its shape to represent an LAD network or a CALL instruction.
3. Place the mouse pointer in the space provided.
4. Choose the menu option or click the icon in the toolbar again.
5. Insert the LAD elements you want to use to program the condition or enter the name of the block you want to call after the CALL instruction and press the Enter key.
6. In the case of a block call, assign actual parameters to the formal parameters displayed.

Creating an Executable Block



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.14Information and Training Center
Knowledge for Automation

Overview

When saving an S7-GRAPH function block, a compilation function is triggered. That is, a function block that can be downloaded to the PLC is created from the sequencer. Only error-free user programs can be saved. If the program cannot be compiled because of errors then it can also not be saved.

If you have to interrupt your work although it still contains several configuration errors, you can at any time save the current state in an S7-GRAPH source by using the menu command *File -> Generate Source*.

Options

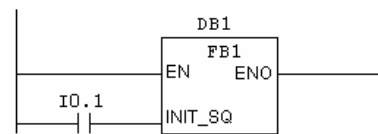
You can set the following options for compiling by choosing the menu items *Options -> Customize* and then the Compile tab:

- Which parameters are to be available when you call S7-GRAPH FBs (minimum, standard or maximum parameter set)
- How steps and transitions are to be stored in the instance DB (as structure arrays or as individual structures) and whether this interface description is also to be downloaded to the PLC or not.
- Whether the generated FB is to be executable on its own or whether a standard FC (FC 70 or FC 71) containing the major part of the management code is to be used. If you choose the standard FC option, you can store more steps in the FB than if it is to be executable on its own
- Whether condition analysis data is to be written in the instance DB, whether Skip mode is activated and whether supervision errors that occur during operation have to be acknowledged.
- Whether program and process are to be synchronized and whether all interlock conditions are always to be processed in manual operation. When displaying the status, a missing interlock and the step which is faulted as a consequence are then displayed.
- Whether warnings are to be displayed in the message window during compiling.
- Whether interlock and supervision errors are to be handled by SFC 52 (diagnostic buffer entry) or using SFC 17 and 18 (send to HMI devices).

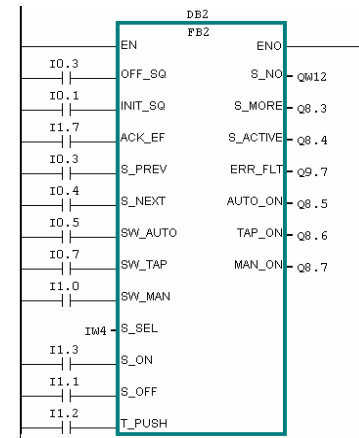
Integrating an FB Call in OB1

Two forms of block call

- **Minimum call parameters (default)**
 - 1 input parameter for controlling the sequencer
- **Standard call parameters**
 - 12 input parameters for controlling the sequencer
 - 7 output parameters for displaying operating states
- **Maximum parameter set**
 - 17 input parameters for controlling the sequencer
 - 12 output parameters for display



Minimum parameter set



Standard parameter set

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.15



Information and Training Center
Knowledge for Automation

Sequencer FB

By selecting the menu options *Options -> Customize* and the Compile tab page you can also choose whether the FB is to be assigned the minimum, standard or maximum set of parameters when it is called.

The FB responds to rising edges at the input parameters.

Minimum Parameters The FB with the minimum parameter set has only one input parameter, INIT_SQ, and activates its sequencers as soon as it has been processed in OB1. This means that the sequencers are executed immediately in Automatic mode.

You use the minimum parameter set when you are running the sequencer in Automatic mode only and when you do not need any additional modifying and monitoring functions.

The initial step(s) is/are activated by a rising edge at the parameter INIT_SQ.

Standard Parameters The operating mode must also be selected when calling this FB. You always use the standard parameter set when the sequencer is to run in different operating modes and when you require feedback information about the process and facilities for acknowledging messages.

The sequencer FB always remains in the operating mode last activated. The previous operating mode can only be deselected by selecting a different one. Parameters that are not required remain unassigned.

Maximum Parameters You always use the maximum parameter set when you need more operator intervention and monitoring facilities for service and startup than those provided by the standard parameter set.

All the FB parameters are displayed and can be assigned in the FB call (requires the most memory capacity).

Activating the Debugging Functions

Procedure

- **Download sequencer FB and instance DB**
 - Use the menu commands **PLC -> Download** to download the sequencer FB and the instance DB to the PLC
- **Select instance DB:**
 - Select the instance DB you want to use for the test by choosing the menu commands
Debug -> Test Environment
- **Start the "Monitor" function:**
 - Select the desired section of the sequential control system. The status information will be given for the part currently visible in the open window.
 - Activate the menu command **Debug -> Monitor** (checkmark)
- **Exit the "Monitor" function:**
 - Deactivate the menu command **Debug -> Monitor**

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.16



Information and Training Center
Knowledge for Automation

Debug Functions

You can execute the S7-GRAPH sequencer in test mode with the Debug functions. The status of the sequencer elements and the signal state of the addresses are displayed on the screen.

Downloading to PLC

In order to download the S7-GRAPH FB with the associated instance DB into the PLC, proceed as follows:

1. With the FB opened, select the options **PLC -> Download**.
2. In the "Download" dialog box select the instance DB which you wish to download into the CPU together with the opened FB,.
3. If necessary, confirm the query as to whether you want to overwrite blocks of the same name that are already in the CPU.

Note

If possible, download the blocks in STOP mode, because error conditions can occur if you overwrite an old program in RUN mode.

Activate Monitor

This function is performed simultaneously for all opened windows of the same sequencer FB. If too much status information from the S7 has to be updated, a warning message may appear that says that as of now not all statuses can be updated time-synchronously.

By acknowledging the message, you can return to the Monitor mode of the sequencer.

Deactivate Monitor

Sequencer monitoring must always be interrupted first before you can make changes in the sequencer or in the logic operations. Only then are actions in the sequencer possible.

It is advisable to save your changes first on the hard disk and then download the block into the S7 CPU. If you do this in the wrong order, a message window appears, that points out the correct sequence.

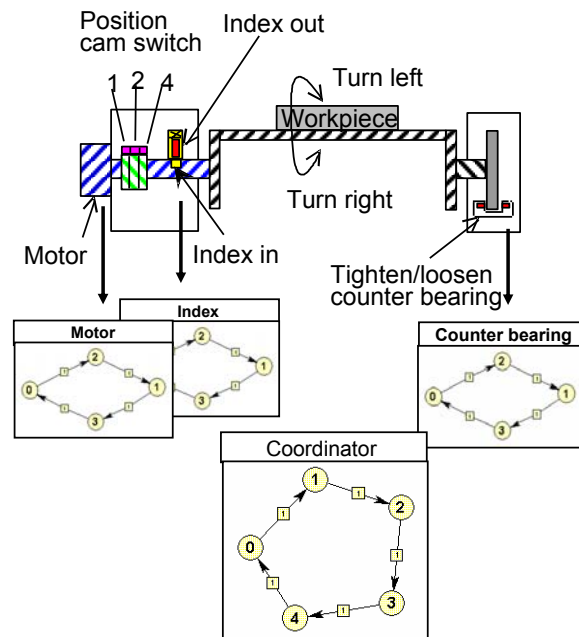
The S7- HiGraph Software Package

S7-HiGraph: Tool for programming with state diagrams

- Machine is divided up into function units
- State diagrams are created for each function unit
- States contain actions
- State diagrams communicate by means of messages

You can optimize the following phases in an automation project with S7-HiGraph:

- Planning, configuring
- Programming and debugging
- Startup
- Maintenance, diagnostics
- Supports reusability



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.17



Information and Training Center
Knowledge for Automation

Motivation for State Diagrams

The construction of machinery and plants is an extensive field in which the control of asynchronous, mechanical movements and responses over time presents the main problem.

The language most commonly used in this field at present is STL and sometimes LAD, FBD and sequencer languages (GRAPH5, S7-GRAPH, etc.). However, these languages are not particularly suitable for mechanical engineers, for example for formulating the application of the mechanical construction.

The result is that each group, for example the mechanical engineers or the automation engineers, uses its own methods (languages), which makes it difficult for them to exchange information.

The aim of a working group (approx. 1980) with the task of investigating CASE tools for PLCs was to specify a tool that could be used in all phases of a project right through from the definition of the problem to programming and even servicing. This tool was to be suitable for use in all areas and allow an increasingly detailed approach to the automation problem. It was also a requirement that the documentation and programs should be re-usable for other projects of similar type.

The result was the state diagram method described here, which is marketed by SIEMENS under the product name S7-HiGraph. S7-HiGraph can be used on the PLCs of the S7-300 series (CPU 314 and higher) and S7-400.

Advantages

The object-oriented approach of S7-HiGraph is ideally suited for:

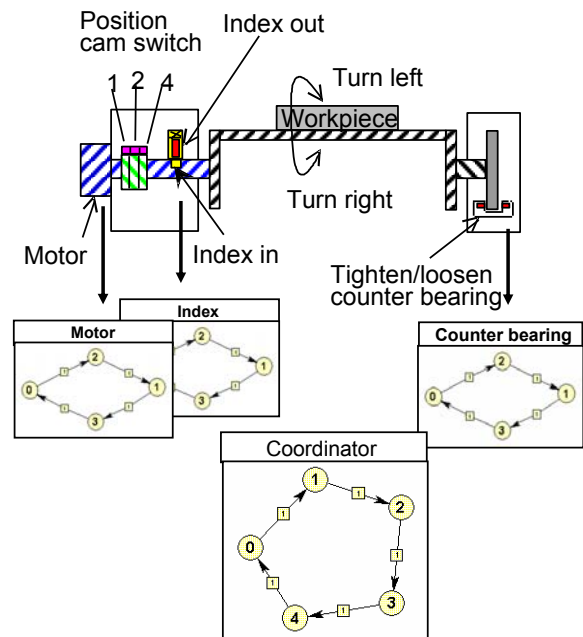
- Machine and plant engineers (mechanical design)
- Automation specialists (electrical engineers) as a common means of description
- Commissioning and maintenance engineers

The state diagram method enables the entire process of constructing a machine or plant to be optimized by reducing the development and turnaround times and the commissioning time.

Principle of the State Diagram Method

Example: Rotary table for milling machine

- **Function units (FUs)**
 - Motor
 - Index
 - Counter bearing
- **State diagrams**
 - One diagram for each FU
 - One coordinating diagram



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.18



Information and Training Center
Knowledge for Automation

Overview

The state diagram method orients itself towards the "objects" in the real world when used for drafting out and programming automation projects. The machine or plant to be automated is seen as a combination of separate elements, or function units.

Function Units

The function unit (FU) is the smallest mechanical unit of a machine or plant. An FU is normally composed of basic mechanical and electrical elements.

During programming, each function unit is assigned a state diagram (or graph) in which the functional, that is the mechanical and electrical, properties of the FU are represented.

For the state diagram method the object to be automated is broken down into separate function units.

State Diagram

The state diagram describes the dynamic behavior of a function unit. It describes the states a function unit can assume and the transitions between them.

State diagrams are program sections that can be used again and again. State diagrams created for a particular function unit can be used in other places in a program where a similar functionality is required.

Graph Group and Instances

The entire functionality of a machine or plant can be described by a combination of parallel state diagrams.

All the state diagrams drawn up within an S7 program are stored centrally in the "Sources" folder. From there you can insert and call them as often as you like in one or more graph groups.

A state diagram call in a graph group is referred to as an instance.

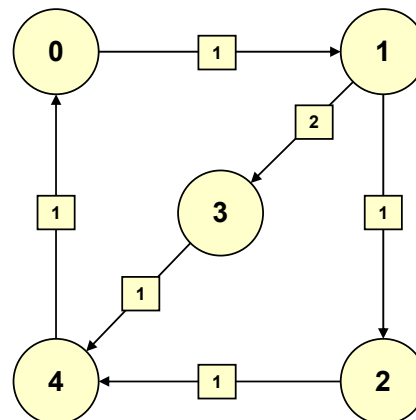
Elements of a State Diagram

States 0,1, ...

- Represented by circles
- Static states
- Dynamic states
- Always one active state
- Actions are assigned to the states

Transitions

- Represented by arrows
- Transition conditions and actions are assigned to the transitions



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.19



Information and Training Center
Knowledge for Automation

States

A state diagram is a continuous, directed graph, that represents all states of a function unit as circles and all transitions as arrows.

The states of a function unit can be of the static type (Door_is_open, Motor_off, etc.) or the dynamic type, that is, movement states (Door_opens, Motor_turns_left, etc.).

At any given time, the subsystem which is described by a state diagram, or graph, is in exactly one state.

Actions

Actions can be assigned to states in state diagrams. These actions can be subdivided into:

- Actions that are executed once when entering a new state.
- Actions that are executed cyclically, as long as the function unit is in the respective state.
- Actions that are executed once when leaving a state.

The actions are formulated in a language similar to STL.

Transitions

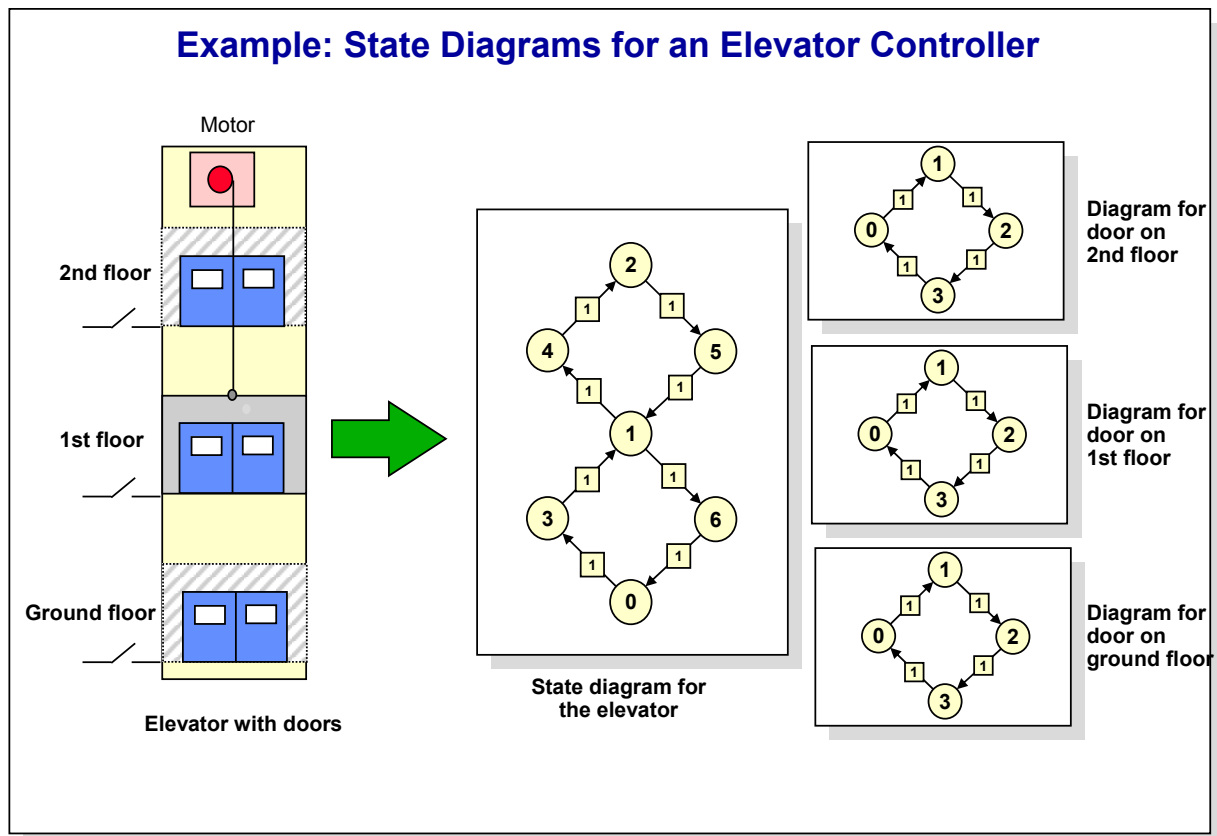
The transitions identify the state changes of a function unit. The transitions between the states are dependent on conditions that are true or not true at any given time.

The subsystem described by a state diagram changes its state when the condition of a transition that leads out of a state is fulfilled. A maximum of one transition occurs per cycle for each state diagram.

The transition conditions are also formulated in a language similar to STL..

For each transition one action can be formulated which is to be performed when the transition takes place.

Example: State Diagrams for an Elevator Controller



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.20Information and Training Center
Knowledge for Automation

Overview

The above example demonstrates the state diagram method by means of an elevator in a three story building.

Breaking down into FUs

The object to be automated (elevator with doors) can be broken down into the following mechanical function units with respective state diagrams:

- Elevator cage including control
- A door on each floor

State Diagram for Elevator Cage

Within a state diagram, or graph, the circles represent the possible states of the function unit and the arrows represent the state transitions.

For the function unit "elevator cage", the states 0, 1 and 2 represent the stopping positions of the elevator cage on the respective floors.

The states 3, 4 and 5 and 6, 7 and 8 represent the dynamic up and down movement of the elevator cage between the floors.

State Diagrams for Door

In the function unit "door" the states 0 and 1 represent the static states "Door is closed" and "Door is open", the states 2 and 3 represent "Door is opening" and "Door is closing".

Messages

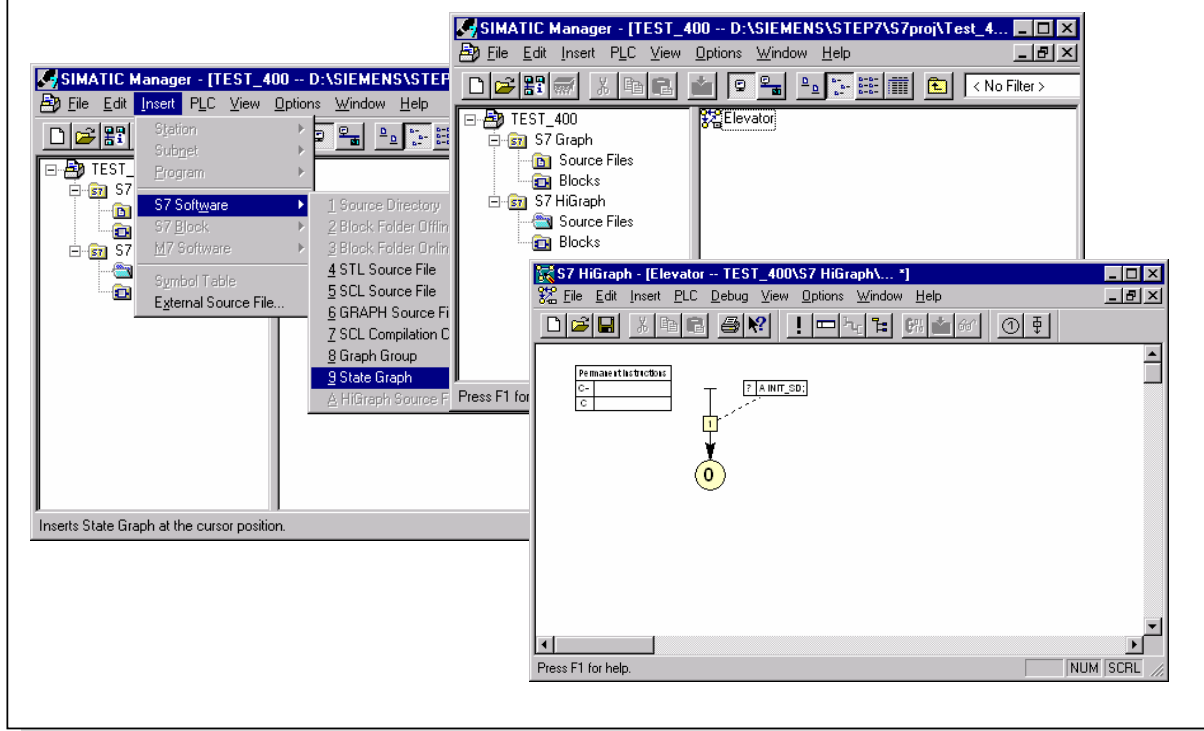
The coordination of the state diagram for the elevator cage with the individual state diagrams of the doors can take place without an additional coordination diagram.

On reaching the "desired floor" the state diagram "Elevator Cage" sends the message "Open_Door" to the associated state diagram "Door".

This state diagram receives the message and "opens" the door, that is, on receipt of the message the corresponding transition is executed. When the door is closed again, the state diagram "Door" sends the message "Door_closed" to the state diagram "Elevator Cage".

The elevator cage can then move to the next desired floor.

Creating a State Diagram



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.21



Information and Training Center
Knowledge for Automation

Overview

In order to edit state diagrams, S7-HiGraph requires an existing project. You may have to create this first with the SIMATIC Manager before you invoke the HiGraph Editor.

Inserting a State Diagram

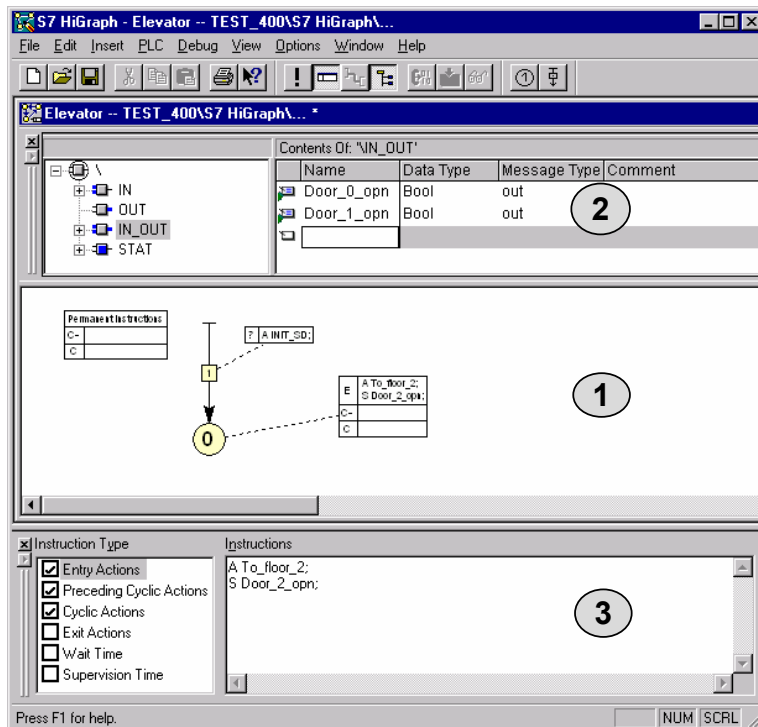
To insert a state diagram with the SIMATIC Manager, proceed as follows:

1. Open the source folder.
2. Select the menu options *Insert -> S7 Software -> State Graph*.
A state diagram is created under a default name in the *Sources* folder. Before you start editing, you should change the name of the state diagram (e.g. Elevator).
3. Double-click the state diagram. The HiGraph Editor is started and the state diagram is opened.

Initial State

The state diagram you have opened already contains a state with the number 0 and an ANY transition. The state with the number 0 is the initial state of the state diagram. This is automatically assumed when the power is switched on.

The HiGraph User Interface



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.22



Information and Training Center
Knowledge for Automation

User Interface

The HiGraph user interface consists of various windows, which you can display or not, as required. To make optimum use of the screen space available, you can change the size of the windows and move them around.

In addition to the editing windows (1) in which you edit the state diagrams (graphs) and graph groups, you can also use the following windows:

- The variable declaration window (2) is used for declaring the variables of the state diagram (graph). You display this window by selecting the menu options *View -> Variables*.
- You use the instruction input window (3) to program the contents of states, transitions and permanent instructions. You display this window by selecting the menu options *View -> Instructions*.
- There is another window in which errors and warnings arising during compilation of a graph group are displayed. This window is automatically opened after each compilation. You can also display it by selecting the menu options *View -> Messages*.
- The input window for actual parameters is only available if a graph group is opened. You use this window for assigning the actual parameters of instances. You can also open this window by selecting the menu options *View -> Actual Values*.

Declaration of Variables

In the variable declaration you declare the local variables and parameters of the state diagram. You also declare the variables to be used for exchanging messages.

The variable declaration consists of the following sections:

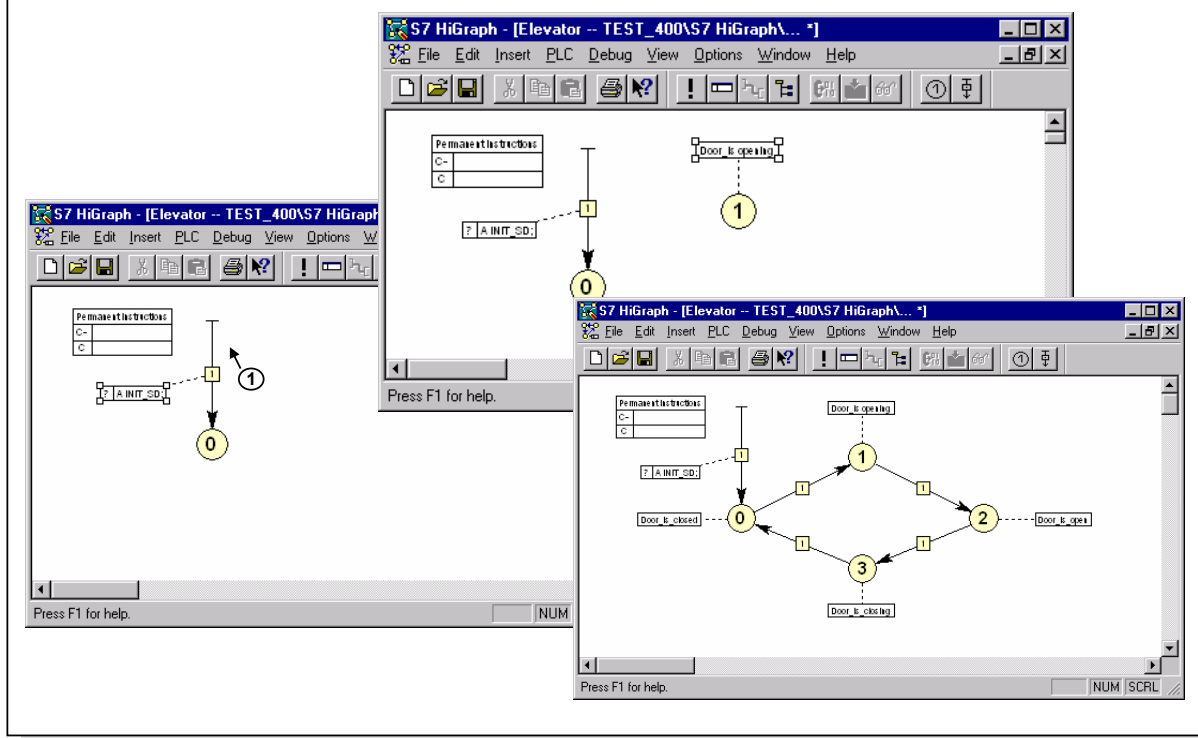
IN: contains the input parameters of the state diagram and the pre-defined variables "AutomaticMode" and "ManualMode".

OUT: contains the output parameters of the state diagram

IN_OUT: contains the input/output parameters of the state diagram and the variables used for exchanging messages.

STAT: contains static variables and variables pre-defined by HiGraph

Inserting States and Transitions



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.23



Information and Training Center
Knowledge for Automation

Inserting States

To insert new states into a state diagram (graph), proceed as follows:

1. With an editing window selected, choose the menu options *Insert -> State*. The cursor assumes the shape of a circle with the next free state number.
2. Position the cursor to the desired position and click with the left mouse button.

At the selected position, a circle is automatically inserted for a state with a default number 0, 1, 2, etc. Repeat this procedure until you have inserted all the necessary states for the description of the state graph.

3. Exit the input mode by clicking the empty space in the editing window with the left mouse button.

Each state has a number which is unique within the state diagram. To make the diagram clearer, you can give each state a name by selecting the menu options *Edit -> Object Properties*.

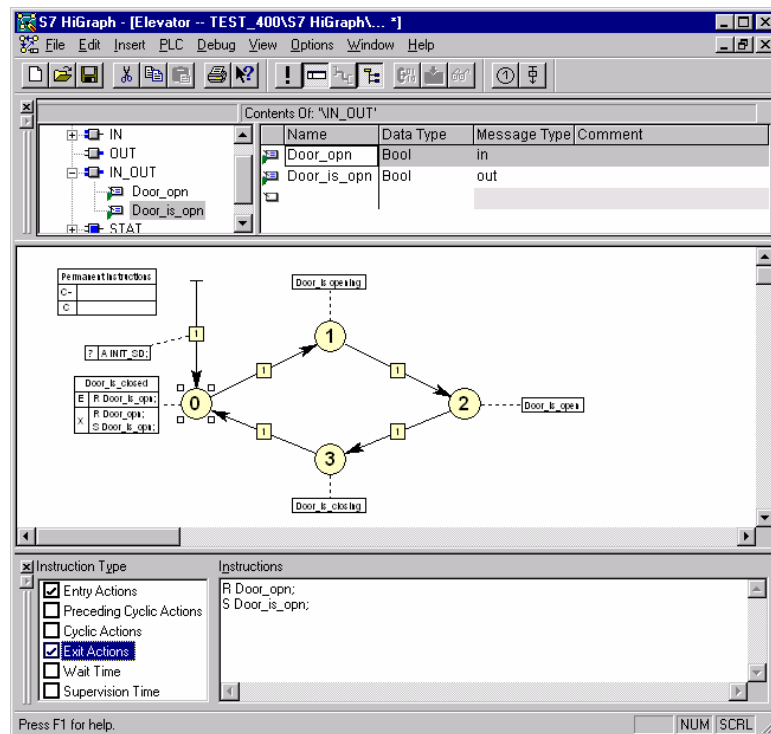
Inserting Transitions

To insert transitions between individual states, proceed as follows:

1. Select the menu item *Insert -> Transition*. The cursor assumes the shape of the transition symbol.
2. Then click with the left mouse button on the initial state and, keeping the left mouse button pressed, drag the arrow to the target state.
A transition is inserted between initial and target state.
3. Exit the input mode by clicking the left mouse button.

As with the states, you can give each transition a name by selecting the menu options *Edit -> Object Properties*.

Programming Actions



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.24



Information and Training Center
Knowledge for Automation

Action Types

You can program actions for each state in the instruction window. The actions are subdivided into the following types:

Entry actions (E): Actions which are performed only once on entering the state.

Preceding cyclic actions (C-): Actions which are executed whilst in the state before the transition conditions are checked.

Cyclic actions (C): Actions which are executed whilst in the state after the transition conditions have been checked.

Exit actions (X): Actions which are only executed once on leaving the state.

To enter instructions, select the type of action in the left-hand pane of the instruction window and then enter the instructions in STL in the right-hand pane. The actions are displayed in the Editor window.

Notes

RLO is always =1 at the beginning of execution of an instruction table.

To make state diagrams re-usable, only variables which have been declared in the declaration window may be used. Actual parameters can then be assigned to these variable when inserting the state diagram into a graph group.

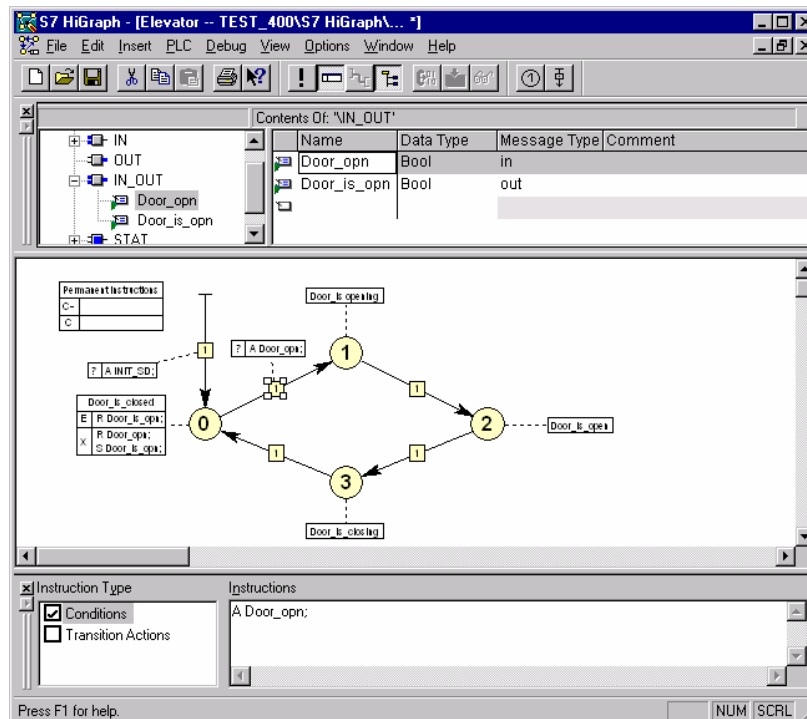
Wait Time

You can specify whether the PLC is to remain in a state for a minimum length of time before checking for step-enabling transition conditions. You can specify either constants or variable values for the wait time. You must then assign the attribute "Waiting" to the transitions that are to observe the wait time.

Supervision Time

The supervision time is used for monitoring the time spent in a certain state. If the state is not exited within the specified time, the predefined variable "ST_Expired" is set and an error message is entered in the diagnostic buffer.

Programming Transitions



SIMATIC S7

Siemens AG 1999. All rights reserved.

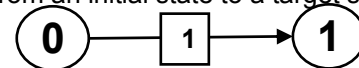
Date: 04.11.2005
File: PRO2_13D.25Information and Training Center
Knowledge for Automation

Transitions

A transition contains the conditions for switching from one state to another. Several outgoing transitions can be assigned to one state. HiGraph operates with the following transitions:

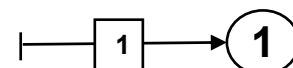
Normal transition: A normal transition leads from an initial state to a target state.

It is represented by the following symbol:



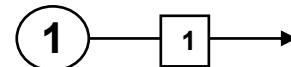
Any transition: An Any transition leads from all states to a target state. It has higher priority than all other transitions.

It is represented by the following symbol:



Return transition: A Return transition leads back from the current state to the state which was previously active.

It is represented by the following symbol:



Priority

Transitions leading out of the same state can be arranged in the right order by assigning them different priorities. If the transition conditions are fulfilled at the same time, the transition with the highest priority is activated.

The highest priority possible in S7-HiGraph has the numerical value 1.

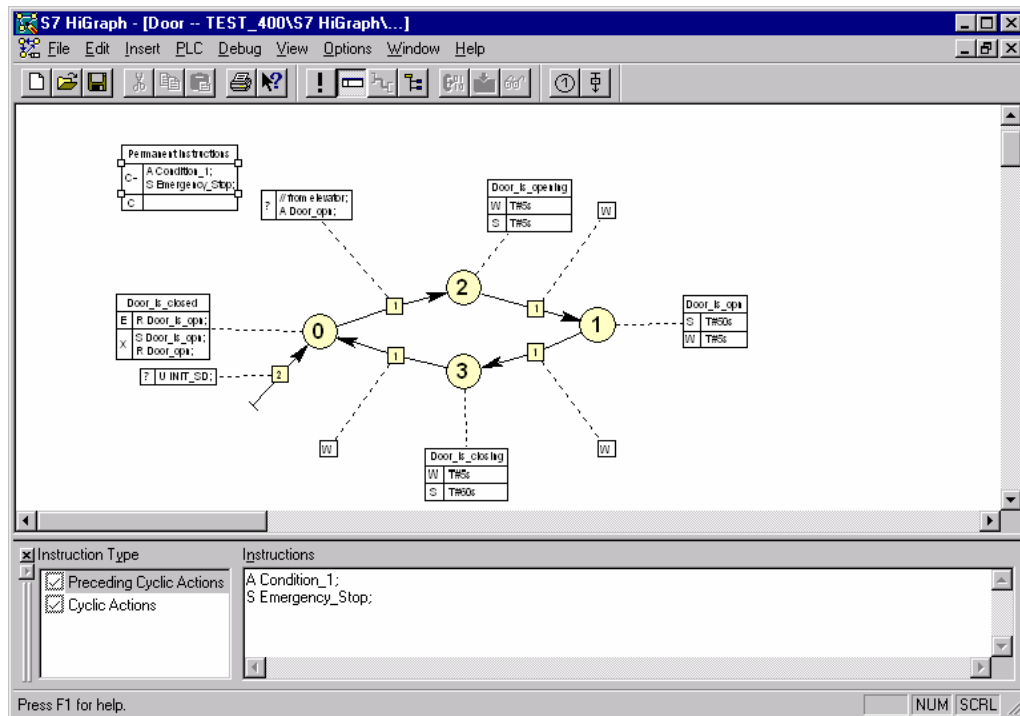
Instructions

You can program conditions and actions for a transition in the instruction window:

Conditions (?): These instructions describe the conditions that must be fulfilled before a change of state can take place.

Actions (!): These instructions are executed once when the transition is activated.

Programming Permanent Instructions



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.26



Information and Training Center
Knowledge for Automation

Permanent Instructions

Permanent instructions are executed once in every cycle regardless of the current state. For example, you can program the following activities centrally in permanent instructions:

- Calculation of process variables that are checked in several places.
- Detection and processing of events which require a response not dependent on the current state (example: monitoring of a safety screen).

The following types of permanent instruction are available:

Preceding cyclic actions (C-): These are always performed before the actual state diagram is executed.

Subsequent cyclic actions (C): These are always performed after the actual state diagram has been executed.

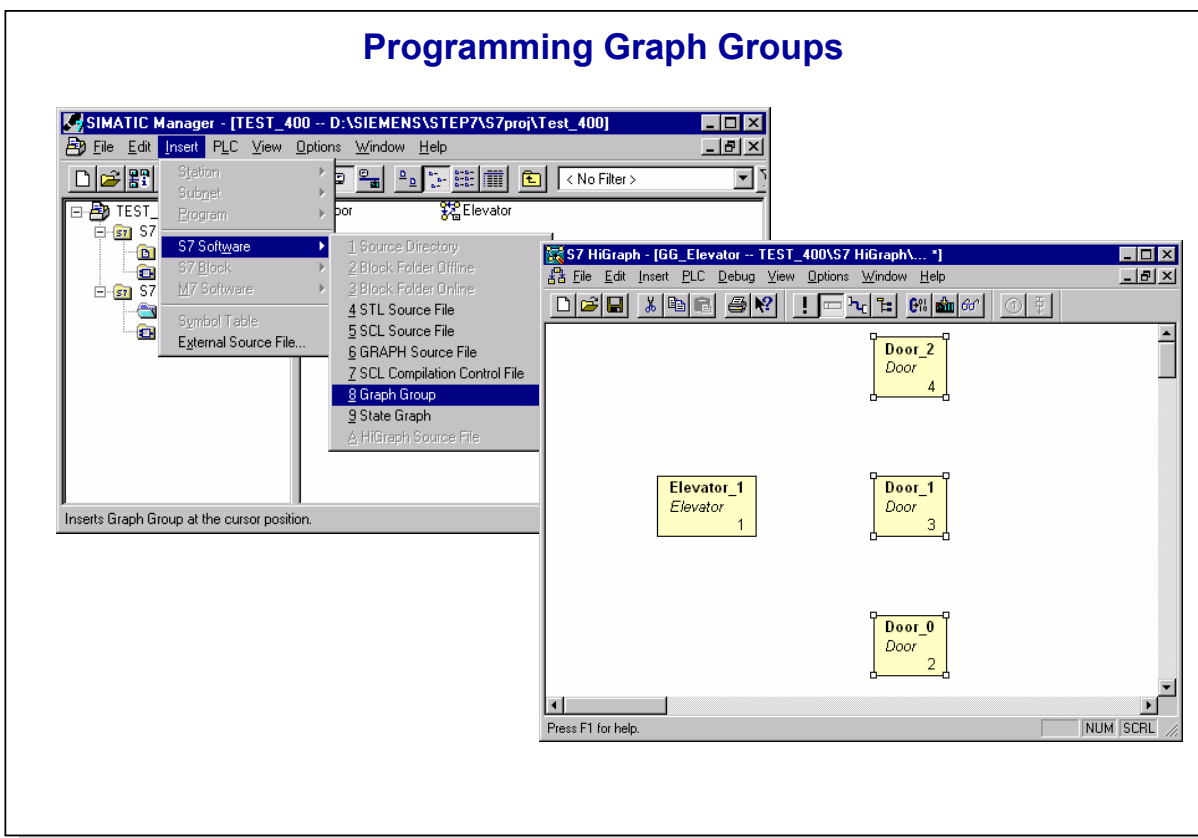
Programming

You program permanent instructions as follows:

1. Double-click the instruction table with the title "Permanent Instructions". The instruction input window is opened.
2. Select a type of instruction in the left-hand pane of the window and enter the instruction in STL in the right-hand pane.

When you have entered the instructions, they are displayed in the form of a table in the editing window for the state diagram.

Programming Graph Groups



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.27



Information and Training Center
Knowledge for Automation

Overview

State diagrams represent individual function units of a machine. In order to describe a whole machine or plant, the state diagrams (graphs) for the separate function units must be collected together in a graph group.

Creating a Graph Group

To create a graph group with the SIMATIC Manager, proceed as follows:

1. Open the source folder in which you want to insert the graph group.
2. Select the menu options *Insert -> S7 Software -> Graph Group*.
A graph group is created under a default name in the *Sources* folder. You should change the name of the graph group before you start editing it (e.g. Elevator).
3. Double-click the graph group. The HiGraph Editor is activated and the graph is opened.

Instanting State Diagrams

The insertion (calling) of a state diagram (graph) in a graph group is known as instancing. When you have inserted the state diagram into a graph group, you must assign actual parameters to the formal parameters declared in it.

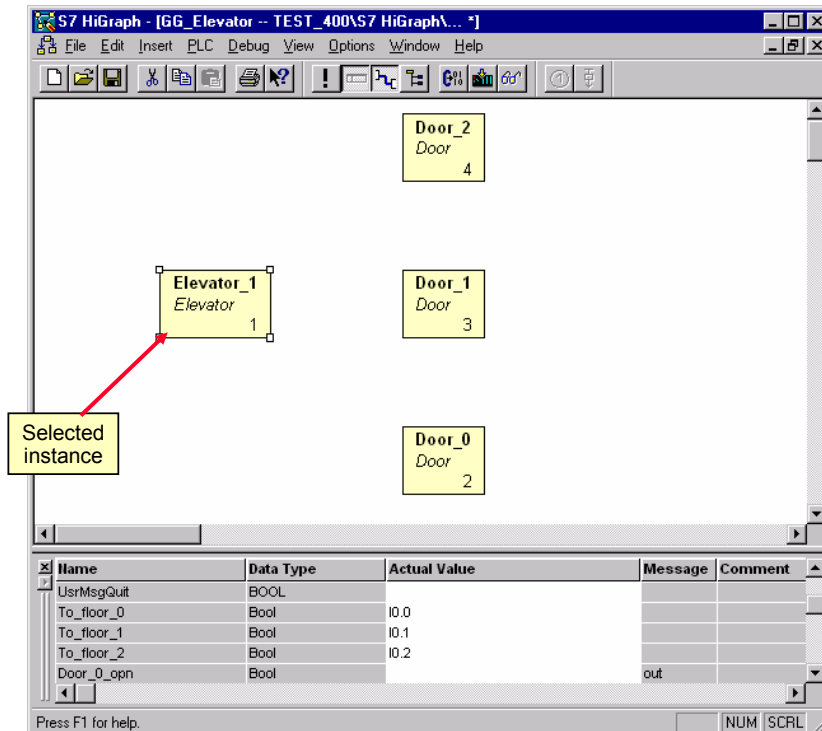
It is the assignment of these actual parameters that enables an instance of the state diagram to control a real function unit (e.g. motor, elevator door, valve, etc.). If several identical function units (e.g. motors of the same type, elevator doors, etc.) need to be controlled, you can do this using several instances of the same state diagram.

Inserting State Diagrams

To insert instances of state diagrams (graphs) into a graph group, select the menu options *Insert -> Instance*.

You can then give the inserted instance a name by selecting the menu options *Edit -> Object Properties*.

Assigning Actual Parameters



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.28Information and Training Center
Knowledge for Automation

Overview

To enable you to use state diagrams again and again by means of instancing, all the signals used in the state diagram must be declared as formal parameters in the declaration window.

The formal parameters save spaces for the actual parameters and are assigned the "actual signals" when an instance is created.

Actual Parameters

When you have inserted the instances of a state diagram into a graph group, you assign actual parameters to the formal parameters as follows:

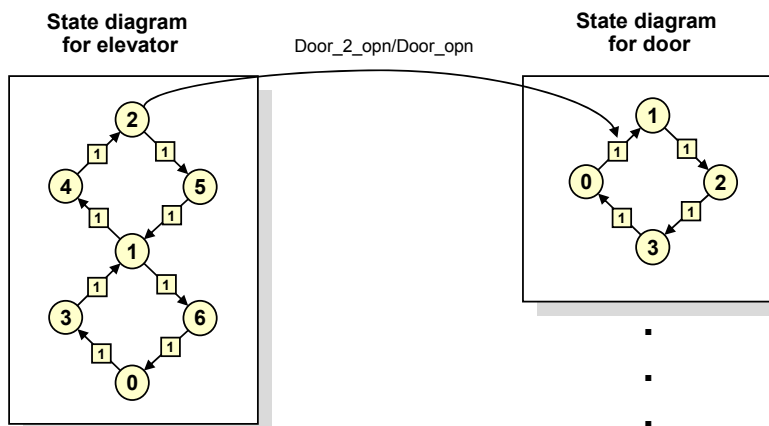
1. First select the required instance of the state diagram in the graph group window.
2. Select the menu options *View -> Actual Parameters* to open the actual parameters window. The names and data types of all the formal parameters in the state diagram appear in this window.
3. Assign addresses to the formal parameters in the "Actual Value" column.
As actual parameters for a HiGraph program you can use the addresses of I/O signals, bit memories, counters, timers, as well as data and code blocks. In your program you can use either absolute addresses (e.g. I 1.1, M 2.0, FB 21) or symbolic names (e.g. "Motor_ON").

You use the symbol table of your S7 program to assign symbolic names to the absolute addresses. You can toggle between absolute and symbolic addressing by selecting the menu options *View -> Symbolic Representation*.

Messages?

You will find out how to assign messages to "actual parameters" on the following pages.

Message Exchange between State Diagrams



A To_floor_2;
S Door_2_opn;

Entry action for state 2

Name	Data type	Message
Door_2_opn	bool	out

Interface declaration for elevator

A Door_opn

Transition condition for t_{01}

Name	Data type	Message
Door_opn	bool	in

Interface declaration for door

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.29



Information and Training Center
Knowledge for Automation

Overview

State diagrams can influence each other by exchanging messages. Messages are normally sent by a source diagram (graph) and then evaluated by the destination diagram.

Messages

A message is a binary variable, that can be sent by a graph within its action part and that can be received by the receiving graph within its action or transition parts.

A message is always sent to a specific state diagram (internal messages) or to an address (external message).

Internal Messages

Internal messages are used for synchronization within the same graph group. They are mapped by the system in the bits of the associated DB.

External Messages

External messages are used for synchronization of graphs in different graph groups (FCs). A global bit variable is declared when assigning the actual parameter.

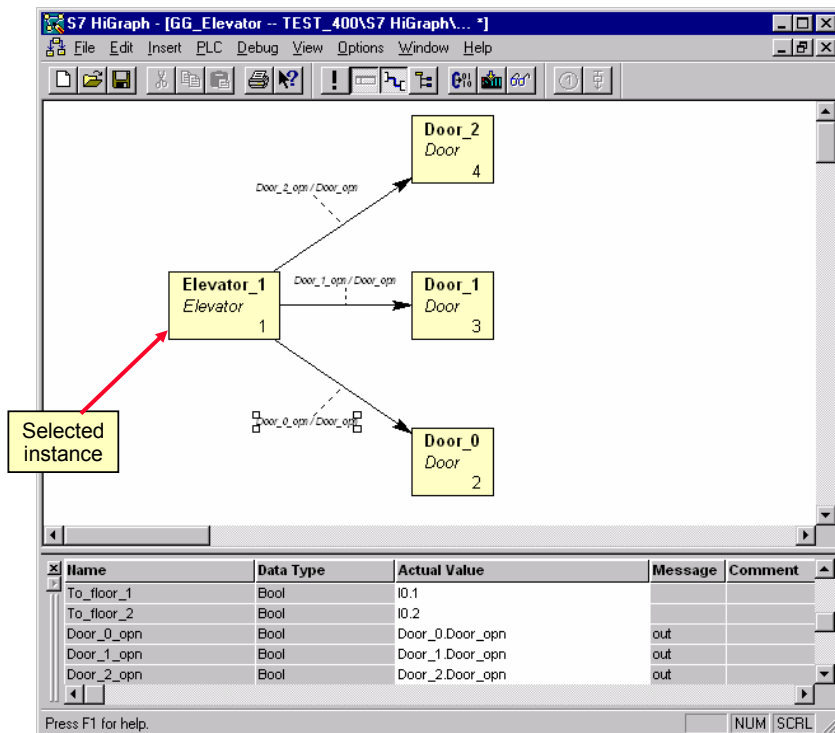
Declaration of Messages

You first declare messages in the IN_OUT section of the declaration window of a state diagram (graph). In addition to the name and the data type (always BOOL) of the message you must also specify the message type, that is input or output message.

Name	Data Type	Message	Comment
Door_0_opn	bool	out;	//Output message
Door_closed	bool	in	//Input message

You assign the actual recipient of the message (internal message) or the bit variable with which the message is connected (external message) in the actual parameter window of the relevant graph group.

Assigning Actual Values for Messages



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.30



Information and Training Center
Knowledge for Automation

Overview

When you have inserted the instances of a sending and receiving state diagram in a graph group, you must tell the sending instance which instance is to receive the message (internal message) or which bit variable the message is connected to (external message).

Assignment for Internal Messages

To assign a recipient to an internal message, proceed as follows:

1. Select the instance of the sending state diagram in the graph group window.
2. In the actual parameter input window, select the message to be sent and enter the receiving instance in the "Actual Value" column.

The full name of the receiving instance consists of the name of the instance that is to receive the message and - separated by a period - the name of the message (type: in), declared as an input message in the receiving graph.

Name	Data Type	Actual Value	Message
Door_0_opn	BOOL	Door_0.Door_opn	out
Door_1_opn	BOOL	Door_1.Door_opn	out

For internal input messages you do not need to assign an actual value in the actual parameter window of the receiving instance.

Assignment for External Messages

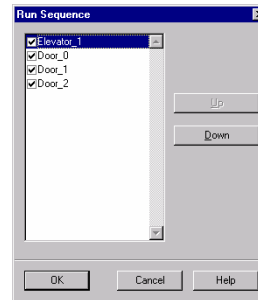
To link an external message to a bit variable proceed as follows:

1. Select the instance of the sending state diagram (graph).
2. In the actual parameter input window, select the message to be sent and enter a global bit address in the "Actual Value" column.
3. Now select the instance of the receiving state diagram (graph) and assign the same global bit address to the relevant input message.

Saving and Compiling

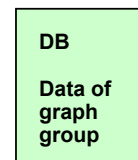
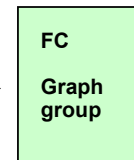
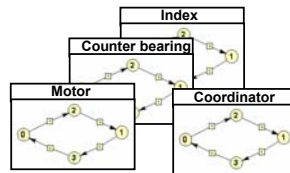
Establish execution sequence

- Menu:
Edit -> Execute Order



Compile

- Menu:
File -> Compile



Integrate in OB1

- Assign parameter
INIT_SD

```
OB1 : Title:
Network 1: Title:
CALL "Elevator"
INIT_SD:=IO.7
```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.31



Information and Training Center
Knowledge for Automation

Saving

When you save HiGraph objects, they are stored in the "Sources" folder of the S7 program in their current state. The syntax is not checked. This means that it is possible to save and later use objects containing errors.

You save HiGraph objects by selecting the menu options *File -> Save*. Please note that any changes you make to a state diagram affect all the instances of it that have already been inserted in graph groups.

Code Execution Sequence

The state diagram system is executed cyclically. You can establish the order in which the individual instances within a graph group are to be executed by selecting the menu options *Edit -> Execute Order*.

Compiling

In HiGraph you only compile graph groups; you cannot compile state diagrams individually. When compiling, HiGraph checks the syntax of the program, generates a function (FC) and a data block (DB) and stores them in the "Blocks" folder of the relevant S7 program.

Any syntax errors detected during compiling are reported in the message window. In this case, no blocks are generated.

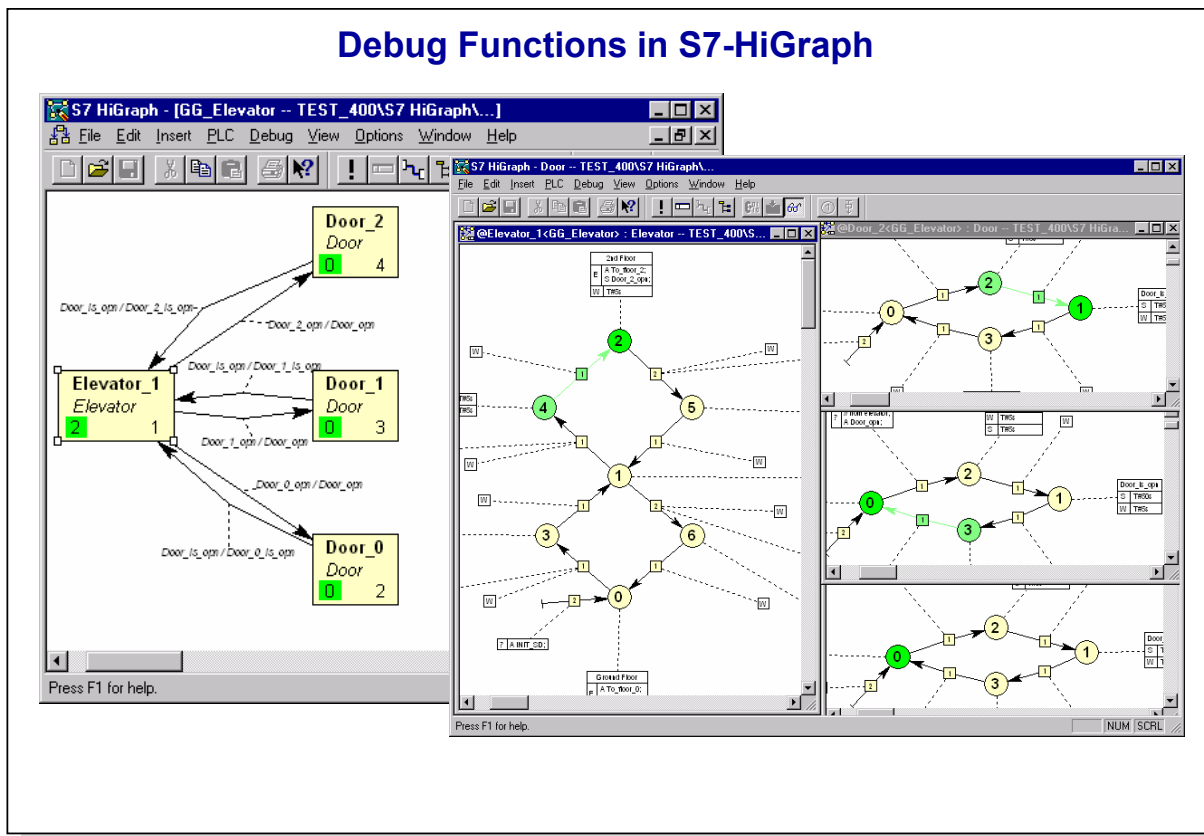
To compile a graph group, you take the following steps:

1. First select the menu options *Options > Customize* and enter the names for the FC and DB and make any other settings for compilation on the "Compile" tab page.
2. In the graph group window select the menu options *File -> Compile*.
3. Watch for any error messages in the message window. To jump to the position of the error, simply double-click the error message.
4. Compile the graph group again.

Calling the FC

To enable the HiGraph program to run in the CPU, the FC must be called in a block that is executed cyclically (e.g. OB1) and the initialization parameter **INIT_SD** must be assigned.

Debug Functions in S7-HiGraph



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.32



Information and Training Center
Knowledge for Automation

Overview

The monitoring functions enable you to monitor and check a program when it is executing in the CPU.

HiGraph provides the following debugging and monitoring functions:

- Monitoring the program status
- Monitoring and modifying variables (as in STL/LAD/FBD)
- Evaluation of reference data (as in STL/LAD/FBD)

Program Status

You can use the program status monitoring function to check the execution of all the instances in a graph group. The execution of the individual states and transitions is highlighted in color, and information about the instruction tables is also displayed.

The following program status monitoring facilities are available in the various HiGraph windows:

- Graph group window: Here you can see the status of all the instances in the graph group. The current state is shown in each instance.
- State diagram window: Here you can obtain detailed status information for a selected instance (current state, transition, etc.).

Procedure

To start monitoring the program status, proceed as follows:

1. With the graph group open, select the menu options *Debug -> Monitor*. The status overview for the graph group is displayed.
2. Select one or more instances and select the menu options *Edit -> Open Object* or double-click the instance. Each instance you have selected is opened on-line and detailed status information is displayed.
4. To monitor more instances, change to the status overview and click the instance you want.
5. To stop monitoring the program status, deactivate the menu option *Debug -> Monitor*.

Programming in the S7- SCL High-Level Language

S7-SCL: High-level language for writing PLC programs

- Compatible with IEC 1131-3 Text (ST=Structured Text))
- Certified to PLCopen Base Level
- Contains all the typical elements of a high-level language, such as operators, expressions, control statements
- PLC functions are integrated (e.g. I/O access, timers, counters...)

Advantages:

- Well structured, easy-to-read programs
- For high-level language users
- For complex algorithms, large amounts of data

```

FUNCTION_BLOCK Integrator
VAR_INPUT
    Init    : BOOL;    // Reset output value
    x       : REAL;    // Input value
    Ta      : TIME;    // Sampling time in ms
    Ti      : TIME;    // Integration time in ms
    ulim    : REAL;    // Output value upper limit
    llim    : REAL;    // Output value lower limit
END_VAR

VAR_OUTPUT
    y : REAL:= 0.0;    // Initialize output value with 0
END_VAR

BEGIN
    IF TIME_TO_DINT(Ti) = 0 THEN    // Division by ?
        OK := FALSE;
        y := 0.0;
        RETURN;
    END_IF;
    IF Init THEN
        y:= 0.0;
    ELSE
        y := y+TIME_TO_DINT(Ta)*x/TIME_TO_DINT(Ti);
        IF y > ulim THEN y := ulim; END_IF;
        IF y < llim THEN y := llim; END_IF;
    END_IF;
END_FUNCTION_BLOCK
  
```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.33



Information and Training Center
Knowledge for Automation

S7-SCL

SCL (Structured Control Language) is a PASCAL-similar high-level textual language. It simplifies the programming of mathematical algorithms and complex data processing tasks for PLCs.

SCL therefore also enables S7 PLCs to be used for more complex tasks such as closed-loop control or statistical evaluation.

The SCL program is created and stored in an SCL source (source file). Executable blocks are then generated during compilation.

SCL is compatible with the ST (Structured Text) language defined in IEC 1131-3 and has PLCOpen certification (Base Level)

Functionality

SCL offers the functional scope of a high-level language such as:

- Loops
- Alternatives
- Branch distributors, etc.

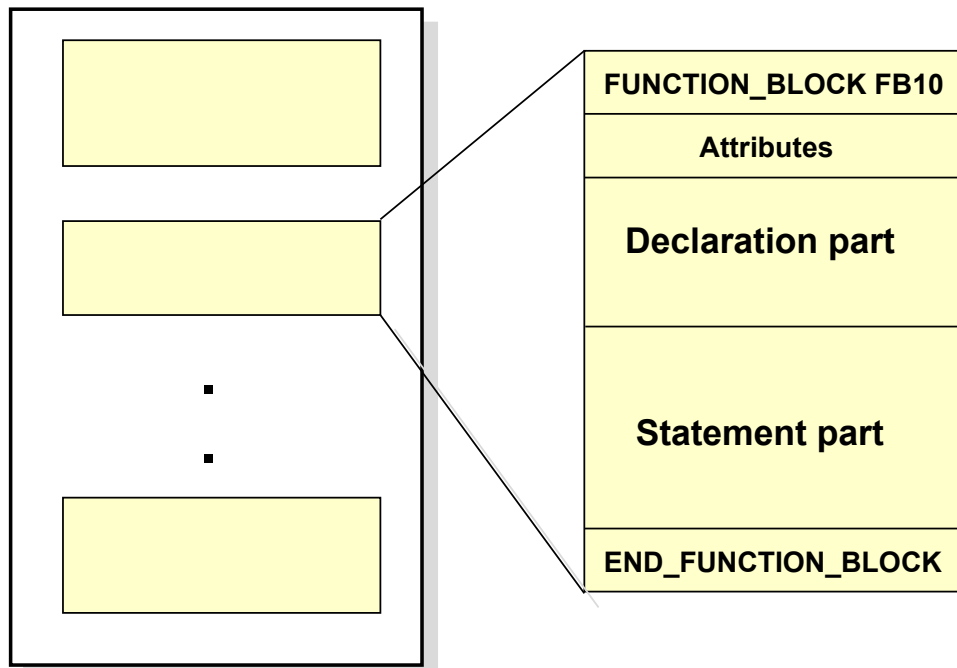
combined with PLC-specific functions such as:

- Bit accesses to the I/O, bit memories, timers, counters etc.
- Access to the symbol table
- STEP 7 block access

Advantages of SCL

- Simple to learn programming language, especially for beginners
- Simple to read (understandable) programs are created.
- Simpler programming of complex algorithms and processing of complex data structures
- Integrated debugger for symbolic debugging of the source code (single-step, breakpoints, etc.).
- System integration in S7 languages, such as STL, LAD and FBD.
- Relatively easy for PLC technicians to understand through similarity with S7 languages.

Structure of an SCL Source File



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.34Information and Training Center
Knowledge for Automation

Structure of an SCL Source File

An SCL source file can include as many blocks as you like (OBs, FBs, FCs, DBs and UDTs).

Structure of a Block

Within the SCL source file, every individual block, dependent on the block type, is framed by a standard identifier for the beginning and for the end of the block. The body of the block itself consists of the declaration part and the statement part.

As an option, an attribute part can be inserted between the identifier for the beginning of the block and the declaration part.

Attributes

Attributes identify block properties, that can also be displayed within the SIMATIC Manager after compilation via the command *Edit -> Object Properties*.

Declaration Part

The local variables, block parameters, constants and jump labels are declared in the declaration part of a block.

Statement Part

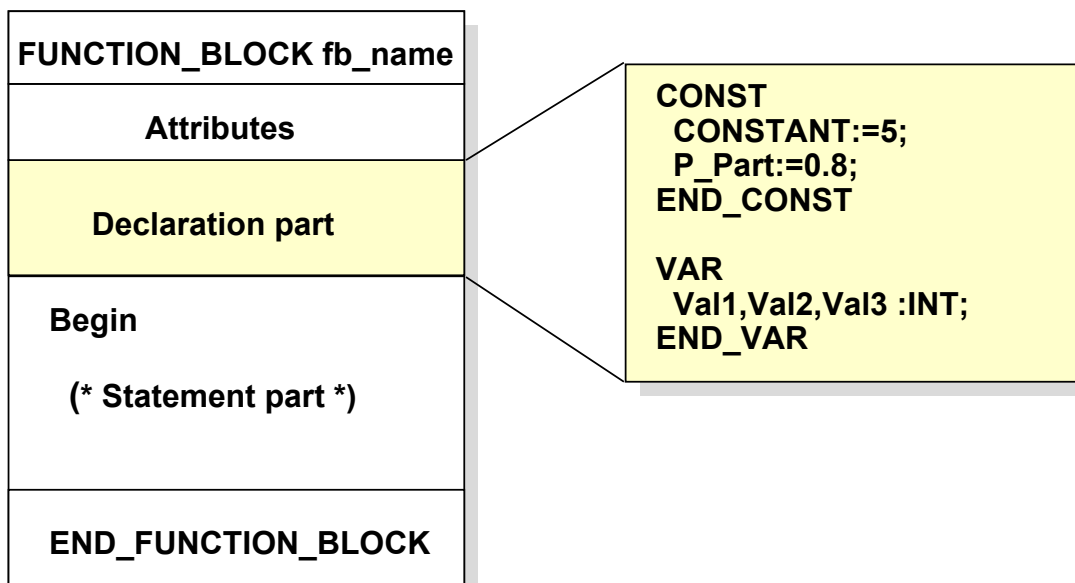
The statement part contains the individual statements to be executed.

Block Sequence

So that your SCL source file can be compiled, you must pay attention to the following in regard to the sequence of the blocks:

Called blocks must always be located before the calling blocks.

The Declaration Part of a Block



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.35Information and Training Center
Knowledge for Automation

Structure

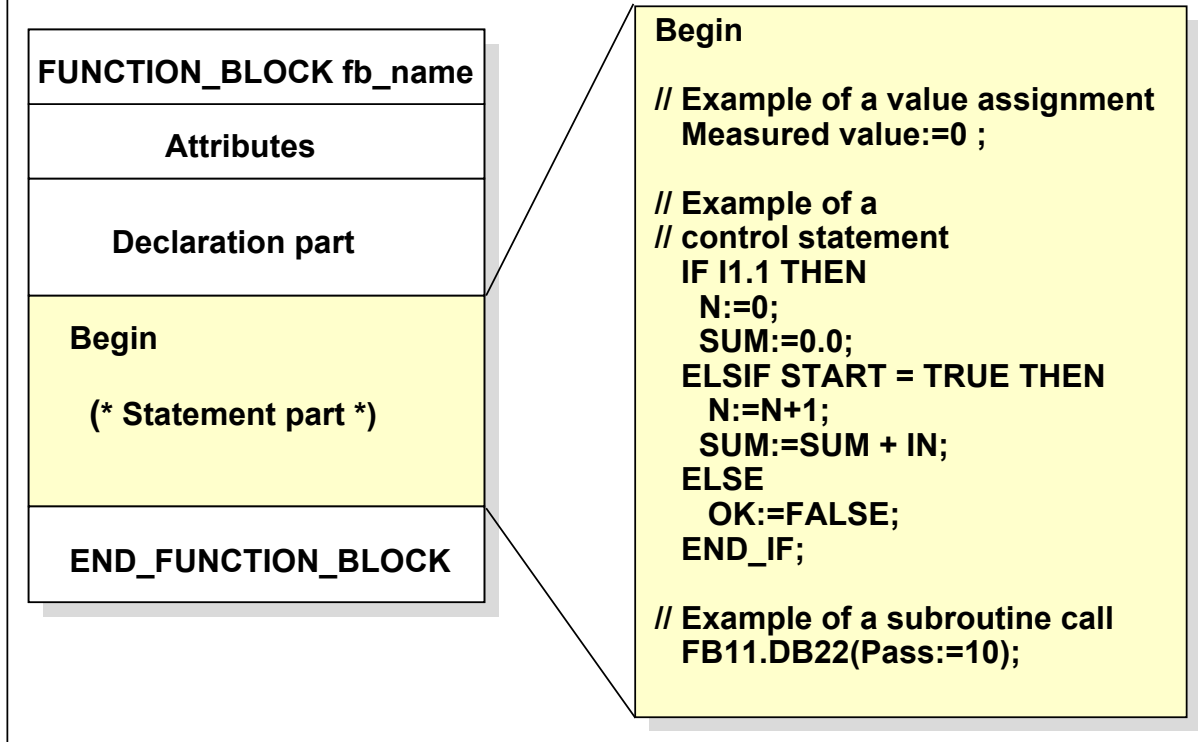
The declaration part is used to define the local and global variables, block parameters, constants and jump labels. It is divided into individual declaration blocks, that are in each case identified by their own keyword pair.

The blocks can be inserted into an SCL source file via *Insert -> Block Template -> Constant, Parameter*.

Blocks

Data	Syntax	FB	FC	OB	DB	UDT
Constants	CONST <i>Declaration list</i> END_CONST	X	X	X		
Jump Labels	LABEL <i>Declaration list</i> END_LABEL	X	X	X		
Temporary Variables	VAR_TEMP <i>Declaration list</i> END_VAR	X	X	X		
Static Variables	VAR (STRUCT) <i>Declaration list</i> END_VAR	X			(X)	(X)
Input Parameter	VAR_INPUT <i>Declaration list</i> END_VAR	X	X			
Output Parameter	VAR_OUTPUT <i>Declaration list</i> END_VAR	X	X			
In/Out Parameter	VAR_IN_OUT <i>Declaration list</i> END_VAR	X	X			

The Statement Part of a Block



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.36Information and Training Center
Knowledge for Automation

Statement Part

The statement part contains instructions that are executed after the call of a logic block (OB, FB, FC). These statements are used to process data and addresses or - in the case of data blocks - to preset individual values within the DB.

Subdivision

The individual statements can essentially be divided into three groups:

- Value assignments: they are used to assign an expression or a value to a variable.
- Control statements: they are used to branch within a program or to repeat groups of instructions.
- Subroutine call: they are used to call functions and function blocks.

Note

You must pay attention to the following points when programming statements:

- The statement part begins with the keyword **BEGIN** and ends with the keyword for block end (e.g. **END_FUNCTION**).
- Every statement must be closed with a semicolon.
- All identifiers (names) used in the statement part must be declared.

Templates

Templates for control structures can be inserted in an SCL source file via *Insert* -> *Control Structure* -> *IF*, *CASE*, *FOR*, *WHILE*, *REPEAT*.

Expressions, Operators and Operands in S7-SCL

Expressions

- | | |
|----------------------------|--|
| • Mathematical expressions | $((3 + \text{CONST_INT}) * (\text{VAR_INT} ** 37) / 3.14)$ |
| • Comparison expressions | $A \geq 9$ |
| • Logical expressions | $(n > 5) \text{ AND } (n < 20)$ |

Operators

- | | |
|--------------------------|--|
| • Assignment operator | $:=$ |
| • Mathematical operators | $*, /, \text{MOD}, \text{DIV}, +, -, **$ |
| • Comparison operators | $<, >, <=, >=, =, <>$ |
| • Logical operators | $\text{NOT}, \text{AND or } \&, \text{XOR}, \text{OR}$ |

OPERANDS

- | | |
|-------------------------------------|---|
| • Constants | 30. 0, FACTOR, 'SIEMENS' |
| • Extended variables
FC12(A:=On) | Status, IB5, DB10.DW5, Motor.Current, |
| • Expressions in (...) | $((3 + \text{CONST_INT}) * (\text{VAR_INT} ** 37))$ |

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.37



Information and Training Center
Knowledge for Automation

Expressions

Expressions consist of operands, operators and round brackets. Within an expression the operators (e.g. +, -, *, /, etc.), that is, the active components of an expression, are linked with the passive elements, such as constants, variables and function values, in order to form a new value.

An expression therefore stands for the value it represents.

SCL permits the formation of standard expressions, that is, mathematical, logical and comparative expressions. Variables from data blocks, arrays, structures and CPU memory areas (inputs, outputs, etc.) can be enlisted for the formation.

Operators and Operands

Expressions consist of operators and operands. Most SCL operators link two operands (e.g. $A + B$) and are therefore termed *binary* operators. The others work with only one operand and are thus called *unary* operators.

The result of an expression can

- be assigned to a variable (e.g. $A := B + C;$)
- be used as a condition for a control statement (e.g. IF $A < B$ DO ...)
- be used as an actual parameter for the call of a function or a function block (e.g. FB20 (Input := $A + B$))

Statements in S7-SCL

Value assignments

- Example: **A := B + C;**

Control statements

- IF statement **IF E1.1 THEN ... ELSIF ... ELSE ... END_IF**
- CASE statement **CASE SELECTOR OF 1: ...; 2: ... ELSE: ... END_CASE**
- FOR statement **FOR INDEX := 1 TO 49 BY 2 DO ... END_FOR**
- WHILE statement **WHILE INDEX <= 50 DO ... END_WHILE**
- REPEAT statement **REPEAT ... UNTIL INDEX:= 51 ... END_REPEAT**
- CONTINUE statement **WHILE BOOL_1 DO ... CONTINUE ... END_WHILE**
- EXIT statement **WHILE BOOL_1 DO ... EXIT ... END_WHILE**
- GOTO statement **IF INDEX <23 THEN GOTO MARK; ...**
- RETURN statement **IF ENABLED THEN RETURN; ...**

Function and function block calls

- FB or SFB call **FB11.DB20(IN:=VAL1, BY:=VAL2);**
- FC or SFC call **RETURN := FC32(IN:=VAL1,OUT:=VAL2);**

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.38Information and Training Center
Knowledge for Automation

Statements

In the declaration part of a block, the actions are described that are to be executed with the variables introduced in the declaration part as well as with global data. Normally, the statements are executed in the sequence in which they are listed in the program text.

Value Assignments

These are used for assigning new values to variables. The variable's old value is then lost.

Control Statements

These are used for changing the sequence in which the statements are normally processed.

A choice from the various alternatives in program execution can be made with conditional statements (IF and CASE statements).

Loop statements (FOR, WHILE and REPEAT statement) are used to repeatedly execute statements.

Jump statements (CONTINUE, EXIT and GOTO statements) permit the sequence of processing to be interrupted and to jump to a resumption point.

FB and FC Calls

According to the principle of structured programming, other functions (FC and SFC) and function blocks can also be called from an SCL block. Blocks that can be called are:

- additional functions and function blocks, that were generated in SCL or in another STEP 7 language (STL, LAD, etc.).
- standard functions and standard function blocks supplied with SCL.
- system functions (SFC) and system function blocks (SFB) that are available in your CPU's operating system.

Value Assignments in S7-SCL

Local variables

- **Elementary data type** COUNTER := (5 + RUNVAR) * 2;
- **Structures**
 - Complete structure STRUCT_1 := STRUCT_2;
 - Components STRUCT_1.COMP3 := STRUCT_2.COMP1;
- **Array**
 - Complete array ARRAY_1 := ARRAY_2;
 - Components ARRAY_1[I] := ARRAY_2 [J];

Global variables

- **CPU memory areas**
 - Absolute access VALUE := IW10;
 - Symbolic VALUE := INPUT; // "Input" in symbol table
 - Indexed VALUE := IW[INDEX];
- **Data blocks**
 - Absolute access VALUE := DB11.DW5;
 - Symbolic VALUE := MOTOR.CURRENT; // MOTOR and CURRENT
 - Indexed VALUE := MOTOR.DW[Index]; // must be in the symbol table
 - Via input parameters VALUE := I_PAR.DW[Index]; // I_PAR is decl. as VAR_IN

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.39Information and Training Center
Knowledge for Automation

Principle

Value assignments replace a variable's present value with a new value, which is specified in an expression. This expression can also contain identifiers of functions (FC), which are thereby called and return corresponding values.

The value of an expression assigned to a variable must be compatible with the variable's type.

Value Assignments with Complex Variables

A complex variable represents either the complete type (the complete structure, the complete array, the string) or a component of the complex variable. There are thus two possibilities for assignment to a complex variable. You can

- Assign the contents of another complete complex variable (structure, array, or string) to each complex variable (structure, array, string).

Please note that, for example, a complete structure can only be assigned to another structure if the structure components coincide in their data type as well as in their name.

A complete array can only be assigned to another array if the component's data types as well as the array limits coincide exactly.

- Assign a type-compatible variable, a type-compatible expression or another component to each component of a complex variable.

The IF Statement in S7-SCL

Syntax

```

IF      <expression> THEN <statements>
[ELSIF <expression> THEN <statements>]      //optional
.
.
[ELSE <statements>]                          //optional
END_IF

```

Example

```

IF INPUT_OK THEN
    N := 0;
    SUM := 0.0;
    OK := FALSE;                // Set OK flag to FALSE
ELSIF START_OK THEN
    N := N + 1;
    SUM := SUM + IN;
ELSE
    OK := FALSE;
END_IF;

```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.40



Information and Training Center
Knowledge for Automation

Principle

The problem often occurs in programs, that various statements are to be executed dependent on specific conditions. It is possible, with program branches, to branch the program flow into alternative statement sequences.

The IF statement is a conditional statement. It offers one or more options and selects one of its statements for execution (or none, if applicable).

Execution

The IF statement is processed according to the following rules:

1. If the value of the first expression is TRUE, then the statement part after THEN is executed, otherwise, the expressions in the ELSIF branches are evaluated.
2. If no boolean expression is TRUE in the ELSIF branches, the statement sequence for ELSE is executed (or no statement sequence, if the ELSE branch does not exist).

Any number of ELSIF statements may exist. You must note that the ELSIF branches and/or the ELSE branch may be missing. These cases are handled as if these branches existed with empty instructions.

Note

The use of one or more ELSIF branches as opposed to a sequence of IF statements offers the advantage that the logical expressions that follow a valid expression are no longer evaluated. The run-time of a program can thus be shortened.

The WHILE Statement in S7-SCL

Syntax

```
WHILE <expression> DO <statements>  
END_WHILE
```

Example

```
FUNCTION_BLOCK SEARCH           // SEARCH is declared in the symbol table  
VAR  
    INDEX      : INT;  
    KEYWORD    : ARRAY[1..50] OF STRING;  
END_VAR  
  
BEGIN  
    INDEX := 1;  
    WHILE INDEX <= 50 AND KEYWORD[INDEX] <> 'KEY'  
    DO  
        INDEX := INDEX + 2;  
    END_WHILE;  
  
END_FUNCTION_BLOCK
```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.41Information and Training Center
Knowledge for Automation

Principle

The WHILE statement permits the repeated execution of a sequence of statements on the basis of an execution condition. The execution condition is formed according to the rules of a logical expression.

The statement part that follows DO is repeated as long as the execution condition has the value TRUE.

Execution

The WHILE statement is processed according to the following rules:

1. The execution condition is evaluated before every execution of the statement part.
2. If the value TRUE occurs, then the statement part is executed.
3. If the value FALSE occurs, the execution of the WHILE statement is concluded. This can already be the case with the first evaluation.

Calling Function Blocks

Calling with instance DB

- **Absolute call**
 FB10.DB20(X1 := 5, X2 := IW12, ...); (* Call FB10 with instance data block DB20 *)
- **Symbolic call**
 DRIVE.ON(X1 =5, X2 := IW12,...); (* DRIVE and ON are declared in the symbol table *)

Calling as a multiple instance

- **Call using identifier**

```

VAR
  MOTOR          : FB10;
END_VAR

BEGIN
  .MOTOR(X1 := 5, X2 := IW12,...);
      
```

 (* Calling as a multiple instance is only possible within other function blocks *)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.42



Information and Training Center
Knowledge for Automation

FB Calls

The multiple instance concept offered by STEP 7 is also available in SCL. You can therefore call FBs referencing an associated instance data block or using the multiple instance model.

The call of an FB as a multiple instance differs from the call with separate DB in the storage of the data.

In the case of a multiple instance, the necessary data area is not in a separate DB, but is embedded in the instance data area of the calling FB.

Call with Instance DB

The call is made in a statement specifying:

- the name of the function block or system function block
- the instance data block (DB identifier)
- as well as the parameter assignment (FB parameters)

You can use either absolute or symbolic addresses in the call with a separate DB. You can call FBs with a separate instance DB in all logic blocks (OB, FB, FC).

Call as Multiple Instance

The call is made in a call statement specifying:

- the local name of the instance (identifier)
- as well as the parameter assignment (FB parameters).

The call of an FB as a multiple instance is only possible in FBs. The multiple instance must also be declared as a variable of data type FBx in the declaration part (VAR ...END_VAR) of the calling FB.

The "OK" Flag for Error Evaluation

**Global bit for error detection
(Copied to the BR bit at the end of the block)**

Example:

```

// Set OK variable to TRUE to enable
// a check to be made to see whether
// the following actions are performed
// correctly

OK := TRUE;
SUM := SUM + IN;
IF OK THEN      // Addition was performed correctly
...
ELSE           // Overflow in addition
...
END_IF;
  
```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.43



Information and Training Center
Knowledge for Automation

Description

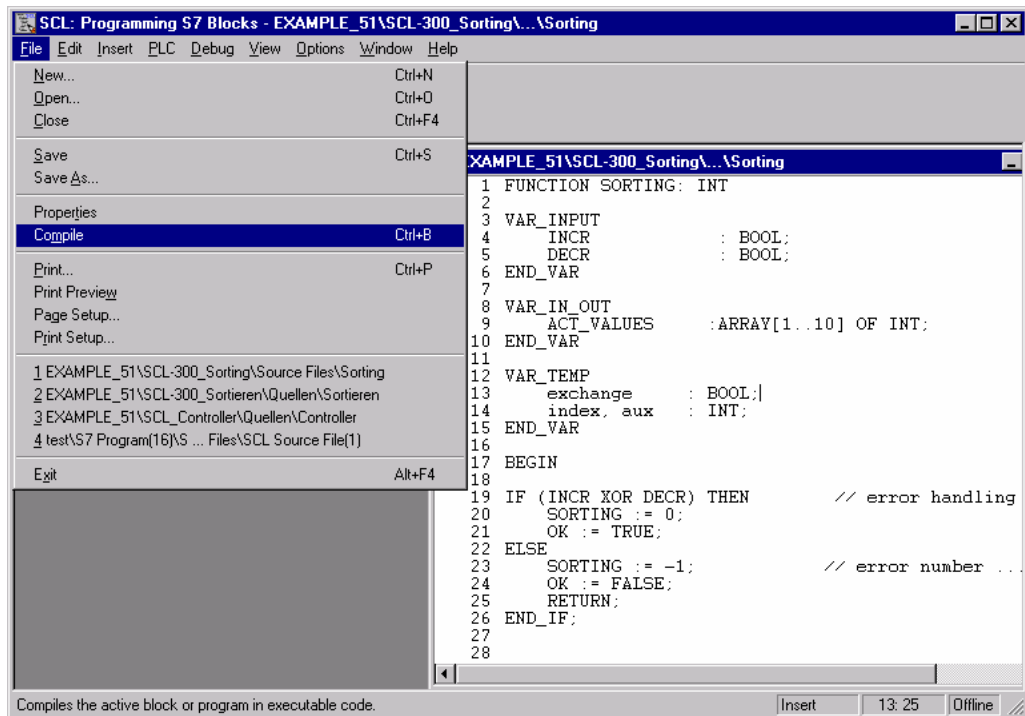
SCL provides a global variable of the BOOL type, known as the OK Flag, for error identification within logic blocks. This flag is used to identify correct or faulty executions of statements and to react accordingly.

Method of Functioning

If an error occurs during the execution of a statement (e.g. overflow), the OK flag is set to FALSE by the system. Upon exiting a block, the OK flag is then copied into the BR bit of the CPU status word and can then be evaluated with the OK flag of the calling block.

The OK flag is set to the value TRUE at the beginning of a program execution. It can be checked anywhere in the block or be set to TRUE or FALSE.

Compiling an SCL Source File



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.44



Information and Training Center
Knowledge for Automation

Compiling

Before you can run or test your program, you must compile it. You activate the compiling function via the *Compile* option in the *File* Menu or via the icon in the toolbar.

The compiler has the following properties:

- The compiler operates in batch mode, that is, it processes an SCL source as a unit. The compilation of individual blocks in an SCL source file is not possible.
- The compiler checks the syntax of an SCL source file and subsequently displays all errors that occurred during compilation.
- The compiler creates STL statements or test information, if the SCL source is error-free and the corresponding options are set. You must select the Create Debug Info option for every program that you subsequently want to test in a high-level language.
- The compiler generates for every function block call an associated instance data block.

Settings

You adapt the compiling function via the entry *Options -> Customize -> Compiler* in the Options menu:

- Create Object Code: You use this option to specify whether or not you want to generate executable code. If you activate the Compile function without selecting this option, only a syntax check is performed.
- Optimize Object Code: Create shorter code
- Monitor Array Limits: Array indexes are checked for allowed range at runtime. If an array index is outside the permissible range, the OK flag is set to FALSE.
- Create Debug Info: Generate debug information for high-level language debugger
- Set OK Flag: This option enables the OK flag to be checked in an SCL source file.

Continuous Monitoring

```

18
19 IF (INCR XOR DECR) THEN          // error handling
20   SORTING := 0;
21   OK := TRUE;
22 ELSE
23   SORTING := -1;                  // error number
24   OK := FALSE;
25   RETURN;
26 END_IF;
27
28
29
30 IF INCR = TRUE THEN
31   REPEAT
32     exchange := FALSE;
33     FOR index := 2 TO 10 BY 1 DO
34       IF ACT_VALUES[index-1] > ACT_VALUES[index] THEN
35         aux := ACT_VALUES[index];
36         ACT_VALUES[index] := ACT_VALUES[index-1];
37         ACT_VALUES[index-1] := aux;
38         exchange := TRUE;
39       END_IF;
40     END_FOR;
41   UNTIL NOT exchange
42   END_REPEAT;
43 ELSEIF DECR = TRUE THEN
44   SORTING := 0;
45   OK := TRUE;
46 END_IF;
47

```

Press F1 for help. Blockstatus 19: 6 Online

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.45



Information and Training Center
Knowledge for Automation

Selection

The Debugger function *Monitor Continuously* can be selected as follows:

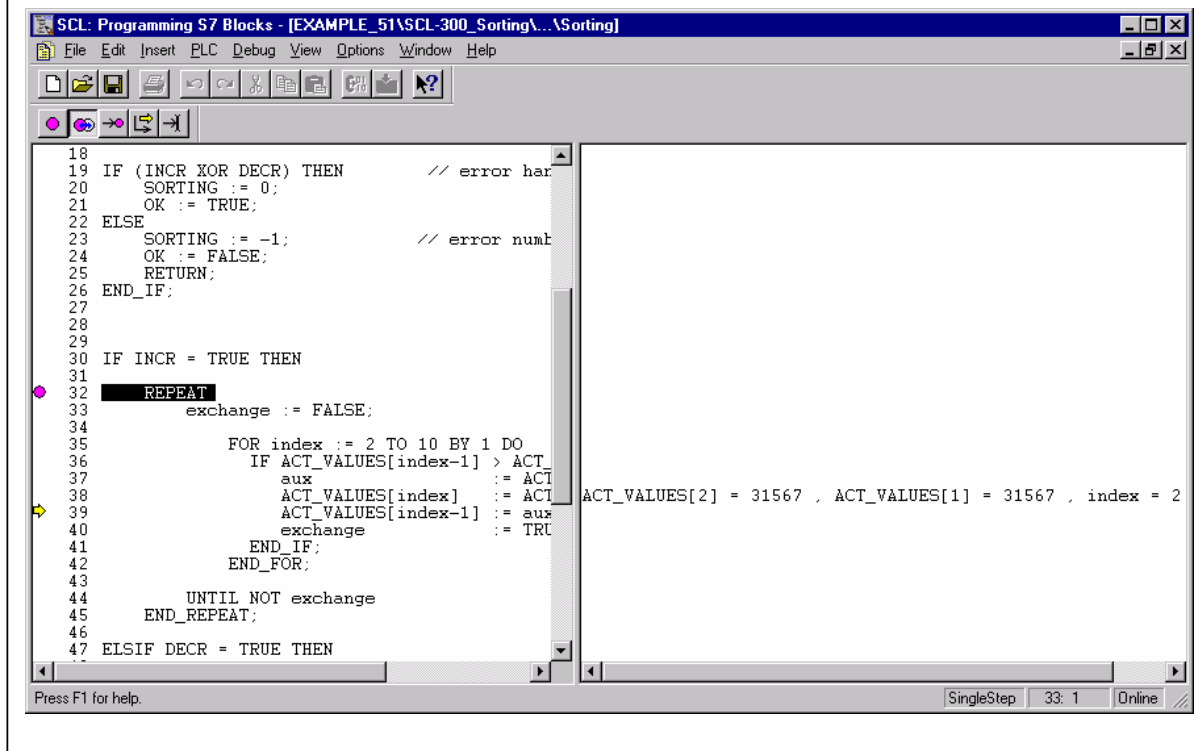
1. Make sure that the program has been compiled with the options "Create Object Code" and "Debug Information" activated.
2. Select the window containing the source of the program to be tested.
3. Position the cursor in the line of the source text containing the first statement of the section you want to test.
4. Select the menu options *Debug -> Monitor Continuously*.
The largest area that can be monitored is determined and indicated by a gray bar at the left-hand edge of the window. The names and current values of the variables in the area being monitored are displayed in the right-hand section of the window.
5. Select the menu options *View -> Symbolic Representation* to activate or deactivate the display of symbolic names in the program.
6. Select the menu options *Debug -> Monitor Continuously* if you want to interrupt monitoring.
7. Select the menu options *Debug -> Finish Testing* to stop monitoring.

Debug Mode

You can change the monitoring area with the entry *Debug -> Test Environment*:

- Process: In this test environment, the SCL debugger reduces the maximum monitoring area so that the cycle time is not or only negligibly prolonged.
- Laboratory: In the Laboratory test environment, the monitoring area is only restricted by the capacity of the CPU. The maximum monitoring area is larger than in the Process test environment.

Setting and Editing Breakpoints



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.46



Information and Training Center
Knowledge for Automation

Overview

When you select the debugging mode "Breakpoints Active", you can monitor your program step by step. You can execute the program one statement at a time and observe the changes in the contents of the variables processed.

After setting breakpoints, you can execute the program up to a breakpoint and monitor it step by step from there on.

Setting Breakpoints

You set breakpoints as follows:

1. Open the SCL source of the program you want to test.
2. Define the breakpoints by positioning the cursor to the required place and selecting the menu options *Debug -> Set Breakpoint*. The breakpoints appear as red circles at the edge of the window.
3. Select the menu options *Debug -> Breakpoints Active*. The window is vertically divided into two halves. When the next breakpoint is reached, the CPU goes into HALT mode, the red breakpoint is marked with a yellow arrow.
4. To continue:
 - Select the menu options *Debug -> Execute Next Statement*. When the CPU has processed the next statement, it goes into HALT mode again.
 - Select the menu options *Debug -> Continue*. When the CPU reaches the next breakpoint, it goes into HALT mode again.
 - Select the menu options *Debug -> Execute To Cursor*. When the CPU reaches the selected point in the program, it goes into HALT mode again.

When the required place is reached, the contents of the variables currently being processed are displayed in the right-hand pane of the window.

Number of Breakpoints

The number of active breakpoints depends on the CPU:

- CPU 416: up to 4 active breakpoints
- CPU 414: up to 2 active breakpoints
- CPU 314: one active breakpoint

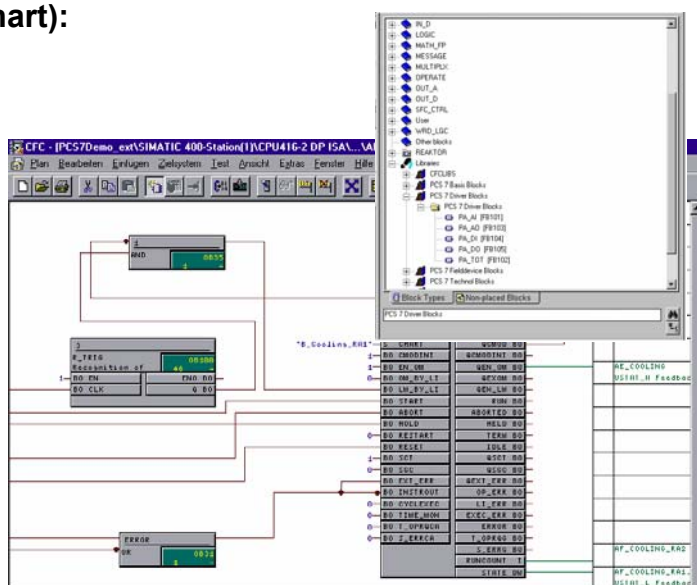
CFC for SIMATIC S7 and SIMATIC M7

CFC (Continuous Function Chart): Graphical tool for writing PLC programs

- **You position the blocks on a drawing sheet and interconnect them**
- **Interconnections are possible:**
 - **between I/O fields**
 - **to blocks in other charts**
- **Margin bar for management of sources and destinations**

Advantages

- **Programming for process engineers**
- **Speeds up writing, debugging and startup**



SIMATIC S7

Siemens AG 1999. All rights reserved

Date: 04.11.2005
File: PRO2 13D.47

Information and Training Center
Knowledge for Automation

Overview

The CFC (Continuous Function Chart) engineering tool enables you to create automation applications for SIMATIC S7 or SIMATIC M7 by drawing a process flow chart - similar to a function diagram for programming a PLC.

In this graphical programming method, you position the blocks in an area like a sheet of drawing paper and interconnect them graphically. With CFC you can quickly and easily convert technological requirements into executable automation programs.

Scope of Supply

The scope of supply of CFC includes:

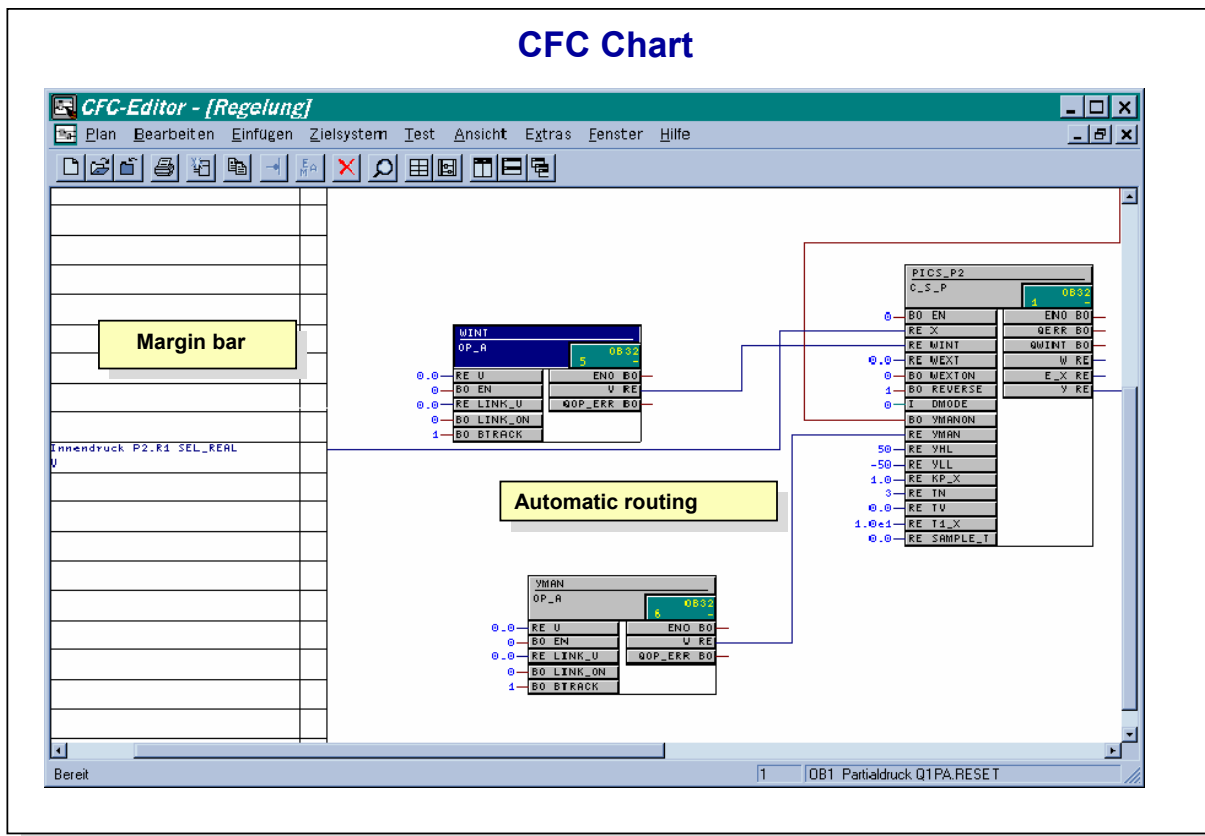
- CFC Editor
- Code generator
- Debugger
- Standard block libraries

Customer Benefits

The CFC product is smoothly integrated into the STEP 7 architecture as an optional package with a unified look&feel, and with universal data management. This makes CFC easy to use and easy to learn, and provides consistent data.

- CFC can be used for simple tasks as well as for very complex task definitions.
- Simple interconnection technology makes communication between blocks user-friendly to configure.
- Manual handling and management of machine resources is no longer necessary.
- User-friendly testing and debugging are supported.

CFC Chart



SIMATIC S7

Siemens AG 1999. All rights reserved.

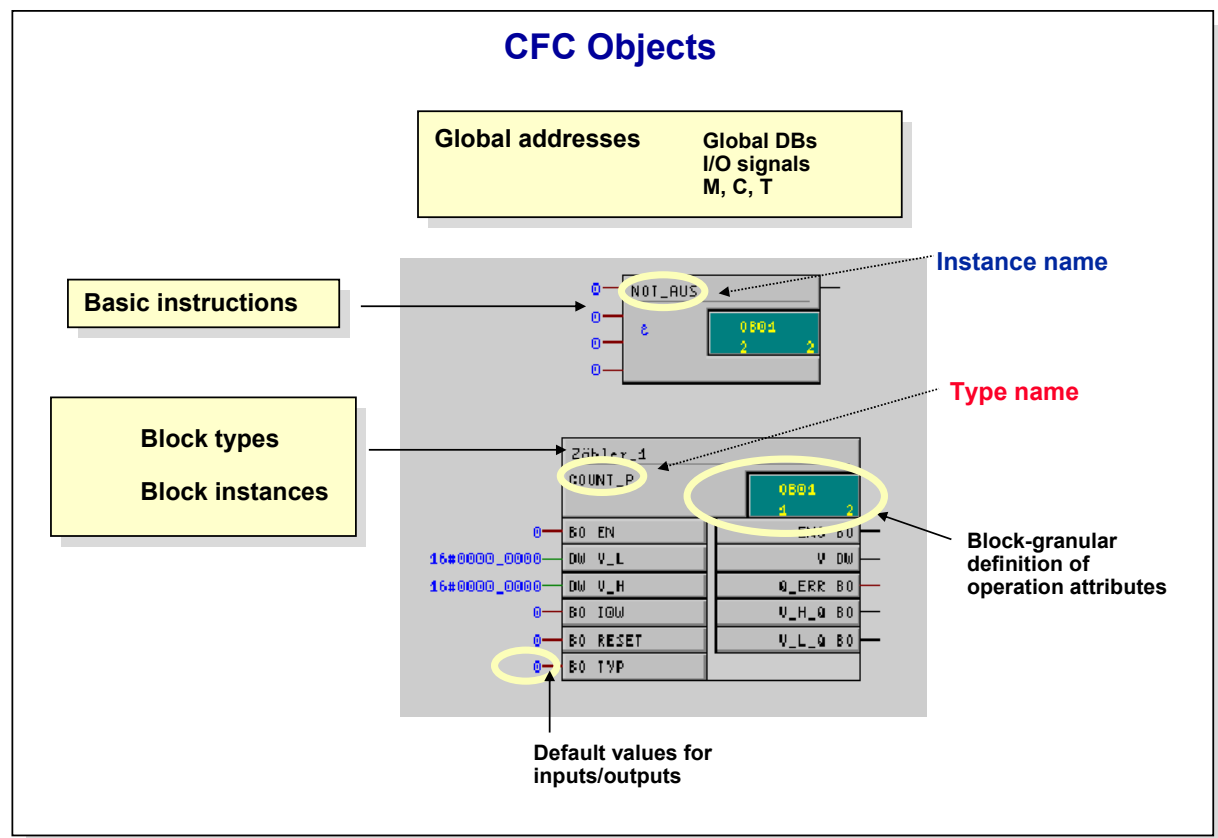
Date: 04.11.2005
File: PRO2_13D.48Information and Training Center
Knowledge for Automation

CFC Charts

The block instances you need to solve a technological task definition can be divided into any number of charts.

A CFC chart consists of six pages (overview display)

- 1 page consists of a work area and two marginal bars.
- Automatic, chart-spanning marginal bar management
- User-friendly signal monitoring
- Autorouter
- Resources are completely managed for the user.
- 1 to 1 documentation for the entire information contents



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.49



Information and Training Center
Knowledge for Automation

CFC Objects

The most important CFC terms are outlined here.

Block types

A block type represents a template for any number of instances and describes how these instances are structured internally. All instances of a block type obey the same basic definition as regards their characteristics and their data structure.

Block Instances (blocks)

A block instance is a concrete object generated according to its type description. The type describes the characteristics and information structure for the instance while the current state of each instance depends on its actually executed operations and is reflected in the information contents. Each instance has a unique identifier that enables instances to be distinguished from one another.

In CFC, the identifier for a block instance is made up of the chart name, which is unique in the CPU, the separator '.', and the block name, which is unique within the chart (maximum of 24 characters for the block name).

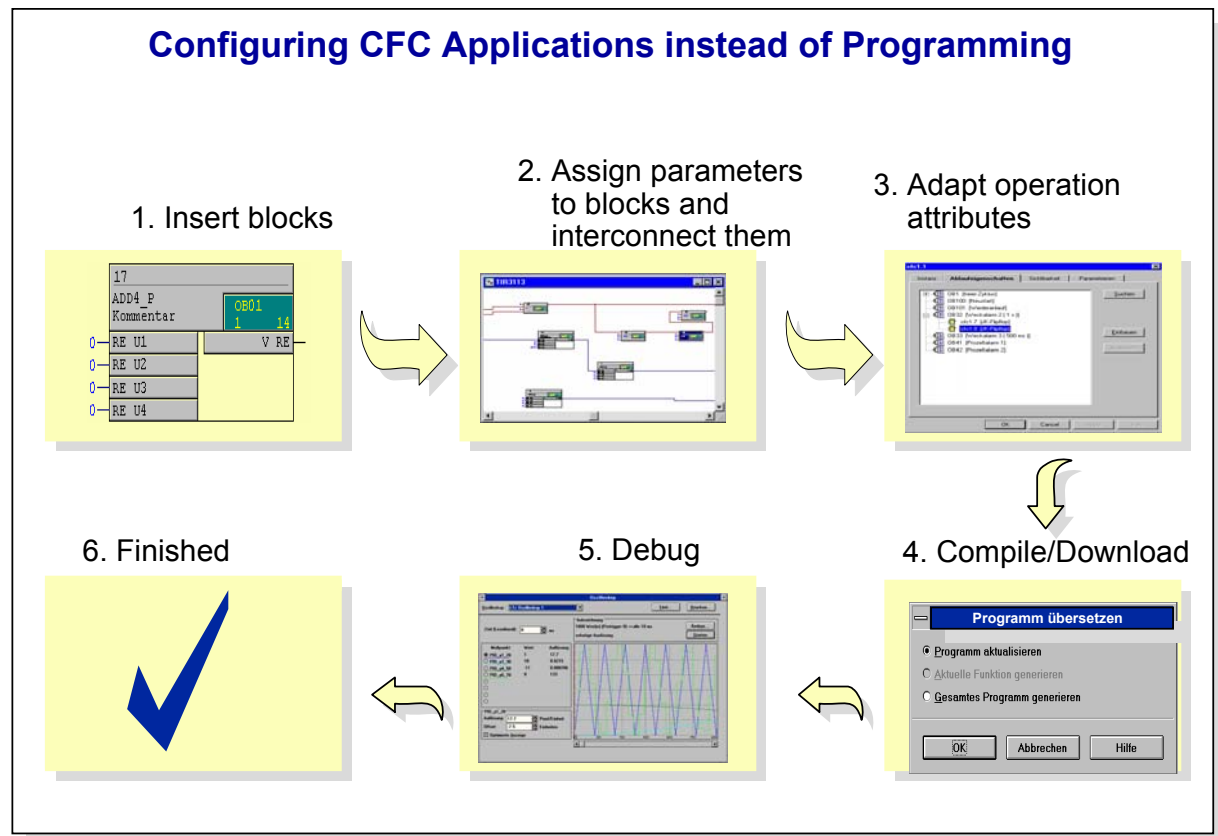
Blocks

In the STEP 7 language usage, blocks are separate parts of the user program defined by their function, their structure, or their application purpose.

There are logic blocks (FB, FC,...), data blocks, and user-defined data types.

- Basic instructions: Functions such as AND, SUM, etc. contained in the S7 machine model
- Global addresses: I/O signals, bit memories, counters, timers, and global data blocks

Configuring CFC Applications instead of Programming



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.50



Information and Training Center
Knowledge for Automation

Configuring CFC

You configure a CFC application as follows:

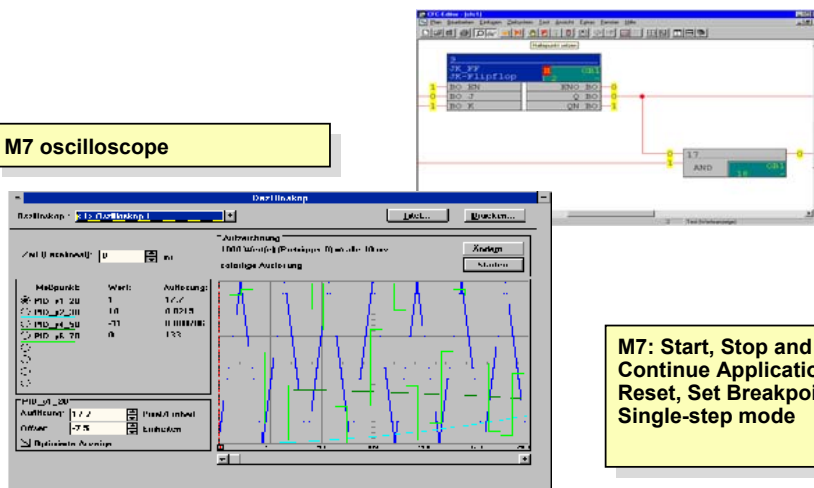
1. Create a project with a charts folder in the S7/M7 user program
2. Create or copy the block types
3. Create a CFC chart
4. Insert blocks
5. Assign parameters to blocks and interconnect them
6. Adapt the operation attributes of the blocks
7. Compile the CFC charts you have created
8. Download compiled program to CPU
9. Test the downloaded program

Integral Test and Debugging Functions

Parameterize variable

Monitor variable

M7 oscilloscope



M7: Start, Stop and Continue Application, Reset, Set Breakpoints, Single-step mode

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.51

Information and Training Center
Knowledge for Automation

CFC Test and Debug The following test and debug facilities are available in CFC:

- Monitor and set parameters for values of variables in the chart.
- Program flow control with breakpoints (only M7)
- Cycle-exact recording of variable values (only M7)

Configuring Sequential Control Systems with S7- SFC

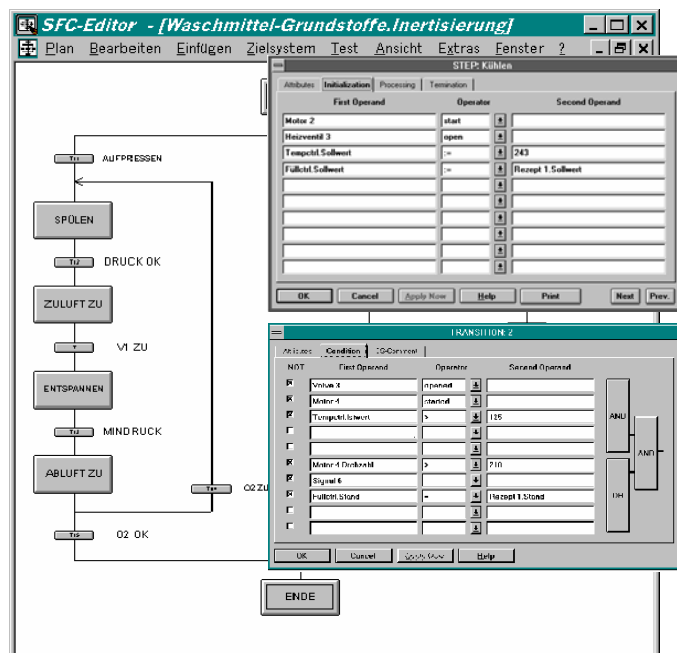
S7-SFC: Tool for programming sequencers

- Designed for the requirements of process automation
- Compatible with IEC 1131-3
- Steps assign values to the blocks in the CFC
- Transitions check the step-enabling conditions
- Syntax checked during writing

Direct link with CFC

- Transfer of values by "Drag&Drop"
- Cross references

Visualization in WinCC



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.52



Information and Training Center
Knowledge for Automation

SFC (Sequential Function Chart)

SFC is a sequential control system that always operates step by step, which was specially designed for the requirements of process automation (process engineering, process control, etc.).

The typical fields of application for sequential control systems of this type are plants with discontinuous operation. Sequential control systems can however also be used in plants which operate continuously, for example, for startup or shutdown, working point changes as well as state changes due to disturbances etc.

With SFC, for example, product manufacturing specifications can be written as event-driven processes.

Principle of Operation

In the SFC-Editor you generate the flow chart by graphic means. The structure elements of the chart are positioned according to fixed rules. You do not have to worry about details such as algorithms or the allocation of machine resources, but instead can concentrate on the technological aspects of the configuration.

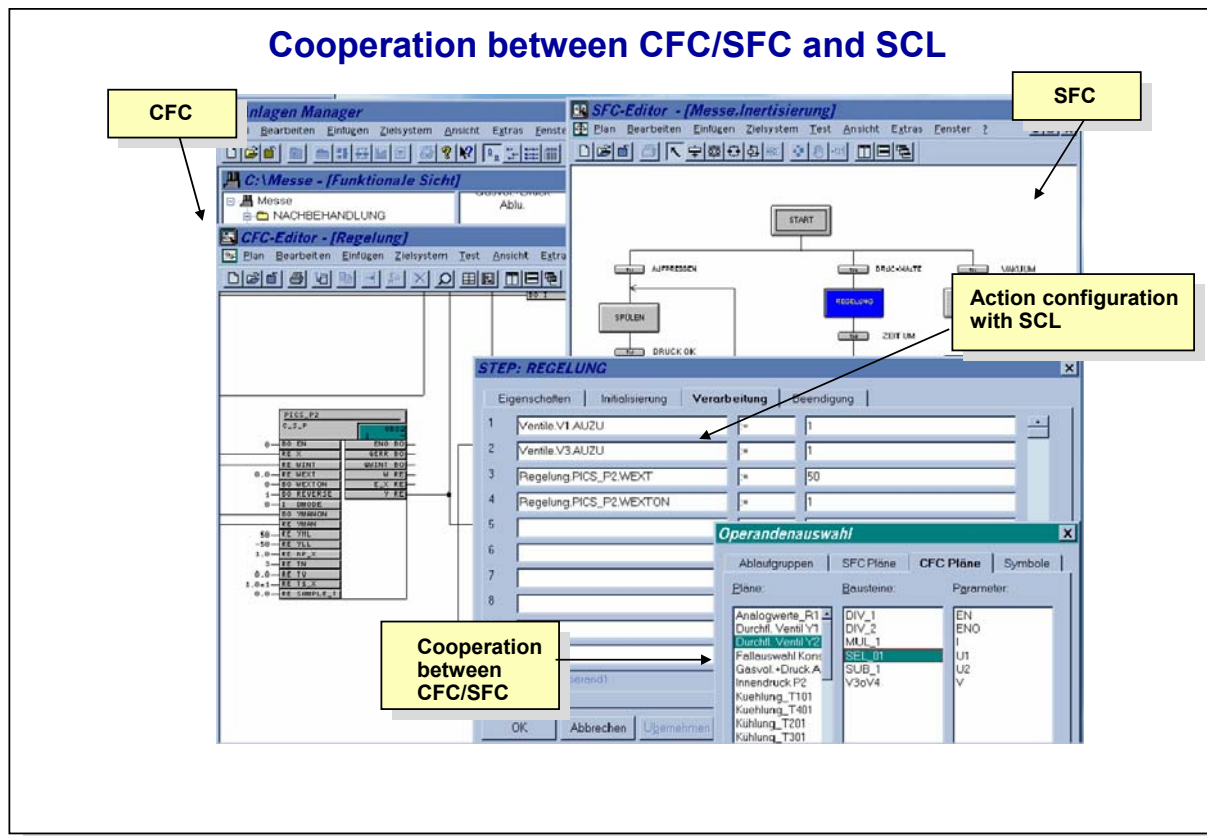
After generating the chart topology, you switch to the detailed view (zoom) and there assign parameters to the individual elements, that is, you configure the actions (steps) and conditions (transitions).

You normally program actions by selectively editing basic automation functions created with CFC using mode and status changes.

After configuration, generate the executable machine code with SFC, download it into the PLC and test it with the SFC debugging functions.

Volume of Project Data

- Sequencers per chart 1
- Steps per chart 2 ... 255
- Transitions per chart 1 ... 255
- Instructions per step <= 50
- Conditions per transition <= 10



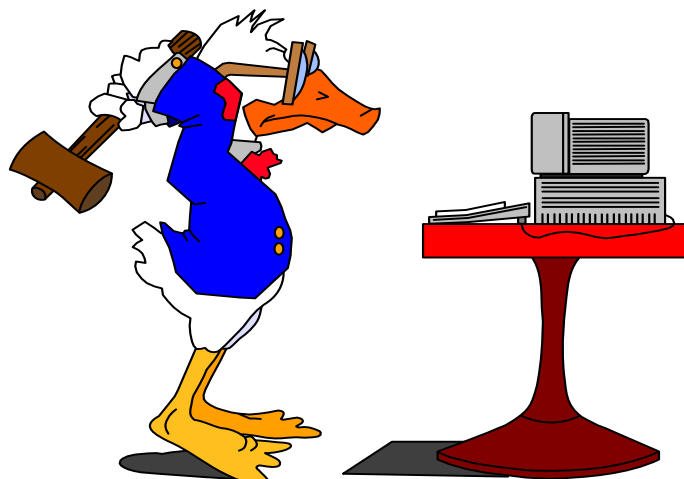
SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_13D.53Information and Training Center
Knowledge for Automation**Cooperation**

- Block programming in the SCL high-level language
- Common data management and code generation with CFC
- Direct cross access from SFC to CFC block instances
- Integration in the STEP 7 SIMATIC Manager

Решения упражнений



SIMATIC S7

Siemens AG 1999. All rights reserved.

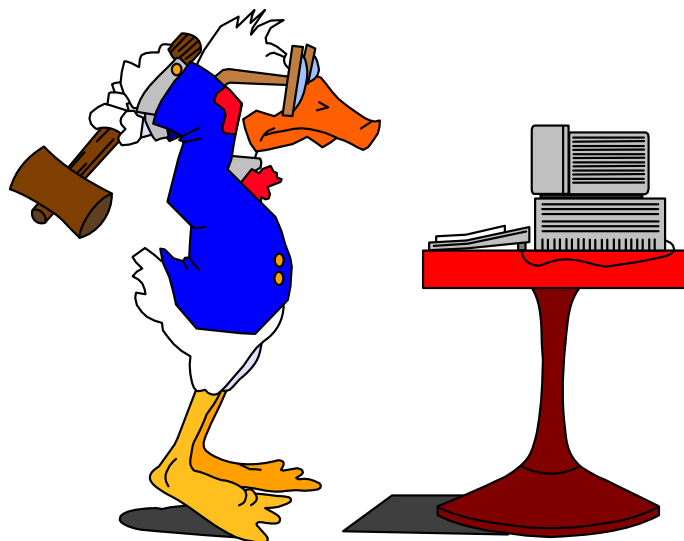
Date: 04.11.2005
File: PRO2_01E.1Information and Training Center
Knowledge for Automation

Содержание

Стр

Тренажер с S7-300	3
Конфигурация учебного модуля S7-300	4
Конфигурация учебного модуля S7-400	5
Симулятор	6
Модель конвейера	7
Решение упражнения 1.1: Переход после вычитания.....	8
Решение упражнения 1.2: Переход после умножения	9
Решение упражнения 1.3: Программирование распределенного перехода	10
Решение упражнения 2.1: Вычисление степени	11
Решение упражнения 2.2: Обмен данных в ACCU1	12
Решение упражнения 2.3 : Формирование дополнения	13
Решение упражнения 3.1: Вычисление расстояния	14
Решение упражнения 4.1: Программирование цикла с косвенной адресацией (часть 1)	15
Решение упражнения 4.2: Программирование цикла с регистровой косвенной адресацией	17
Решение упражнения 4.3: Функция для вычисления суммы и среднего значения	18
Решение упражнения 5.2: Доступ к сложным типам данных	19
Решение упражнения 5.3: Чтение системного времени с помощью SFC 1 (READ_CLK)	20
Решение упражнения 6.1a: Установка розлива - операционный режим	21
Решение упражнения 6.1b: Установка розлива -управление конвейером	22
Решение упражнения 6.2a: FB1 для рабочей станции	26
Решение упражнения 6.2a: FB2 для транспортера	28
Решение упражнения 6.2a: OB1	30
Решение упражнения 6.2b: Расширение до 3-х станций	31

Решения упражнений



SIMATIC S7

Siemens AG 1999. All rights reserved.

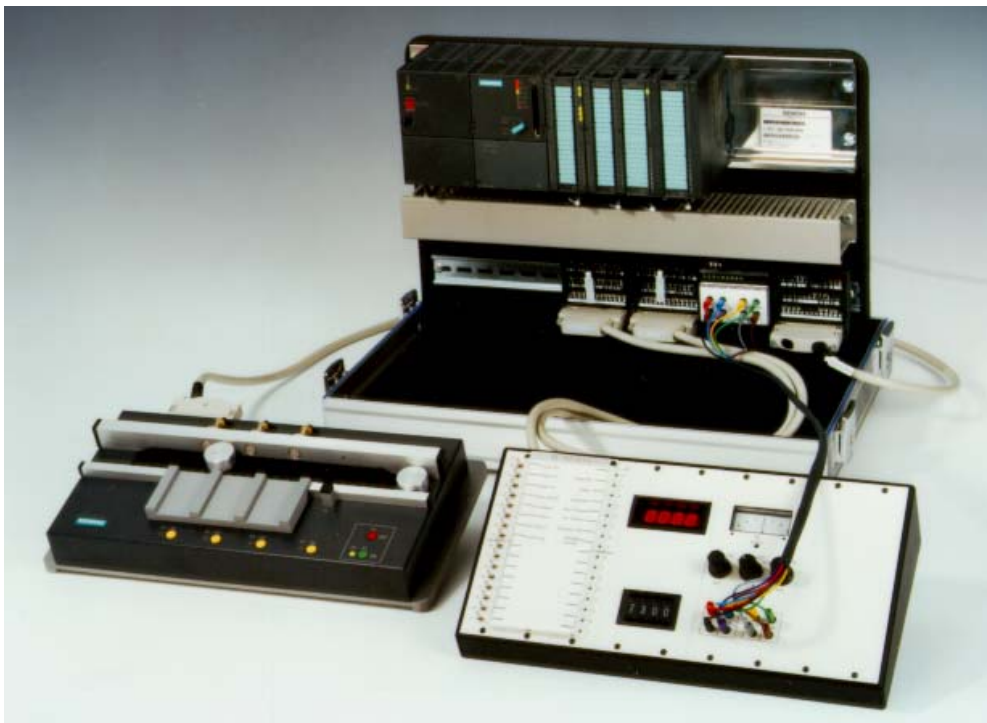
Date: 04.11.2005
File: PRO2_01E.2Information and Training Center
Knowledge for Automation

Содержание

Стр

Решение упражнения 7.2: Тестирование блока данных	35
Решение упражнения 7.3: Создание DB	36
Решение упражнения 7.4: Копирование DB из загрузочной в рабочую память	37
Решение упражнения 7.5: Инициализация DB	38
Решение упражнения 7.6: Запись сообщения в диагностический буфер (SFC 52)	39
Решение упражнения 7.7: Счетчик	40
Решение упражнения 8.1: Обработка ошибок в FC43	41
Решение упражнения 9.2: Подсчет готовых деталей	43
Решение упражнения 10.2: Коммуникации с SFB PUT/GET	49
Решение упражнения 10.3: Коммуникации с SFB START/STOP	51

Тренажер с S7-300



SIMATIC S7

Siemens AG 1999. All rights reserved.

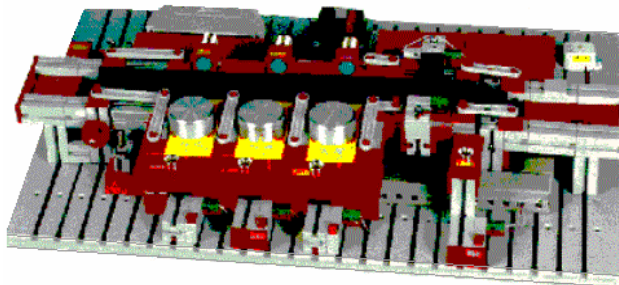
Date: 04.11.2005
File: PRO2_01E.3Information and Training Center
Knowledge for Automation**Содержание
обучающего
комплекта**

Обучающийся комплект состоит из следующих компонентов:

- S7-300 программируемый логический диспетчер с CPU 314 или CPU 315-DP
- Цифровые модули входов и выходов, аналоговые модули
- Тренажер с цифровыми и аналоговыми секциями
- Модель конвейера

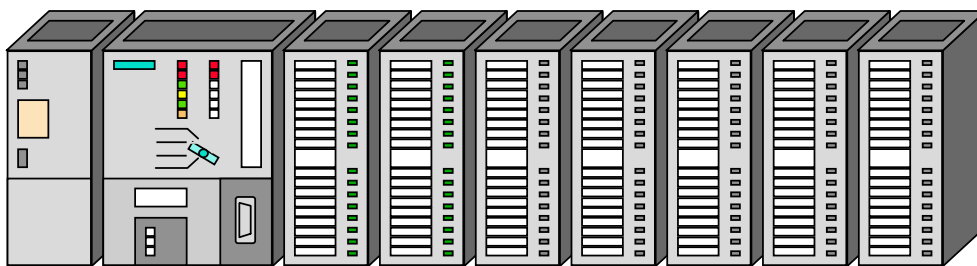
Обратить внимание:

Весьма возможно, что ваша учебная установка не оборудована конвейером



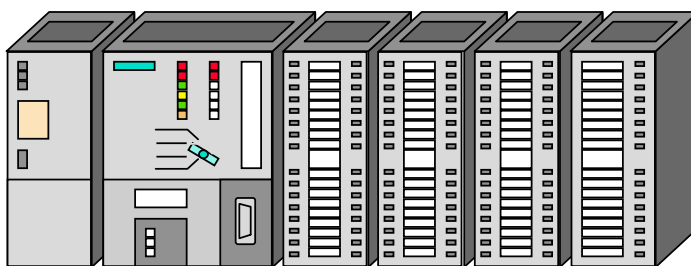
Конфигурация учебного модуля S7-300

Версия А (16 канальные I/O-модули)



Модуль	-->	PS	CPU	DI 16	DI 16	DO 16	DO 16	DI 16	DO 16	AI/AO4
№ слота	-->	1	2	4	5	6	7	8	9	10
Адрес I/O	-->			0	4	8	12	16	20	352

Версия В (32 канальные I/O-модули)



Модуль	-->	PS	CPU	DI 32	DO 32	DI8/DO8	AI 2
№ слота	-->	1	2	4	5	6	7
Адрес I/O	-->			0	4	8	304

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.4Information and Training Center
Knowledge for Automation

Конфигурация версии А

PLC состоит из следующих модулей:

Слот 1:	Источник питания 24V/5A	
Слот 2:	CPU 314 или CPU 315-2 DP	
Слот 4:	Цифр. входы 16x24V	Входы на тренажере
Слот 5:	Цифр. входы 16x24V	Кнопки
Слот 6:	Цифр. выходы 16x24V 0.5A	Выходы на тренажере
Слот 7:	Цифр. выходы 16x24V 0.5A	Цифровой дисплей
Слот 8:	Цифр. входы 16x24V	Входы модели конвейера
Слот 9:	Цифр. выходы 16x24V 0.5A	Выходы модели конвейера
Слот 10:	Аналог. модуль 4 AI/4 AO	Приспособление на тренажере

Конфигурация версии В

PLC состоит из следующих модулей:

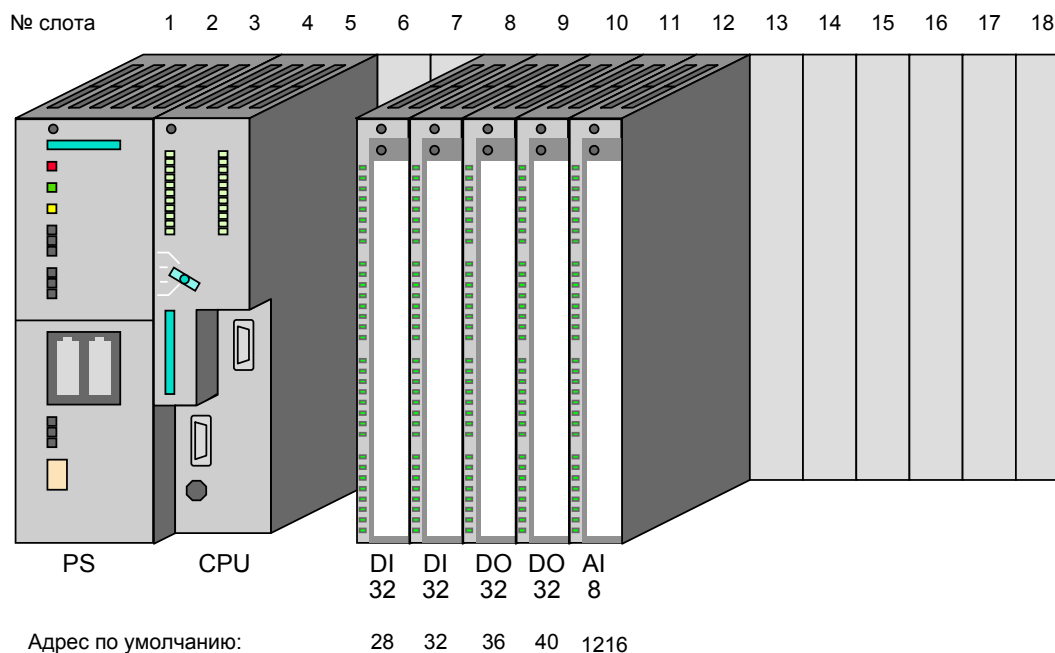
Слот 1:	Источник питания 24V/5A	
Слот 2:	CPU 314 или CPU 315-2 DP	
Слот 4:	Цифр. входы 32x24V	Входы на тренажере и кнопки
Слот 5:	Цифр. выходы 32x24V/0.5A	Выходы на тренажере и цифровой дисплей
Слот 6:	Цифр. входы и выходы 8x24V/ 8x24V 0.5A	Модель конвейера
Слот 7:	Аналог. входы 2 AI	Аналоговая секция на тренажере

Адресация

Фиксированная адресация для S7-300 (CPU 312-314). Адреса модулей показываются в слайде.

Стартовые адреса модулей могут быть изменены на CPU 315-2DP и для S7-400.

Конфигурация учебного модуля S7-400



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.5



Information and Training Center
Knowledge for Automation

Проект Конфигурация

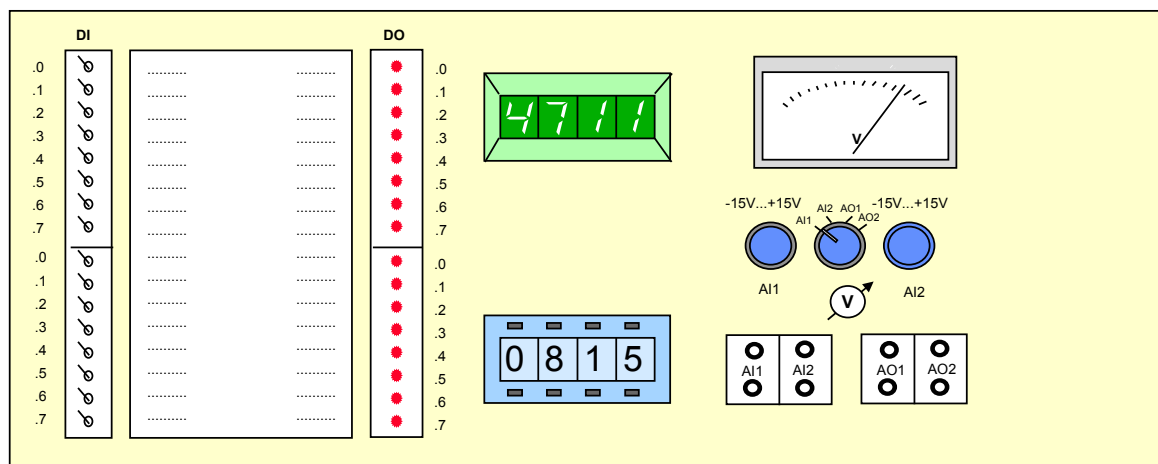
Вы можете видеть проект учебного тренажера S7-400 в слайде выше.
Монтажная стойка UR 1 - конфигурирована следующими модулями:

Слот 1:	Источн. питания 24V и 5V/20A	
Слот 2:	- " -	
Слот 3:	- " -	
Слот 4:	CPU 412 или др.	
Слот 5:	свободно (когда CPU имеет одиночную ширину)	
Слот 6:	свободно	
Слот 7:	свободно	
Слот 8:	Цифр. входы 32x24V	(от симулятора)
Слот 9:	Цифр. входы 32x24V	(от модели конвейера)
Слот 10:	Цифр. выходы 32x24V 0.5A	(к симулятору)
Слот 11:	Цифр. выходы 32x24V 0.5A	(к модели конвейера)
Слот 12:	Аналог. входы 8X13 Bit	(от потенциометра симулятора)
Слот 13:	свободно	
Слот 14:	свободно	
Слот 15:	свободно	
Слот 16:	свободно	
Слот 17:	свободно	
Слот 18:	свободно	

Адресация

Вы имеете адресацию по умолчанию, как показано в слайде выше, пока не выполнено никакой другой конфигурации или назначения параметров.

Симулятор



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.6Information and Training Center
Knowledge for Automation

Проект

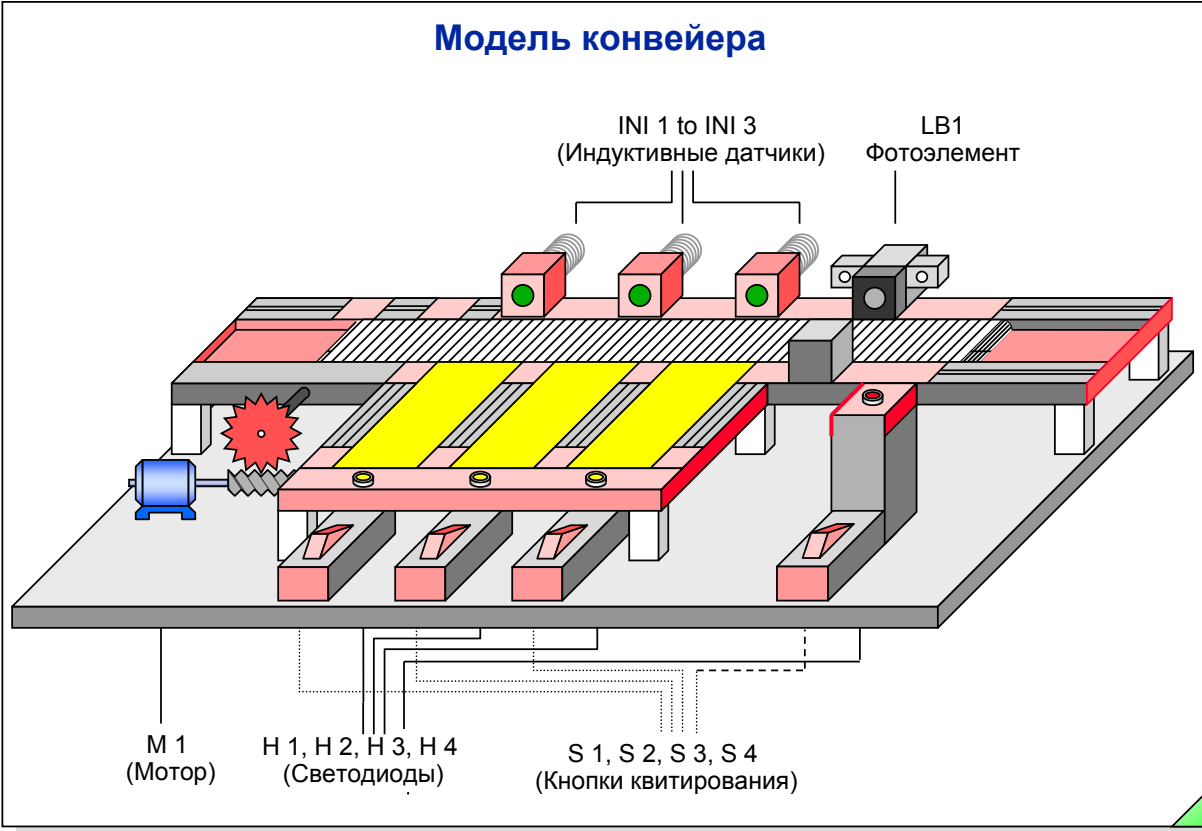
Тренажер связан с S7-300 или S7-400 двумя кабелями. Он состоит из трех секций:

- Двоичная секция с 16 переключателями и 16 светодиодами
- Цифровая секция с 4 кнопками и цифровым дисплеем. Они работают с BCD-значениями.
- Аналоговая секция с вольтметром для показа значений аналоговых каналов № 0 и 1 или аналоговых выходов № 0 и 1. Вы используете переключатель, чтобы выбрать значение, которое Вы хотите контролировать. Имеются два отдельных потенциометра для установки значений для аналоговых входов.

Адресация

Вы используете следующие адреса, чтобы адресовать входы и выходы в вашей программе:

Датчик / Привод	Версия A (DI16, DQ16)	Версия B (DI32, DQ32)	S7-400 (Адреса по умолчанию)
Переключатели	IW 0	IW 0	IW 28
Светодиоды	QW 8	QW 4	QW 36
Кнопки	IW 4	IW 2	IW 30
Цифровой диспл.	QW 12	QW 6	QW 38
Аналог. каналы	PIW 352/354	PIW 304/306	PIW 1216/1230



SIMATIC S7
Siemens AG 1999. All rights reserved.

Date: 04.11.2005
File: PRO2_01E.7



Information and Training Center
Knowledge for Automation

Проект Слайд показывает диаграмму модели конвейера с датчиками и приводами.

Адресация

S7-300 Вер. A (DI16, DO16)	S7-300 Вер. B (DI32, DO32)	S7-400 (w/o HW Config)	Датчик/Привод	Символ
I 16.0	I 8.0	I 32.0	Фотоэлемент LB 1	LB1
I 16.1	I 8.1	I 32.1	Квит. ключ, место 1	S1
I 16.2	I 8.2	I 32.2	Квит. ключ, место 2	S2
I 16.3	I 8.3	I 32.3	Квит. ключ, место 3	S3
I 16.4	I 8.4	I 32.4	Квит. ключ, место, финиш	S4
I 16.5	I 8.5	I 32.5	Индукт. датчик 1	INI1
I 16.6	I 8.6	I 32.6	Индукт. датчик 2	INI2
I 16.7	I 8.7	I 32.7	Индукт. датчик 3	INI3
Q 20.1	Q 8.1	Q 40.1	Светодиод, место 1	H1
Q 20.2	Q 8.2	Q 40.2	Светодиод, место 2	H2
Q 20.3	Q 8.3	Q 40.3	Светодиод, место 3	H3
Q 20.4	Q 8.4	Q 40.4	Светодиод, фин. сборка	H4
Q 20.5	Q 8.5	Q 40.5	Конвейер вправо	K1_CONVR
Q 20.6	Q 8.6	Q 40.6	Конвейер влево	K2_CONVL
Q 20.7	Q 8.7	Q 40.7	Сирена	HORN

Решение упражнения 1.1: Переход после вычитания

```
FUNCTION FC 11 : VOID
TITLE =Exercise 1.1 : Jump After a Substraction
//Version for 16Bit SM
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

BEGIN
NETWORK
TITLE =

L    IW  4;           // Переключатель
BTD ;                // Преобразование из BCD в DINT
L    IW  0;           // Входное слово 0
BTD;
-D;
JM   NEG;             // Переход, если результат отрицательный
L    IW  4;
JU   END;
NEG: L    0;
END: T    QW 12;       // Цифровой дисплей
END_FUNCTION
```

Решение упражнения 1.2: Переход после умножения

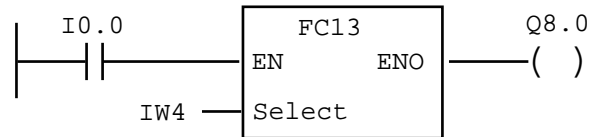
```
FUNCTION FC 12 : VOID
TITLE =Exercise 1.2 : Jump After a Multiplication
//Version for 16Bit SM
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

BEGIN
NETWORK
TITLE =

L    IW  4;           // Переключатель
BTD;                  // Преобразование из BCD в DINT
L    IW  0;           // Переключатель на симуляторе
BTD;
*I;
JO   OVL;             // Переход, если переполнение
DTB;                  // Преобразование из DINT в BCD
JU   END;
OVL: L    0;
END: T    QW 12;       // Цифровой дисплей
END_FUNCTION
```

Решение упражнения 1.3: Программирование распределенного перехода

ОВ1



```

FUNCTION FC 13: VOID
//Version for 16Bit SM
// Programming a Jump Distributor
VAR_INPUT
Select: INT;
END_VAR
BEGIN
SET;
L   #Select;
AW   W#16#FF00; // Проверка: выбор > 255 или
JN   Err;       // переход, если > 255
L   #Select;    // Загрузить значение снова
JL   GT5;       // Переход, если ACCU1-L-L >5
JU   Err;       // если выбор = 0 (not possible)
JU   Dr_1;      // Лента вправо (выбор n=1)
JU   Dr_2;      // Лента влево (выбор =2)
JU   Dr_3;      // Стоп (выбор =3)
JU   Ho_1;      // Сирена вкл.
JU   Ho_2;      // Сирена выкл.
GT5:
JU   Err;
Dr_1:
S   Q   20.5;    // Лента вправо
R   Q   20.6;
JU   End;
Dr_2:
S   Q   20.6;    // Лента влево
R   Q   20.5;
JU   End;
Dr_3:
R   Q   20.5;    // Стоп
R   Q   20.6;
JU   End;
Ho_1:
S   Q   20.7;    // Сирена вкл.
JU   End;
Ho_2:
R   Q   20.7;    // Сирена выкл.
JU   End;
Err:
R   Q   20.5;    // Стоп
R   Q   20.6;
R   Q   20.7;    // Сирена выкл.
CLR;             // Сброс ENO
SAVE;
End:
BE;
END_FUNCTION
  
```


Решение упражнения 2.1: Вычисление степени

```
FUNCTION FC 21 : VOID
TITLE =Exercise 2.1: Calculation of Exponents
//Version for 16Bit SM
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

BEGIN
NETWORK
TITLE =
L   IB   5;           // Загрузите правый байт кнопочного переключателя
BTI;                 // BCD в INT -> входн. значение
  PUSH;              // Копирование ACCU1 в ACCU2
*D;                  // Формирование квадрата из вх. знач. в ACCU1
  PUSH;              // Копирование квадрата вх. знач. из ACCU1 в ACCU2
  PUSH;              // Только для S7-400: квадрат - > ACCU3
*D;                  // Формирование 4-й степени вх. знач. в ACCU1
*D;                  // Формирование 6-й степени вх. знач. в ACCU1
  DTB;               // Преобразование в BCD
T   QW 12;           // Передача мл. слова на цифровой дисплей
END_FUNCTION
```

Решение упражнения 2.2: Обмен данных в ACCU1

```
FUNCTION FC 22 : VOID
TITLE =Exercise 2.2: Data Exchange in ACCU1
//Version for 16Bit SM
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

BEGIN
NETWORK
TITLE =

L    IW  4;           // Загрузка BCD-числа
CAW;                  // Замена двух байтов в ACCU1-L
T    QW 12;          // Показать результат
END_FUNCTION
```

Решение упражнения 2.3 : Формирование дополнения

```
FUNCTION FC 23 : VOID
TITLE = 2.3: Forming Complements
//Version for 16Bit SM
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

BEGIN
NETWORK
TITLE =Ones Complement in STL

L    IW  0;           // Загрузка входного слова с переключателя
INVI;                 // Формирование дополнения
T    QW  8;           // Перенос результата на светодиоды симулятора
END_FUNCTION
```

Решение упражнения 3.1: Вычисление расстояния

```
FUNCTION FC 31 : REAL
TITLE =Exercise 3.1: Calculating the Distance
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

VAR_INPUT
X1: REAL;
Y1: REAL;
X2: REAL;
Y2: REAL ;
END_VAR
VAR_TEMP
XSquare : REAL;
END_VAR
BEGIN
NETWORK
TITLE =
L   #X1;           // Загрузка координаты X точки P1
L   #X2;           // Загрузка координаты X точки P2
-R;                // Вычисление (X1-X2)
SQR;               // Формирование квадрата (X1-X2)
T   #XSquare;      // сохранение рез-та во врем. переменной
L   #Y1;           // Загрузка координаты Y точки P1
L   #Y2;           // Загрузка координаты Y точки P2
-R;                // Вычисление (Y1-Y2)
SQR;               // Формирование квадрата (Y1-Y2)
L   #XSquare;      // Восстановление квадрата (X1-X2)
+R;                // Формирование суммы
SQRT;              // Вычисление квадр. корня
T   #RET_VAL;      // Перенос в RET_VAL
END_FUNCTION
```

Решение дополнительного упражнения (см. упражнение 3.1)

```
FUNCTION FC 1 : REAL
TITLE = Exercise 3.1: Calculating the Distance Additional Task
AUTHOR : Sokolov
VERSION : 0.1

VAR_INPUT
X1 : REAL ;
Y1 : REAL ;
X2 : REAL ;
Y2 : REAL ;
END_VAR
BEGIN
NETWORK
TITLE =
L   #X1;
L   #X2;
-R ;
PUSH ;
SQR ;
PUSH ;
PUSH ;
L   #Y1;
L   #Y2;
-R ;
SQR ;
+R ;
SQRT ;
T   #RET_VAL;
END_FUNCTION
```

Решение упражнения 4.1: Программирование цикла с косвенной адресацией (часть 1)

```
FUNCTION FC 41 : VOID
TITLE =Exercise 4.1:Loop Programming with Memory Indirect Adressing

VAR_INPUT
    DB_Num : WORD ;
END_VAR
VAR_TEMP
    L_Counter : INT ;
    Ini_Value : REAL ;
    I_DB_Num : WORD;
    Par_Pointer : DWORD ;END_VAR
BEGIN
NETWORK
TITLE =Open the DB

    L   #DB_Num;           // Загрузка номера DB
    T   #I_DB_Num;         // и перенос его во внутреннюю переменную;
    OPN DB [#I_DB_Num];    // открыть DB

NETWORK
TITLE =LOOP

    L   P#0.0;             // Загрузка адреса 1-го компонента
    T   #Par_Pointer;       // и перенос в #T_Pointer
    L   1.0;               // Загрузка константы 1.0 и
    T   #Ini_Value;         // и перенос в #Ini_Value
    L   100;               // Инициализация счетчика цикла значением 100
    BEGN: T   #L_Counter;   // и перенос в #L_Counter
    L   #Ini_Value;
    T   DBD [#Par_Pointer]; // Перенос #Ini_Value в Meas_Value[i]
    L   1.0;               // Инкрементирование ACCU1 (#Ini_Value)
    +R   ;                 // на 1.0
    T   #Ini_Value;         // и перенос в #Ini_Value
    L   #Par_Pointer;       // Загрузка #Par_Pointer в ACCU1
    L   P#4.0;             // Инкрементирование адреса
    +D   ;                 // в #Par_Pointer на 4 байта
    T   #Par_Pointer;       // и перенос результата в #Par_Pointer
    L   #L_Counter;         // Загрузка счетчика цикла,
    LOOP BEGN;             // Декрементирование счетчика цикла и, если необходимо,
                           // переход

END_FUNCTION
```

Решение упражнения 4.1: Программирование цикла с косвенной адресацией (часть 2)

```
ORGANIZATION_BLOCK OB 1
TITLE =
VERSION : 0.1
VAR_TEMP
  OB1_EV_CLASS : BYTE ;      //Биты 0-3 = 1 (Приход. соб.), Bits 4-7 = 1 (Класс события 1)
  OB1_SCAN_1 : BYTE ;        //1 (Холодный рестарт 1ый скан OB 1), 3 ( 2-ой скан OB 1)
  OB1_PRIORITY : BYTE ;      //1 (Класс приоритета)
  OB1_OB_NUMBR : BYTE ;      //1 (Номер OB, OB1)
  OB1_RESERVED_1 : BYTE ;    //Резерв
  OB1_RESERVED_2 : BYTE ;    //Резерв
  OB1_PREV_CYCLE : INT ;     //Время цикла предыд. скана OB1 (ms)
  OB1_MIN_CYCLE : INT ;      //Мин. время цикла OB1 (ms)
  OB1_MAX_CYCLE : INT ;      //Макс. время цикла OB1 (ms)
  OB1_DATE_TIME : DATE_AND_TIME ; //Дата и время старта OB1
END_VAR
BEGIN
NETWORK
TITLE =
  CALL FC 41 (
    DB_Num := W#16#29);
NOP 0;

END_ORGANIZATION_BLOCK
```

Решение упражнения 4.2: Программирование цикла с регистровой косвенной адресацией

```
FUNCTION FC 42: VOID
TITLE =Exercise 4.2: Loop Programming with Register Indirect Addressing
// Version for S7-300 and S7-400

VAR_INPUT
    DB_Num : WORD ;
END_VAR
VAR_TEMP
    I_DB_Num : WORD ;
END_VAR
BEGIN
NETWORK
TITLE =Open the DB

    L    #DB_Num;           // Загрузка номера DB
    T    #I_DB_Num;         // перенос во временную переменную;
    OPN  DB [#I_DB_Num];    // Открыть DB

NETWORK
TITLE =LOOP

    LAR1 P#DBX0.0;          // Загрузка адреса 1-го компонента
    L    L#1;               // 1 в ACCU1(Ini_Value.)
    L    100;               // 100 в AKKU1 (L_Counter); 1 в ACCU2 (Ini_Value)
    BEGN: TAK ;             // L_Counter в ACCU2, Ini_Value в ACCU1
    T    D [AR1,P#0.0];     // Перенос Ini_Value в Tank[i]
    +    L#1;               // Инкрементирование Ini_Value
    +AR1 P#4.0;             // Инкрементирование AR1 на 2 байта
    TAK ;                   // L_Counter в ACCU1, Ini_Value в ACCU2
    LOOP BEGN;              // Декрементирование и переход

END_FUNCTION
```

Решение упражнения 4.3: Функция для вычисления суммы и среднего значения

```

FUNCTION FC 43 : VOID
TITLE = Exercise 4.3: Calculating Sum and Mean Value
VERSION : 0.0
VAR_INPUT
    Measured_values : ANY ;
END_VAR
VAR_OUTPUT
    Sum : REAL ;
    Mean_value : REAL ;
END_VAR
VAR_TEMP
    Num_Elements : WORD ;
    L_Counter : WORD ;
    DB_No : WORD ;
END_VAR
BEGIN
NETWORK
TITLE =
    L   P##Measured_values; // Загрузка адреса указателя "ANY"
    LAR1 ;                  // Перенос адреса в AR1
    L   B [AR1,P#1.0];      // Загрузка идентификатора типа данных
    L   8;                  // Загрузка идентификатора REAL (16#08)
    ==I ;
    JC  REAL;               // Переход, если тип данных REAL
    NOP 0;                  // Инструкции для типа данных, отличного от REAL
    CLR ;                   // RLO=0
    SAVE ;                  // BR=0
    L   L#-1;               // Загрузка не-REAL числа
    T   #Sum;
    T   #Mean_value;
    BEU ;
REAL: NOP 0;               // Инструкции для типа данных : REAL
    L   W [AR1,P#2.0];      // Загрузить число элементов массива
    T   #Num_Elements;      // Сохранить число элементов
    L   W [AR1,P#4.0];      // Загрузить номер DB или 0
    T   #DB_No;             // Если № DB =0, то: OP DB[DB_No]=NOP
    OPN DB [#DB_No];       // Ошибка времени выполнения!!, если DB не существует
    L   D [AR1,P#6.0];      // Загрузка указателя на вктуальный операнд
    LAR1 ;                  // в AR, ошибка !! Если область операнда не совпадает с "DI"
    L   0.000000e+000;      // 0 в ACCU1 (сумма =0.0)
    L   #Num_Elements;      // Num_Elements в ACCU1; сумма =0 в ACCU2
BEGN: T   #L_Counter;      // Установить L_Counter
    TAK ;                   // сумма в ACCU1
    L   D [AR1,P#0.0];      // Элемент массива в ACCU1, сумма в ACCU2
    +R ;                     // Сумма в ACCU1
    +AR1 P#4.0;             // Инкрементирование AR1 на 4 байта
    L   #L_Counter;        // L_Counter в ACCU1, сумма в ACCU2
    LOOP BEGN;              // Декрементирование и переход
    TAK ;                   // Сумма в ACCU1
    T   #Sum;               // Сумма в #Sum
    L   #Num_Elements;      // Сумма в ACCU2, число элементов в ACCU1
    DTR ;                   // Преобразовать беззнаковое целое (16 бит) в REAL
    /R ;                     // Среднее значение в ACCU1
    T   #Mean_value;        // Перенос среднего значения в #Mean_value
    SET ;                   // Установить BR-бит
    SAVE ;
END_FUNCTION

```


Решение упражнения 5.2: Доступ к сложным типам данных

```
FUNCTION FC 52 : VOID
TITLE =Monitoring Motors
//Version for S7-300 and S7-400
VERSION : 0.1

VAR_INPUT
    Motor : "Motor";
END_VAR
VAR_OUTPUT
    Motor_OK : BOOL ;
    SetActDiff : DINT ;
    SetActDiffDisp : DWORD ;
END_VAR
VAR_TEMP
    SetActDifference : REAL ;
END_VAR
BEGIN
NETWORK
TITLE =

//Вычисление процента отклонения
    SET ; //Сначала проверка, установить RLO в "1"
    SAVE ; //Установить BR-бит в "1"
    L #Motor.SetSpeed; //Установить скорость в ACCU1
    PUSH ; //только для S7-400, установить скорость в ACCU2
    PUSH ; //Установить скорость в ACCU3
    L #Motor.ActualSpeed; // Установить скорость в ACCU2, актуальную скорость в
    //ACCU1
    -R ; //Разность в ACCU1, установить скорость в ACCU2
    T #SetActDifference; //Сохранить разность во врем. переменной
    TAK ; // Разность в ACCU2, установить скорость в ACCU1
    /R ; //Значение отклонения в процентах в ACCU1
    ABS ; //Абсолютное значение отклонения ACCU1
    L #Motor.SetActDiffMax; //Загрузить max. отклон. в процентах ACCU1
    <=R ; //Актуальное отклонение равно или меньше заданного?
    AN #Motor.Disturbance; //и нет вибраций
    = #Motor_OK; //то мотор OK
NETWORK
TITLE =Показ разности и между заданной скоростью и фактической скоростью

    L #SetActDifference; //промежуточная загрузка в SetActDifference
    RND ; //преобразование в DINT
    PUSH ; //сохранение SetActDifference в ACCU2
    DTB ; //число DINT в ACCU2, число BCD в ACCU1
    JO ERR; //переход если ошибка преобразования
    T #SetActDiffDisp; //передача правильного BCD-числа на цифровой дисплей
    TAK ;
    T #SetActDiff; // передача правильного DINT числа в #SetActDiff
    BEU ; //если ошибка - закончить функцию
ERR: CLR ;
    SAVE ; //очистить BR-бит
END_FUNCTION
```

Решение упражнения 5.3: Чтение системного времени с помощью SFC 1 (READ_CLK)

```
FUNCTION FC 53 : VOID
TITLE = Exercise 5.3: Read System Clock
//Version for 16Bit-SM
AUTHOR : PT41
FAMILY : A2_0
NAME : ST7PRO2
VERSION : 0.0

VAR_TEMP
  Date_Time : DATE_AND_TIME ;      //Текущая дата и время
  RET_VAL_SFC1 : INT ;             //RET_VAL SFC 1
END_VAR
BEGIN
NETWORK
TITLE =Call SFC 1 (READ_CLK)

  CALL SFC1 (
    RET_VAL      := #RET_VAL_SFC1,
    CDT          := #Date_Time);

  NOP 0;
NETWORK
TITLE = Display hours and minutes

  LAR1 P##Date_Time;               // Взять адрес #Date_Time
  L   LB [AR1, P#3.0];              // Загрузить часы
  T   QB   12;                      // и перенести на цифровой дисплей
  L   LB [AR1, P#4.0];              // Прочитать минуты
  T   QB   13;                      // и перенести на цифровой дисплей

END_FUNCTION
```

Решение упражнения 6.1а: Установка розлива - операционный режим

```
FUNCTION_BLOCK "Mode_Selection"  
TITLE =Mode_Selection  
VERSION : 0.1
```

```
VAR_INPUT  
    Start : BOOL ;  
    Stop : BOOL ;  
    Auto_Man : BOOL ;  
    OM_activate : BOOL ;  
END_VAR  
VAR_OUTPUT  
    Plant_on : BOOL ;  
    OM_Man : BOOL ;  
    OM_Auto : BOOL ;  
END_VAR
```

```
BEGIN
```

```
NETWORK
```

```
TITLE =Plant on/off
```

```
    A   #Start;           // если установка включена,  
    S   #Plant_on;       // установить выход plant_on;  
    AN  #Stop;           // если установка выключена,  
    R   #Plant_on;       // сбросить выход plant_on;  
    A   #Plant_on;       //  
    =   #Plant_on;       //
```

```
NETWORK
```

```
TITLE =OM: Manual
```

```
    A   #Plant_on;       // если установка включена,  
    AN  #Auto_Man;       // если выбран ручной режим,  
    A   #OM_activate;    // если вход enter_mode активен,  
    S   #OM_Man;         // установить выход ручного режима;  
    A(   ;  
    ON  #Plant_on;       // если установка выключена,  
    O   ;               // или  
    A   #Auto_Man;       // выбран автоматический режим  
    A   #OM_activate;    // и enter_mode активен,  
    )   ;  
    R   #OM_Man;         // сбросить выход manual_mode;  
    A   #OM_Man;         //  
    =   #OM_Man;         //
```

```
NETWORK
```

```
TITLE =OM: Automatic
```

```
    A   #Plant_on;       // если установка включена,  
    A   #Auto_Man;       // если выбран автоматический режим,  
    A   #OM_activate;    // если вход enter_mode активен,  
    S   #OM_Auto;        // установить выход automatic_mode;  
    A(   ;  
    ON  #Plant_on;       // если установка выключена,  
    O   ;               // или  
    AN  #Auto_Man;       // если выбран ручной режим  
    A   #OM_activate;    // и enter_mode активен,  
    )   ;  
    R   #OM_Auto;        // сбросить выход automatic_mode;  
    A   #OM_Auto;        //  
    =   #OM_Auto;        //
```

```
END_FUNCTION_BLOCK
```

Решение упражнения 6.1b: Установка розлива -управление конвейером (часть 1)

```
FUNCTION_BLOCK "Conveyor_Control"
TITLE =
VERSION : 0.1

VAR_INPUT
    OM_Man : BOOL ;
    OM_Auto : BOOL ;
    Jog_for : BOOL ;
    Jog_back : BOOL ;
    Sensor_fill : BOOL ;
    Sensor_full : BOOL ;
END_VAR
VAR_OUTPUT
    Conv_for : BOOL ;
    Conv_back : BOOL ;
    Filling_active : BOOL ;
    Full_bottles : WORD ;
END_VAR
VAR
    Filling_time : "TP";
    Bottle_counter : "CTU";
END_VAR
VAR_TEMP
    bottles : INT ;
END_VAR

BEGIN
NETWORK
TITLE =Branch between Manual and Automatic Mode
    SET ; // проверка,
    SAVE ; // и установить BR-бит в "1";
    A #OM_Man; // если manual_mode активен,
    JC Man; // переход на ручной режим;
    A #OM_Auto; // если automatic_mode активен,
    JC Auto; // переход на автоматический режим;
    R #Conv_for; // если OM активен,
    R #Conv_back; // остановить ленту,
    R #Filling_active; // сбросить filling_active
    CALL #Bottle_counter (
        R := TRUE); // сбросить счетчик

    L 0; // сбросить full_bottles
    T #Full_bottles;
    BEU ;

NETWORK
TITLE =OM_Man
//Управление конвейером через ключ JOG
Man: A #Jog_for;
    AN #Jog_back;
    = #Conv_for;
    A #Jog_back;
    AN #Jog_for;
    = #Conv_back;
    BEU ;

// (Продолжение на следующей странице)
```

Решение упражнения 6.1b: Установка розлива - управление конвейером (часть 2)

```
NETWORK
TITLE =OM_Auto
//Старт Filling_time
Auto: A  #Sensor_fill;
    = L 2.0;
    BLD 103;
    CALL #Filling_time (
        IN      := L 2.0,
        PT      := T#3S,
        Q        := #Filling_active);

    NOP 0;
```

```
NETWORK
TITLE =OM_Auto
//Подсчет полных бутылок
A  #Sensor_full;
    = L 2.0;
    BLD 103;
    CALL #Bottle_counter (
        CU      := L 2.0,
        R        := FALSE,
        CV        := #bottles);
    NOP 0;
```

```
NETWORK
TITLE =OM_Auto
//Преобразование #bottles в BCD
//
    L  #bottles;
    ITB ;
    T  #Full_bottles;
    NOP 0;
```

```
NETWORK
TITLE =OM_Auto
//Conveyor_forward вкл., пока идет заполнение
    AN #Filling_active;
    =  #Conv_for;
END_FUNCTION_BLOCK
```

Решение упражнения 6.1b: Установка розлива - управление конвейером (часть 3)

ORGANIZATION_BLOCK "Cycle"

TITLE =

VERSION : 0.1

VAR_TEMP

OB1_EV_CLASS : BYTE ; //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)

OB1_SCAN_1 : BYTE ; //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)

OB1_PRIORITY : BYTE ; //1 (Priority of 1 is lowest)

OB1_OB_NUMBR : BYTE ; //1 (Organization block 1, OB1)

OB1_RESERVED_1 : BYTE ; //Reserved for system

OB1_RESERVED_2 : BYTE ; //Reserved for system

OB1_PREV_CYCLE : INT ; //Cycle time of previous OB1 scan (milliseconds)

OB1_MIN_CYCLE : INT ; //Minimum cycle time of OB1 (milliseconds)

OB1_MAX_CYCLE : INT ; //Maximum cycle time of OB1 (milliseconds)

OB1_DATE_TIME : DATE_AND_TIME ; //Date and time OB1 started

Full_bottles : INT ;

END_VAR

BEGIN

NETWORK

TITLE =Operating mode

//Управление режимом работы

A "Start";

= L 22.0;

BLD 103;

A "Stop";

= L 22.1;

BLD 103;

A "Man/Auto";

= L 22.2;

BLD 103;

A "Enter_Mode";

= L 22.3;

BLD 103;

CALL "Mode_selection", "Mode_Selection_DB" (

Start := L 22.0,

Stop := L 22.1,

Auto_Man := L 22.2,

OM_activate := L 22.3,

Plant_on := "Plant_on",

OM_Man := "Manual_Mode",

OM_Auto := "Automatic_Mode");

NOP 0;

// (Продолжение на следующей странице)

Решение упражнения 6.1b: Установка розлива - управление конвейером (часть 4)

NETWORK

TITLE =Управление лентой

```
A  "Manual_Mode";
=  L  22.0;
BLD 103;
A  "Automatic_Mode";
=  L  22.1;
BLD 103;
A  "Jog_forward";
=  L  22.2;
BLD 103;
A  "Jog_backward";
=  L  22.3;
BLD 103;
A  "Filling_Position";
=  L  22.4;
BLD 103;
A  "Counting_Bottles";
=  L  22.5;
BLD 103;
CALL "Conveyor_Control" , "Conveyor_Control_DB" (
    OM_Man      := L  22.0,
    OM_Auto     := L  22.1,
    Jog_for     := L  22.2,
    Jog_back    := L  22.3,
    Sensor_fill := L  22.4,
    Sensor_full := L  22.5,
    Conv_for    := "Conveyor_forward",
    Conv_back   := "Conveyor_backward",
    Filling_active := "Filling_in_progress",
    Full_bottles := "Display");
NOP 0;
```

END_ORGANIZATION_BLOCK

Решение упражнения 6.2а: FB1 для рабочей станции (часть 1)

```
FUNCTION_BLOCK "Station"
TITLE =управление рабочей станцией
AUTHOR : PT41
FAMILY : A2_0
NAME : ST7PRO2
VERSION : 0.0

VAR_INPUT
    Initial : BOOL ;
    Proxy_switch : BOOL ;
    Acknowledge : BOOL ;
    Clock_bit_q : BOOL ;
    Clock_bit_s : BOOL ;
END_VAR
VAR_OUTPUT
    LED : BOOL ;
    Transp_req : BOOL ;
END_VAR
VAR_IN_OUT
    Conv_busy : BOOL ;
END_VAR
VAR
    State : STRUCT
        Process_piece : BOOL ;
        Piece_finished : BOOL ;
        Place_piece_on_conv : BOOL ;
        Wait_for_piece : BOOL ;
        Take_piece_from_conv : BOOL ;
    END_STRUCT ;
END_VAR
BEGIN
NETWORK
TITLE =Initialization
//Посредством входа"Initial" устанавливается основное состояние #Process_piece
    A    #Initial;
    S    #State.Process_piece;
    R    #State.Piece_finished;
    R    #State.Place_piece_on_conv;
    R    #State.Wait_for_piece;
    R    #State.Take_piece_from_conv;
    R    #Conv_busy;

NETWORK
TITLE =State: Process_piece
// В этом состоянии рабочая часть обработана. Обработка закончена,
// когда оператор подтверждает завершение рабочей части
// посредством кнопки "S1"
    AN   #State.Process_piece;
    JC   Pfin;
    S    #LED;           //светодиод горит постоянно ;
    R    #Transp_req;
    A    #Acknowledge;    //Когда оператор подтверждает,
    R    #State.Process_piece; //изменение состояния выполняется;
    R    #LED;
    S    #State.Piece_finished;

// (Продолжение на следующей странице)
```


Решение упражнения 6.2а: FB1 для рабочей станции (часть 2)

NETWORK

TITLE =State: Piece_finished

// В состоянии #Piece_finished оператор ждет разрешение

// разместить деталь на конвейере. Сигнал #Conv_busy указывает,

// занят конвейер или нет. Когда конвейер свободен, изменение

// состояния Place_piece_on_conv выполнено.

Pfin: AN #State.Piece_finished;

JC PpCo;

A #Clock_bit_s; //медленное мигание;

= #LED;

AN #Conv_busy; //если конвейер свободен,

S #Conv_busy; //он занимается

R #LED;

R #State.Piece_finished;

S #State.Place_piece_on_conv;

NETWORK

TITLE =State: Place_piece_on_conv

PpCo: AN #State.Place_piece_on_conv;

JC Wait;

A #Clock_bit_q; //быстрое мигание;

= #LED;

A #Proxy_switch; //Если деталь лежит на конвейере,

S #Transp_req; //то он стартует,

R #LED; //и светодиод очищен;

A #Transp_req; //Когда лента движется,

AN #Proxy_switch; //деталь покидает индуктивный датчик,

R #State.Place_piece_on_conv; // происходит изменение состояния;

S #State.Wait_for_piece;

NETWORK

TITLE =State: Wait_for_piece

// Ожидание новой необработанной детали. Достижение нового

// индуктивного датчика

Wait: AN #State.Wait_for_piece;

JC TpCo;

R #LED; //светодиод выкл.;

A #Proxy_switch; //прибывает новая необработанная деталь,

R #Transp_req; //конвейер останавливается,

R #State.Wait_for_piece; //и состояние изменяется;

S #State.Take_piece_from_conv;

NETWORK

TITLE =State: Take_piece_from_conv

// В этом состоянии новая необраб. деталь принята с конвейера на

// рабочее место

TpCo: AN #State.Take_piece_from_conv;

JC END;

A #Clock_bit_q; //светодиод мигает быстро

= #LED;

AN #Proxy_switch; //Когда необраб. деталь принята с транспортера ,

R #Conv_busy; //конвейер установив. в состояние "свободный",

R #LED; //светодиод выключен

R #State.Take_piece_from_conv; //и состояние изменяется;

S #State.Process_piece;

END: BEU ;

END_FUNCTION_BLOCK

Решение упражнения 6.2а: FB2 для транспортера (часть 1)

```
FUNCTION_BLOCK "Transport"
TITLE =Управление лентой транспортера
VERSION : 0.1
VAR_INPUT
    Initial : BOOL ;
    L_Barrier : BOOL ;
    Acknowledge : BOOL ;
    Transp_req : BOOL ;
    Clock_Bit : BOOL ;
END_VAR
VAR_OUTPUT
    LED : BOOL ;
    Conv_right : BOOL ;
    Conv_left : BOOL ;
END_VAR
VAR
    State : STRUCT
        Waiting : BOOL ;
        Conv_right : BOOL ;
        Assembly : BOOL ;
        Conv_left : BOOL ;
    END_STRUCT ;
END_VAR
BEGIN
NETWORK
TITLE =Initialization
    A    #Initial;
    S    #State.Waiting;
    R    #State.Conv_right;
    R    #State.Assembly;
    R    #State.Conv_left;
NETWORK
TITLE =State: Waiting
//Транспортер ждет в этом состоянии готовую деталь
    AN   #State.Waiting;
    JC   RECH;
    R    #Conv_right;
    R    #Conv_left;
    R    #LED;
    A    #Transp_req;
    R    #State.Waiting;
    S    #State.Conv_right;
NETWORK
TITLE =State: Conv_right
//Это состояние описывает транспортировки части на место заключительной сборки
RECH: AN   #State.Conv_right;
    JC   ENDM;
    S    #Conv_right;
    A    #Clock_Bit;
    =    #LED;
    AN   #L_Barrier;
    R    #Conv_right;
    R    #State.Conv_right;
    S    #State.Assembly;
    AN   #L_Barrier;
    =    #L_Barrier;
```

// (Продолжение на следующей странице)

Решение упражнения 6.2а: FB2 для транспортера (часть 2)

```
NETWORK
TITLE =State: Assembly
// В этом состоянии обработанная деталь удалена и новая необработанная деталь
// положена на ленту. После этого необработанная деталь транспортируется в
// направлении места обработки. Обработка начинается нажатием S4.
ENDM: AN  #State.Assembly;
      JC  LINK;
      S   #LED;
      A   #Acknowledge;
      R   #LED;
      R   #State.Assembly;
      S   #State.Conv_left;

NETWORK
TITLE =State: Conv_left
// В состоянии происходит транспортировка необработанной детали к месту, где она
// обрабатывается
LINK: AN  #State.Conv_left;
      JC  ENDE;
      S   #Conv_left;
      A   #Clock_Bit;
      =   #LED;
      AN  #Transp_req;
      R   #Conv_left;
      R   #State.Conv_left;
      S   #State.Waiting;
ENDE: BEU  ;

END_FUNCTION_BLOCK
```

Решение упражнения 6.2а: OB1

ORGANIZATION_BLOCK "Cycle"

TITLE = "Main Program Sweep (Cycle)"

VERSION : 0.1

VAR_TEMP

OB1_EV_CLASS : BYTE ; //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)

OB1_SCAN_1 : BYTE ; //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)

OB1_PRIORITY : BYTE ; //1 (Priority of 1 is lowest)

OB1_OB_NUMBR : BYTE ; //1 (Organization block 1, OB1)

OB1_RESERVED_1 : BYTE ; //Reserved for system

OB1_RESERVED_2 : BYTE ; //Reserved for system

OB1_PREV_CYCLE : INT ; //Cycle time of previous OB1 scan (milliseconds)

OB1_MIN_CYCLE : INT ; //Minimum cycle time of OB1 (milliseconds)

OB1_MAX_CYCLE : INT ; //Maximum cycle time of OB1 (milliseconds)

OB1_DATE_TIME : DATE_AND_TIME ; //Date and time OB1 started

END_VAR

BEGIN

NETWORK

TITLE =Вызов управляющего блока места

```
CALL "Station" , "Station_DB" (
    Initial           := "INITIALIZATION",
    Proxy_switch      := "INI1",
    Acknowledge       := "S1",
    Clock_bit_q       := "CLOCK_BIT_FAST",
    Clock_bit_s       := "CLOCK_BIT_SLOW",
    LED               := "H1",
    Transp_req        := "Transport_DB".Transp_req);
```

NETWORK

TITLE =Вызов управляющего блока транспортера

```
CALL "Transport" , "Transport_DB" (
    Initial           := "INITIALIZATION",
    L_Barrier         := "LB1",
    Acknowledge       := "S4",
    Clock_bit         := "CLOCK_BIT_FAST",
    LED               := "H4",
    Conv_right        := "K1_CONVR",
    Conv_left         := "K2_CONVL");
```

END_ORGANIZATION_BLOCK

Решение упражнения 6.2b: Расширение до 3-х станций (FB10, часть 1)

```
FUNCTION_BLOCK "ASSEMBLY_LINE"
TITLE =
VERSION : 0.1

VAR
    Station_1 : "STATION";
    Station_2 : "STATION";
    Station_3 : "STATION";
    Transport : "TRANSPORT";
    Conv_busy : BOOL ;
END_VAR
VAR_TEMP
    trans_1 : BOOL ;
    trans_2 : BOOL ;
    trans_3 : BOOL ;
    trans : BOOL ;
END_VAR

BEGIN
NETWORK
TITLE =Вызов Station_1
    A  "INITIALIZATION";
    =  L   1.0;
    BLD 103;
    A  "INI1";
    =  L   1.1;
    BLD 103;
    A  "S1";
    =  L   1.2;
    BLD 103;
    A  "CLOCK_BIT_FAST";
    =  L   1.3;
    BLD 103;
    A  "CLOCK_BIT_SLOW";
    =  L   1.4;
    BLD 103;
    CALL #Station_1 (
        Initial      := L   1.0,
        Proxy_switch := L   1.1,
        Acknowledge  := L   1.2,
        Clock_bit_q   := L   1.3,
        Clock_bit_s   := L   1.4,
        LED           := "H1",
        Transp_req    := #trans_1,
        Conv_busy     := #Conv_busy);
    NOP 0;
```

// (Продолжение на следующей странице)

Решение упражнения 6.2b: Расширение до 3-х станций (FB10, часть 2)

```
NETWORK
TITLE =Invoke Station_2
  A  "INITIALIZATION";
  =  L  1.0;
  BLD 103;
  A  "INI2";
  =  L  1.1;
  BLD 103;
  A  "S2";
  =  L  1.2;
  BLD 103;
  A  "CLOCK_BIT_FAST";
  =  L  1.3;
  BLD 103;
  A  "CLOCK_BIT_SLOW";
  =  L  1.4;
  BLD 103;
  CALL #Station_2 (
    Initial      := L  1.0,
    Proxy_switch := L  1.1,
    Acknowledge  := L  1.2,
    Clock_bit_q  := L  1.3,
    Clock_bit_s  := L  1.4,
    LED          := "H2",
    Transp_req   := #trans_2,
    Conv_busy    := #Conv_busy);
  NOP 0;
```

```
NETWORK
TITLE =Вызов Station_3
  A  "INITIALIZATION";
  =  L  1.0;
  BLD 103;
  A  "INI3";
  =  L  1.1;
  BLD 103;
  A  "S3";
  =  L  1.2;
  BLD 103;
  A  "CLOCK_BIT_FAST";
  =  L  1.3;
  BLD 103;
  A  "CLOCK_BIT_SLOW";
  =  L  1.4;
  BLD 103;
  CALL #Station_3 (
    Initial      := L  1.0,
    Proxy_switch := L  1.1,
    Acknowledge  := L  1.2,
    Clock_bit_q  := L  1.3,
    Clock_bit_s  := L  1.4,
    LED          := "H3",
    Transp_req   := #trans_3,
    Conv_busy    := #Conv_busy);
  NOP 0;
```

// (Продолжение на следующей странице)

Решение упражнения 6.2b: Расширение до 3-х станций (FB10, часть 3)

NETWORK

TITLE =Связывание выходов с входами

```
O  #trans_1;  
O  #trans_2;  
O  #trans_3;  
=  #trans;
```

NETWORK

TITLE =Вызов Transport

```
A  "INITIALIZATION";  
=  L  1.0;  
BLD 103;  
A  "LB1";  
=  L  1.1;  
BLD 103;  
A  "S4";  
=  L  1.2;  
BLD 103;  
A  #trans;  
=  L  1.3;  
BLD 103;  
A  "CLOCK_BIT_FAST";  
=  L  1.4;  
BLD 103;  
CALL #Transport (  
    Initial      := L  1.0,  
    L_Barrier    := L  1.1,  
    Acknowledge  := L  1.2,  
    Transp_req   := L  1.3,  
    Clock_Bit    := L  1.4,  
    LED          := "H4",  
    Conv_right   := "K1_CONVR",  
    Conv_left    := "K2_CONVL");  
NOP 0;  
END_FUNCTION_BLOCK
```

Решение упражнения 6.2b: Расширение до 3-х станций (OB1)

```
ORGANIZATION_BLOCK "CYCLE"  
TITLE =  
VERSION : 0.1  
  
VAR_TEMP  
OB1_EV_CLASS : BYTE ;      //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)  
OB1_SCAN_1 : BYTE ;        //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)  
OB1_PRIORITY : BYTE ;      //1 (Priority of 1 is lowest)  
OB1_OB_NUMBR : BYTE ;      //1 (Organization block 1, OB1)  
OB1_RESERVED_1 : BYTE ;    //Reserved for system  
OB1_RESERVED_2 : BYTE ;    //Reserved for system  
OB1_PREV_CYCLE : INT ;     //Cycle time of previous OB1 scan (milliseconds)  
OB1_MIN_CYCLE : INT ;      //Minimum cycle time of OB1 (milliseconds)  
OB1_MAX_CYCLE : INT ;      //Maximum cycle time of OB1 (milliseconds)  
OB1_DATE_TIME : DATE_AND_TIME ; //Date and time OB1 started  
END_VAR  
  
BEGIN  
NETWORK  
TITLE =Invoke Assembly Line  
  
    CALL "ASSEMBLY_LINE" , "ASSEMBLY_LINE_DB" ;  
    NOP 0;  
  
END_ORGANIZATION_BLOCK
```


Решение упражнения 7.2: Тестирование блока данных

```
FUNCTION FC 72 : INT
TITLE =Exercise 7.2: Testing Data Blocks (only S7-400)
VERSION : 0.1

VAR_INPUT
  DB_NUM : WORD ;
END_VAR
VAR_TEMP
  I_DB_Length : WORD ;
  I_RET_VAL : INT ;
  I_Write_Protect : BOOL ;
END_VAR
BEGIN
NETWORK
TITLE =Testing DB
//only for S7-400

  CALL SFC 24 (
    DB_NUMBER      := #DB_NUM,
    RET_VAL        := #I_RET_VAL,
    DB_LENGTH      := #I_DB_Length,
    WRITE_PROT     := #I_Write_Protect);
  L   #I_RET_VAL;
  L   W#16#0;
  ==I ;
  JC  DBOK;                // DB доступен в рабочей памяти
  TAK ;
  L   W#16#80A1;
  ==I ;
  JC  NODB;                // DB не доступен в CPU
  TAK ;
  L   W#16#80B1;
  ==I ;
  JC  NODB;                // DB доступен в рабочей памяти
  TAK ;
  L   W#16#80B2;
  ==I ;
  JC  DBLM;                // DB только в загрузочной памяти
NODB: L   -1;
  T   #RET_VAL;            // DB не доступен в CPU
  BEU ;
DBLM: L   1;
  T   #RET_VAL;            // DB только в загрузочной памяти
  BEU ;
DBOK: L   0;
  T   #RET_VAL;            // DB доступен в рабочей памяти

END_FUNCTION
```

Решение упражнения 7.3: Создание DB

```
ORGANIZATION_BLOCK OB 100
TITLE =Exercise 7.3: Generating a DB
//Version for S7-400
VERSION : 0.1
```

```
VAR_TEMP
OB100_EV_CLASS : BYTE ;      //16#13, Event class 1, Entering event state, Event
                                // logged in diagnostic buffer
OB100_STARTUP : BYTE ;      //16#81/82/83/84 Method of startup
OB100_PRIORITY : BYTE ;     //27 (Priority of 1 is lowest)
OB100_OB_NUMBR : BYTE ;     //100 (Organization block 100, OB100)
OB100_RESERVED_1 : BYTE ;   //Reserved for system
OB100_RESERVED_2 : BYTE ;   //Reserved for system
OB100_STOP : WORD ;         //Event that caused CPU to stop (16#4xxx)
OB100_START_INFO : DWORD ;  //Information on how system started
OB100_DATE_TIME : DATE_AND_TIME ; //Date and time OB100 started
END_VAR
```

```
BEGIN
NETWORK
TITLE =Creating DB10
```

```
CALL SFC 22(
    LOW_LIMIT      := W#16#A,    // = 10 (DB10)
    UP_LIMIT       := W#16#A,    // "
    COUNT          := W#16#28,   // = 40 ( 40Bytes)
    RET_VAL        := MW 0,
    DB_NUMBER      := QW 38);
```

```
END_ORGANIZATION_BLOCK
```

Решение упражнения 7.4: Копирование DB из загрузочной в рабочую память

```
ORGANIZATION_BLOCK OB 1
TITLE =Exercise: Copying a DB from Load into Working Memory
//Version für S7-400
VERSION : 2.10

VAR_TEMP
OB1_EV_CLASS : BYTE ;      //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1 : BYTE ;        //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY : BYTE ;      //1 (Priority of 1 is lowest)
OB1_OB_NUMBR : BYTE ;      //1 (Organization block 1, OB1)
OB1_RESERVED_1 : BYTE ;    //Reserved for system
OB1_RESERVED_2 : BYTE ;    //Reserved for system
OB1_PREV_CYCLE : INT ;      //Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE : INT ;       //Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE : INT ;       //Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME : DATE_AND_TIME ; //Date and time OB1 started
END_VAR
BEGIN
NETWORK
TITLE =

    A   I   28.0;
    FP  M   0.0;
    JNB _001;
    CALL SFC 20 (
        SRCBLK      := P#DB20.DBX 0.0 BYTE 40,
        RET_VAL     := QW 38,
        DSTBLK      := P#DB10.DBX 0.0 BYTE 40);
    _001: NOP 0;
END_ORGANIZATION_BLOCK
```

Решение упражнения 7.5: Инициализация DB

```

FUNCTION FC 75 : BOOL
TITLE =Exercise 7.5: Initializing DB (only S7-400)
VERSION : 0.1

VAR_INPUT
    DB_NUM : WORD ;
    INI : BYTE ;
END_VAR
VAR_TEMP
    I_RET_VAL : INT ;
    I_DB_Length : WORD ;
    I_WRITE_PROT : BOOL ;
    I_ANY : ANY ;
    DB_No : WORD ;
    I_INI : BYTE ;
    I_RET_VAL1 : INT ;
END_VAR
BEGIN
NETWORK
TITLE =
//Check if DB is in work memory
    CALL "TEST_DB" (
        DB_NUMBER      := #DB_NUM,
        RET_VAL        := #I_RET_VAL,
        DB_LENGTH      := #I_DB_Length,
        WRITE_PROT     := #I_WRITE_PROT);

    L   #I_RET_VAL;
    L   W#16#0;
    ==I ;                               // DB в рабочей памяти
    AN  #I_WRITE_PROT;
    JC  OK;
    CLR ;                               // Инициализация не возможна
    =   #RET_VAL;                       // Возвращается FALSE
    BEU ;
OK:   LAR1 P##I_ANY;                   // Инициализация временной переменной ANY
    L   B#16#10;                       // Идентификатор для ANY
    T   LB [AR1,P#0.0];                // в байте со смещением 0
    L   B#16#2;                        // идентификатор для типа данных BYTE
    T   LB [AR1,P#1.0];                // в байте со смещением 1
    L   #I_DB_Length;                  // длина DB в байтах
    T   LW [AR1,P#2.0];                // в байте со смещением 2
    L   #DB_NUM;                       // номер DB
    T   LW [AR1,P#4.0];                // в байте со смещением 4
    L   P#DBX 0.0;                     // указатель на DBX0.0
    T   LD [AR1,P#6.0];                // в байте со смещением 6
    L   #INI;                          // Инициализация байта
    T   #I_INI;                        // во временной переменной

    CALL SFC 21 (
        BVAL := #I_INI, // возможно только с временной переменной
        RET_VAL := #I_RET_VAL,
        BLK := #I_ANY);
    SET ;
    =   #RET_VAL;
    BE ;

END_FUNCTION

```

Решение упражнения 7.6: Запись сообщения в диагностический буфер (SFC 52)

```
FUNCTION FC 76 : VOID
TITLE =
//Exercise 7.6: Writing a Message in the Diagnostics Buffer (SFC 52)
//Version for S7-300 16 Bit SM
VERSION : 0.0
VAR_TEMP
  I_RET_VAL : INT ;
  info1 : WORD ;
  info2 : DWORD ;
END_VAR

BEGIN
NETWORK
TITLE =

  L   W#16#8;
  T   #info1;
  L   W#16#1;
  T   #info2;

  CALL "WR_USMSG" (
    SEND      := TRUE,
    EVENTN    := W#16#9B0A,
    INFO1     := #info1,
    INFO2     := #info2,
    RET_VAL   := #I_RET_VAL);

END_FUNCTION
```

Решение упражнения 7.7: Счетчик

```
FUNCTION_BLOCK FB 71
TITLE =Exercise 7.7:
//Version for S7-300 16 bit modules
VERSION : 0.1

VAR_INPUT
  CU : BOOL ;
  R : BOOL ;
  PV : INT ;
  PT : TIME ;
END_VAR
VAR_OUTPUT
  Q : BOOL ;
  CV : INT ;
END_VAR
VAR
  Pulse_Counter : "CTU";
  Pulse_Time : "TON";
END_VAR
VAR_TEMP
  Edge_memory : BOOL ;
END_VAR

BEGIN
NETWORK
TITLE =
  A  #Edge_memory;
  = L 1.0;
  BLD 103;
  A  #R;
  = L 1.1;
  BLD 103;
  A( ;
  A  #CU;
  = L 1.2;
  BLD 103;
  CALL #Pulse_Time (
    IN      := L 1.2,
    PT      := #PT,
    Q       := #Edge_memory);

  A  BR;
  ) ;
  JNB _001;
  CALL #Pulse_Counter (
    CU      := L 1.0,
    R       := L 1.1,
    PV      := #PV,
    Q       := #Q,
    CV      := #CV);
_001: NOP 0;

END_FUNCTION_BLOCK
```

Решение упражнения 8.1: Обработка ошибок в FC43 (часть 1)

```

FUNCTION FC 81 : INT
TITLE =Exercise 8.1: Calculation of sum, mean value with error handling
// Solution for S7-300/400
VERSION : 0.0
VAR_INPUT
    Measured_values : ANY ;
END_VAR
VAR_OUTPUT
    Sum : REAL ;
    Mean_value : REAL ;
END_VAR
VAR_TEMP
    Num_Elements : WORD ;
    L_Counter : WORD;
    DB_No : WORD ;
    Sum_1 : REAL ;
    sfc_ret_val : INT ;
    sfc_prgrflt : DWORD ;
    sfc_accflt : DWORD ;
    I_BR : BOOL ;           //пользовательский BR-бит
END_VAR
BEGIN
NETWORK
TITLE =
    L   P##Measured_values; // Загрузите указатель области в указатель"ANY"
    LAR1 ;                  // Указатель области в AR1
    L   B [AR1,P#1.0];      // Прочитать идентификатор типа данных
    L   8;                  // Загрузить идентификатор REAL (16#08)
    ==I ;
    L   -1;                 // Идентификатор типа данных не равен REAL
    JCN ERRO;              // Переход, если тип данных не равен REAL
// Следующие события маскируются:
// Дефектный номер Global DB
// Дефектный номер экземпляра DB,
// Ошибка области в чтении,
// Ошибка длины области в чтении
    CALL SFC 36 (
        PRGFLT_SET_MASK := DW#16#40C0014,
        ACCFLT_SET_MASK := DW#16#0,
        RET_VAL         := #sfc_ret_val,
        PRGFLT_MASKED   := #sfc_prgrflt,
        ACCFLT_MASKED   := #sfc_accflt);

    L   W [AR1,P#2.0];      // Загрузка числа элементов массива
    T   #Num_Elements;      // Инициализация счетчика цикла
    L   W [AR1,P#4.0];      // Загрузка номера DB или 0
    T   #DB_No;             // Если DB_No=0, то OPN DB[DB_No]=NOP
    OPN DB [#DB_No];       // Ошибка времени выполнения теперь маскируется
    L   D [AR1,P#6.0];      // Загрузить указатель на исполнительный адрес
    LAR1 ;                  // в AR1, щибка!! если идентификатор области "DI"
    L   0.000000e+000;      // 0 в Accu1 (сумма =0.0)
    L   #Num_Elements;      // Счетчик в ACCU1; сумма =0 в ACCU2
    BEGN: T   #L_Counter;   // Установить L_Counter
    TAK ;                  // Сумма in ACCU1
    L   D [AR1,P#0.0];      // Элемент массива в ACCU1, сумма in ACCU2
    +R ;                  // Сумма в ACCU1
    +AR1 P#4.0;            // Инкрементирование AR1 на 4 байта

// (Продолжение на следующей странице)

```

Решение упражнения 8.1: Обработка ошибок в FC43 (часть 2)

```

L   #L_Counter;           // L_Counter в ACCU1, сумма в ACCU2
LOOP_BEGN;                // Декрементирование и переход
TAK ;                     // сумма в ACCU1
T   #Sum_1;               // сумма в #Sum_1
// Оценка ошибки
CALL SFC 38 (
  PRGFLT_QUERY             := DW#16#40C0014,
  ACCFLT_QUERY             := DW#16#0,
  RET_VAL                  := #sfc_ret_val,
  PRGFLT_CLR               := #sfc_prgrft,
  ACCFLT_CLR               := #sfc_accflt);

L   #sfc_prgrft;          // Проверка на дефектность DB
L   DW#16#40C0000;
AD  ;                     // Поразрядно "Roundup"
L   -2;                   // Код ошибки для DB не существует
JN  ERRO;                 // Переход, если ошибка
L   #sfc_prgrft;          // Проверка ошибки область или длины области
L   DW#16#14;
AD  ;
L   -4;                   // Идентификатор ошибки для области или длины области
JN  ERRO;                 // Переход, если ошибка
//
// Никакой ошибки не произошло, продолжать "нормальную" обработку
L   #Sum_1;
T   #Sum;                 // назначить выходной параметр #Sum
L   #Num_Elements;        // Сумма в ACCU2, число в ACCU1
DTR ;                     // Беззнаковое целое (16-бит) в REAL
/R  ;                     // Среднее значение в ACCU1
T   #Mean_value;          // Среднее значение в #Mean_value
SET ;                     // Установка BR-бита в 1
=   #I_BR ;
L   0;                   // Идентификатор - все О.К.
T   RET_VAL;
JU  DMSK;                 // Перейдите к демаскированию синхронной ошибки
//
//error evaluation
//
ERRO: CLR ;               // Инструкции в случае ошибки RLO=0
=   #I_BR ;               // BR =0
T   #RET_VAL;             // Загрузить код ошибки в RET_VAL
L   L#-1;                 // Загрузить invalidное Real-число
T   #Sum;
T   #Mean_value;
DMSK: NOP 0;              // Демаскирование синхронных ошибок
CALL SFC 37 (
  PRGFLT_RESET_MASK       := DW#16#40C0014,
  ACCFLT_RESET_MASK       := DW#16#0,
  RET_VAL                  := #sfc_ret_val,
  PRGFLT_MASKED           := #sfc_prgrft,
  ACCFLT_MASKED           := #sfc_accflt);
CLR ;                     // Сначала проверка, RLO = 0
A   #I_BR ;               // копирование пользовательского BR-бита
SAVE ;                    // в системный BR-бит
BEU ;
END_FUNCTION

```


Решение упражнения 9.2: Подсчет готовых деталей (FB1, часть 1)

```
FUNCTION_BLOCK "Station"
TITLE =управление рабочей станцией
AUTHOR : PT41
FAMILY : A2_0
NAME : ST7PRO2
VERSION : 0.0

VAR_INPUT
    Initial : BOOL ;
    Proxy_switch : BOOL ;
    Acknowledge : BOOL ;
    Clock_bit_q : BOOL ;
    Clock_bit_s : BOOL ;
END_VAR
VAR_OUTPUT
    LED : BOOL ;
    Transp_req : BOOL ;
END_VAR
VAR_IN_OUT
    Conv_busy : BOOL ;
END_VAR
VAR
    State : STRUCT
        Process_piece : BOOL ;
        Piece_finished : BOOL ;
        Place_piece_on_conv : BOOL ;
        Wait_for_piece : BOOL ;
        Take_piece_from_conv : BOOL ;
    END_STRUCT ;
END_VAR
BEGIN
NETWORK
TITLE =Initialization
//Посредством ввода "Initial" устанавливается базисное состояние #Process_piece
    A    #Initial;
    S    #State.Process_piece;
    R    #State.Piece_finished;
    R    #State.Place_piece_on_conv;
    R    #State.Wait_for_piece;
    R    #State.Take_piece_from_conv;
    R    #Conv_busy;

NETWORK
TITLE =Состояние: обработка детали
// В этом состоянии заготовка обработана. Обработка завершается,
// когда оператор подтверждает
// посредством кнопки "S1"
    AN   #State.Process_piece;
    JC   Pfin;
    S    #LED;                //светодиод горит постоянно ;
    R    #Transp_req;
    A    #Acknowledge;        //Когда оператор подтверждает,
    R    #State.Process_piece; //выполняется изменение состояния;
    R    #LED;
    S    #State.Piece_finished;
```

// (продолжение на следующей странице)

Решение упражнения 9.2: Подсчет готовых деталей (FB1, часть 2)

NETWORK

TITLE =State: Piece_finished

// В состоянии #Piece_finished оператор ждет разрешение

// поместить заготовку на конвейер. Сигнал #Conv_busy указывает,

// занят конвейер или нет. Когда конвейер свободен, выполняется изменение состояния

// на состояние Place_piece_on_conv.

Pfin: AN #State.Piece_finished;

JC PpCo;

A #Clock_bit_s; //Медленно мигает;

= #LED;

AN #Conv_busy; //Если конвейер свободен,

S #Conv_busy; //он отмечается занятым

R #LED; //выполняется изменение состояния;

R #State.Piece_finished;

S #State.Place_piece_on_conv;

NETWORK

TITLE =State: Place_piece_on_conv

PpCo: AN #State.Place_piece_on_conv;

JC Wait;

A #Clock_bit_q; //quick flashing;

= #LED;

A #Proxy_switch; //When the piece is palced on the conveyor,

S #Transp_req; //the transport is strted,

R #LED; //and the LED is cleared;

A #Transp_req; //When the belt is moving,

AN #Proxy_switch; //an the workpiece has left the proxy switch,

R #State.Place_piece_on_conv; // a state change is performed;

S #State.Wait_for_piece;

NETWORK

TITLE =Состояние: ожидание детали

// Ожидание новой необработанной детали.

Wait: AN #State.Wait_for_piece;

JC TpCo;

R #LED; //светодиод выключен;

A #Proxy_switch; //прибывает новая необработанная деталь,

R #Transp_req; //конвейер стоит,

R #State.Wait_for_piece; //и выполняется изменение состояния;

S #State.Take_piece_from_conv;

NETWORK

TITLE =State: Take_piece_from_conv

// В этом состоянии новая необработанная деталь принимается с конвейера на

// рабочее место

TpCo: AN #State.Take_piece_from_conv;

JC END;

A #Clock_bit_q; //светодиод мигает медленно

= #LED; //

AN #Proxy_switch; //Когда необработанная деталь принимается с ленты,

R #Conv_busy; //конвейер становится свободным,

R #LED; //светодиод гаснет

R #State.Take_piece_from_conv; //и выполняется изменение состояния;

S #State.Process_piece;

END: BEU ;

END_FUNCTION_BLOCK

Решение упражнения 9.2: Подсчет готовых деталей (FB1, часть 3)

```
FUNCTION_BLOCK "Transport"
TITLE =Управление лентой конвейера
VERSION : 0.1

VAR_INPUT
    Initial : BOOL ;
    L_Barrier : BOOL ;
    Acknowledge : BOOL ;
    Transp_req : BOOL ;
    Clock_Bit : BOOL ;
END_VAR
VAR_OUTPUT
    LED : BOOL ;
    Conv_right : BOOL ;
    Conv_left : BOOL ;
    Count_Value : INT ;
END_VAR
VAR
    State : STRUCT
        Waiting : BOOL ;
        Conv_right : BOOL ;
        Assembly : BOOL ;
        Conv_left : BOOL ;
    END_STRUCT ;
    Count: "CTU";           // SFB 0 должна быть включена
END_VAR
BEGIN
NETWORK
TITLE =Initialization

    A   #Initial;
    S   #State.Waiting;
    R   #State.Conv_right;
    R   #State.Assembly;
    R   #State.Conv_left;
    CALL #Count (R:= #Initial,
                CV := #Count_Value);

NETWORK
TITLE =состояние: "Ожидание"
//Конвейер в этом состоянии ждет обработанную деталь.
    AN  #State.Waiting;
    JC  RIGH;
    R   #Conv_right;
    R   #Conv_left;
    R   #LED;
    A   #Transp_req;
    R   #State.Waiting;
    S   #State.Conv_right;

// (Продолжение на следующей странице)
```

Решение упражнения 9.2: Подсчет готовых деталей (FB1, часть 4)

NETWORK

TITLE =состояние: Conv_right

// Это состояние описывает транспортировку законченной детали на

// окончательную сборку

RIGH: AN #State.Conv_right;

JC FINM;

S #Conv_right;

A #Clock_Bit;

= #LED;

AN #L_Barrier;

R #Conv_right;

R #State.Conv_right;

S #State.Assembly;

AN #L_Barrier;

= #L_Barrier;

CALL #Count (CU := #L_Barrier,
CV := #Count_Value);

NETWORK

TITLE =State: Assembly

// В этом состоянии законченная деталь удалена и новая необработанная деталь

// положена на ленту. После этого необработанная деталь транспортируется в

//направлении пустого места обработки нажатием S4.

FINM: AN #State.Assembly;

JC LEFT;

S #LED;

A #Acknowledge;

R #LED;

R #State.Assembly;

S #State.Conv_left;

NETWORK

TITLE =State: Conv_left

// В этом состоянии необработанная деталь транспортируется к месту окончательной

//сборки.

LEFT: AN #State.Conv_left;

JC ENDE;

S #Conv_left;

A #Clock_Bit;

= #LED;

AN #Transp_req;

R #Conv_left;

R #State.Conv_left;

S #State.Waiting;

ENDE: BEU ;

END_FUNCTION_BLOCK

Решение упражнения 9.2: Подсчет готовых деталей (FB1, часть 5)

```
FUNCTION_BLOCK FB 10
TITLE =
VERSION : 0.1
```

```
VAR
  Station_1 : "Station";
  Station_2 : "Station";
  Station_3 : "Station";
  Transport : "Transport";
  Conv_busy : BOOL ;
```

```
END_VAR
```

```
VAR_TEMP
```

```
  Trans_1 : BOOL ;
  Trans_2 : BOOL ;
  Trans_3 : BOOL ;
  Trans : BOOL ;
```

```
END_VAR
```

```
BEGIN
```

```
NETWORK
```

```
TITLE =Station_1
```

```
  CALL #Station_1 (
```

```
    Initial      := "INITIALIZATION",
    Proxy_Switch := "INI1",
    Acknowledge  := "S1",
    Clock_Bit_q  := "CLOCK_BIT_FAST",
    Clock_Bit_s  := "CLOCK_BIT_SLOW",
    LED          := "H1",
    Transp_req   := #Trans_1,
    Conv_busy    := #Conv_busy);
```

```
NETWORK
```

```
TITLE =Station_2
```

```
  CALL #Station_2 (
```

```
    Initial      := "INITIALIZATION",
    Proxy_Switch := "INI2",
    Acknowledge  := "S2",
    Clock_Bit_q  := "CLOCK_BIT_FAST",
    Clock_Bit_s  := "CLOCK_BIT_SLOW",
    LED          := "H2",
    Transp_req   := #Trans_2,
    Conv_busy    := #Conv_busy);
```

```
NETWORK
```

```
TITLE =Station_3
```

```
  CALL #Station_3 (
```

```
    Initial      := "INITIALIZATION",
    Proxy_Switch := "INI3",
    Acknowledge  := "S3",
    Clock_Bit_q  := "CLOCK_BIT_FAST",
    Clock_Bit_s  := "CLOCK_BIT_SLOW",
    LED          := "H3",
    Transp_req   := #Trans_3,
    Conv_busy    := #Conv_busy);
```

```
// (Продолжение на следующей странице)
```

Решение упражнения 9.2: Подсчет готовых деталей (FB1, часть 6)

```

NETWORK
TITLE =Logic: Transp_req
//Формирование логики для #Transp_req
O   #Trans_1;
O   #Trans_2;
O   #Trans_3;
=   #Trans;

NETWORK
TITLE =Transport

    CALL #Transport (
        Initial           := "INITIALIZATION",
        L_Barrier         := "LB1",
        Acknowledge       := "S4",
        Transp_req        := #Trans,
        Clock_Bit         := "CLOCK_BIT_FAST",
        LED               := "H4",
        Conv_right        := "K1_CONVR",
        Conv_left         := "K2_CONVL");

    L           #Transport.Count_Value ;
    ITD ;
    DTB ;
    T           QW12 ;
                                // Расширить до DINT
                                // Преобразовать в BCD

END_FUNCTION_BLOCK
DATA_BLOCK "ASSEMBLY_LINE_DB"
VERSION : 0.1
"ASSEMBLY_LINE"
BEGIN
END_DATA_BLOCK

ORGANIZATION_BLOCK OB 1
TITLE =
VERSION : 0.1

VAR_TEMP
OB1_EV_CLASS : BYTE ;    //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1 : BYTE ;    //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY : BYTE ;    //1 (Priority of 1 is lowest)
OB1_OB_NUMBR : BYTE ;    //1 (Organization block 1, OB1)
OB1_RESERVED_1 : BYTE ; //Reserved for system
OB1_RESERVED_2 : BYTE ; //Reserved for system
OB1_PREV_CYCLE : INT ;    //Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE : INT ;    //Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE : INT ;    //Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME : DATE_AND_TIME ; //Date and time OB1 started
END_VAR

BEGIN
NETWORK
TITLE =

    CALL FB   10 , DB   10 ;

END_ORGANIZATION_BLOCK

```

Решение упражнения 10.2: Коммуникации с SFB PUT/GET (часть 1)

// Загрузить компилируемые блоки в S7-400

DATA_BLOCK DB 14
VERSION : 0.1

"GET"
BEGIN
END_DATA_BLOCK

DATA_BLOCK DB 15
VERSION : 0.1

"PUT"
BEGIN
END_DATA_BLOCK

ORGANIZATION_BLOCK OB 1
TITLE = S7400 writes into S7-300 and reads from S7-300
AUTHOR : PT41
FAMILY : A2_0
NAME : ST7PRO2
VERSION : 0.0

VAR_TEMP
OB1_EV_CLASS : BYTE ; //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1 : BYTE ; //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY : BYTE ; //1 (Priority of 1 is lowest)
OB1_OB_NUMBR : BYTE ; //1 (Organization block 1, OB1)
OB1_RESERVED_1 : BYTE ; //Reserved for system
OB1_RESERVED_2 : BYTE ; //Reserved for system
OB1_PREV_CYCLE : INT ; //Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE : INT ; //Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE : INT ; //Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME : DATE_AND_TIME ; //Date and time OB1 started
NDR_FLAG_14 : BOOL ;
ERROR_FLAG_14 : BOOL ;
DONE_FLAG_15 : BOOL ;
ERROR_FLAG_15 : BOOL ;
STATUS_WORD_14 : WORD ;
STATUS_WORD_15 : WORD ;
END_VAR

BEGIN
NETWORK
TITLE ="SFB_GET"
CALL SFB 14, DB 14 (
REQ := I 28.0,
ID := W#16#1,
NDR := #NDR_FLAG_14,
ERROR := #ERROR_FLAG_14,
STATUS := #STATUS_WORD_14,
ADDR_1 := P#I 0.0 BYTE 1,
ADDR_2 := P#I 4.0 WORD 1,
RD_1 := P#Q 40.0 BYTE 1,
RD_2 := P#Q 42.0 WORD 1);

// (Продолжение на следующей странице)

Решение упражнения 10.2: Коммуникации с SFB PUT/GET (часть 2)

```
NETWORK
TITLE ="SFB_PUT"
CALL SFB_15, DB_15 (
    REQ      := I_28.1,
    ID       := W#16#1,
    DONE     := #DONE_FLAG_15,
    ERROR    := #ERROR_FLAG_15,
    STATUS   := #STATUS_WORD_15,
    ADDR_1   := P#Q 12.0 WORD 1,
    SD_1     := P#I 30.0 WORD 1);
```

```
NETWORK
TITLE =STATUS_WORD to QW38
A( ;
O  #NDR_FLAG_14;
O  #ERROR_FLAG_14;
) ;
JCN_002;
L  #STATUS_WORD_14;
T  QW_38;
_002: NOP 0;
```

```
NETWORK
TITLE =
A( ;
O  #DONE_FLAG_15;
O  #ERROR_FLAG_15;
) ;
JCN_001;
L  #STATUS_WORD_15;
T  QW_38;
_001: NOP 0;
```

```
NETWORK
TITLE =
// Иначе FFFF в QW38
A  I_28.0;
BEC ;
A  I_28.1;
BEC ;
L  W#16#FFFF;
T  QW_38;
END_ORGANIZATION_BLOCK
```


Решение упражнения 10.3: Коммуникации с SFB START/STOP (часть 1)

// Загрузить компилируемые блоки в S7-400

```
DATA_BLOCK DB 19
VERSION : 0.1
"START"
BEGIN
END_DATA_BLOCK
```

```
DATA_BLOCK DB 20
VERSION : 0.1
"STOP"
BEGIN
END_DATA_BLOCK
```

```
ORGANIZATION_BLOCK OB 1
TITLE =
VERSION : 0.1
VAR_TEMP
  OB1_EV_CLASS : BYTE ;      //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
  OB1_SCAN_1 : BYTE ;       //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
  OB1_PRIORITY : BYTE ;     //1 (Priority of 1 is lowest)
  OB1_OB_NUMBR : BYTE ;     //1 (Organization block 1, OB1)
  OB1_RESERVED_1 : BYTE ;   //Reserved for system
  OB1_RESERVED_2 : BYTE ;   //Reserved for system
  OB1_PREV_CYCLE : INT ;     //Cycle time of previous OB1 scan (milliseconds)
  OB1_MIN_CYCLE : INT ;     //Minimum cycle time of OB1 (milliseconds)
  OB1_MAX_CYCLE : INT ;     //Maximum cycle time of OB1 (milliseconds)
  OB1_DATE_TIME : DATE_AND_TIME ; //Date and time OB1 started

  DONE_FLAG_20 : BOOL ;
  ERROR_FLAG_20 : BOOL ;
  DONE_FLAG_19 : BOOL ;
  ERROR_FLAG_19 : BOOL ;
  STATUS_WORD_20 : WORD ;
  STATUS_WORD_19 : WORD ;
END_VAR
```

```
BEGIN
NETWORK
TITLE =
//Введите STRING "P_PROGRAM" в PI_NAME
  L   'P_PR';
  T   MD 100;
  L   'OGRA';
  T   MD 104;
  L   'M';
  T   MB 108;

NETWORK
TITLE ="SFB_STOP"
  CALL SFB 20 , DB 20 (
    REQ      := I 28.0,
    DONE     := #DONE_FLAG_20,
    ERROR    := #ERROR_FLAG_20,
    STATUS   := #STATUS_WORD_20);
```

// (Продолжение на следующей странице)

Решение упражнения 10.3: Коммуникации с SFB START/STOP (часть 2)

```
NETWORK
TITLE ="SFB_START"
CALL SFB 19, DB 19 (
    REQ          := I 28.1,
    DONE         := #DONE_FLAG_19,
    ERROR        := #ERROR_FLAG_19,
    STATUS       := #STATUS_WORD_19);
```

```
NETWORK
TITLE = STATUS_WORD to QW38
A( ;
O #DONE_FLAG_19;
O #ERROR_FLAG_19;
) ;
JCN_001;
L #STATUS_WORD_19;
T QW 38;
_001: NOP 0;
```

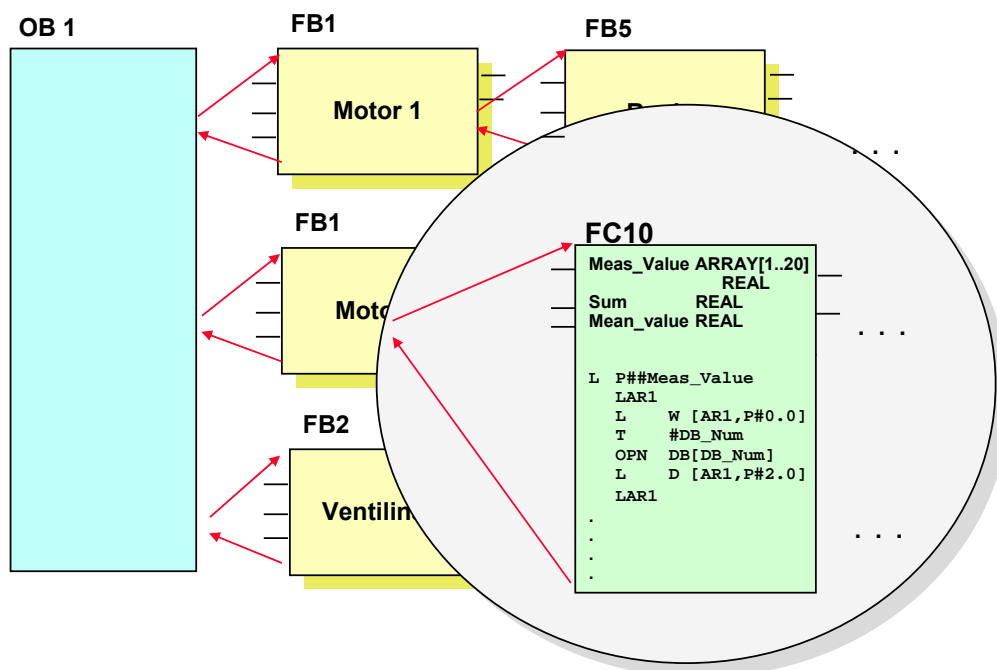
```
NETWORK
TITLE = STATUS_WORD to QW38
A( ;
O #DONE_FLAG_20;
O #ERROR_FLAG_20;
) ;
JCN_002;
L #STATUS_WORD_20;
T QW 38;
_002: NOP 0;
```

```
NETWORK
TITLE =

A I 28.2;      // Иначе FFFF в QW38
BEC ;
A I 28.3;
BEC ;
L W#16#FFFF;
T QW 38;
```

```
END_ORGANIZATION_BLOCK
```

Приложение: косвенный доступ к параметрам FC и FB



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
Datei: PRO2_15D.1Informations- und Trainings-Center
Wissen für Automatisierung

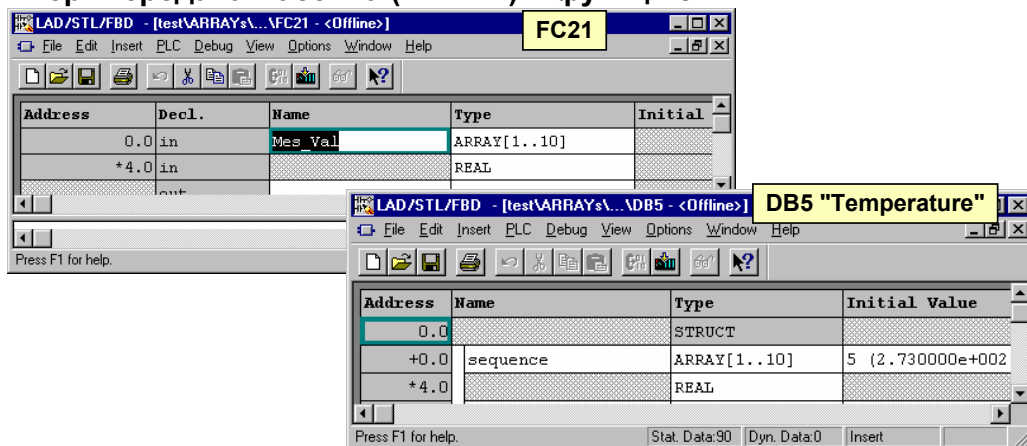
Содержание

Стр

Вызов функции с параметрами комплексного типа	2
Передача параметров комплексных типов	3
Косвенный доступ к сложным типам данных	4
Передача параметра типа POINTER	5
Передача параметров параметрического типа	6
Специальные возможности для элементарных фактических параметров в DB и константах ...	7
Упражнение A.1: Оценка параметров типа DATE_AND_TIME в FC	8
Вызов FB с параметрами сложных типов	9
Косвенная адресация входных и выходных параметров	10
Косвенная адресация in_out - параметров	11
Передача параметров	12
Упражнение A.2: Оценка параметра типа DATE_AND_TIME в FB	13
Упражнение A.3: Оценка In_Out параметров в FB	14
Решение к упражнению A.1: Доступ к параметрам типа DT в FC	15
Решение к упражнению A.2: Доступ к параметрам типа DT в FB	16
Решение к упражнению A.3: Доступ к I/O параметрам в FB (Part 1)	17
Решение к упражнению A.3: Доступ к I/O параметрам в FB (Part 2)	18

Вызов функции с параметрами комплексного типа

Пример: Передача массива (ARRAY) в функцию



Назначение параметров сложного типа возможно только символически

Network 1: Meas_Val объявлен как array в FC21

```
CALL FC 21
Meas_Val:="Temperature".sequence
```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
Date: PRO2_15D.2



Informations- und Trainings-Center
Wissen für Automatisierung

Краткий обзор

Параметры сложного типа данных (ARRAY и STRUCT) предлагают ясный и эффективный способ для передачи больших количеств связанных данных между вызывающим и вызываемым блоками и, таким образом, реализуется концепция "Структурного программирования". Массив или структура может быть передан в вызываемую функцию как полная переменная.

Назначение параметров

Для передачи в вызываемую функцию, передаваемый параметр должен быть того же самого типа данных, что и фактический параметр, Параметр сложного типа (тип данных: ARRAY, STRUCT, DATE_AND_TIME и STRING) может быть назначен только символически. Так как переменные сложного типа данных могут находиться только в блоках данных или в локальном стеке данных, фактический параметр нужно помещать или в блок данных (глобальный блок или экземпляр DB) или в локальном стеке данных вызывающего блока. После того, как STL/LAD/FBD редактор проверил совместимость типов данных фактического параметра и блочного параметра при передаче параметров в FC, в вызываемый FC передаются только POINTER с DB номером и межзонным указателем на фактический параметр. Этот POINTER макрокомандой CALL помещается в L-стек вызываемого блока (область V). Этот POINTER имеет затем большое значение для программиста, в частности когда к переданному параметру нужно обращаться косвенно (см. приложение).

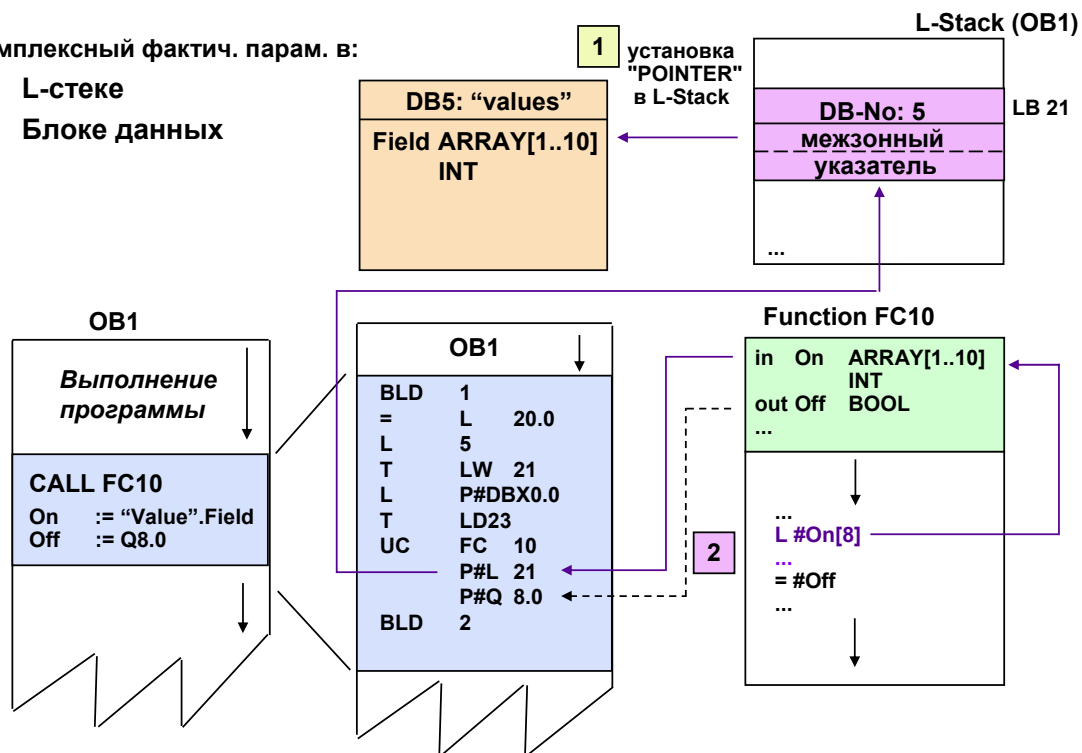
Замечание

- Число занятых локальных данных определяется выбором опции меню *View -> Block Properties*.
- Компоненты массивов или структур могут также быть переданы блочному параметру, если блочный параметр и соответствующая компонента имеют один и тот тип данных.

Передача параметров комплексных типов

Комплексный фактич. парам. в:

- L-стеке
- Блоке данных



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
 Datei: PRO2_15D.3



Informations- und Trainings-Center
 Wissen für Automatisierung

Передача параметров

Для сложных типов данных (DT, STRING, ARRAY, STRUCT и UDT) фактические параметры находятся или в блоке данных или в L-стеке вызывающего блока (V-область).

Так как 32-разрядный межзонный указатель не может адресовать фактический параметр в DB, STL/LAD/FBD-редактор сохраняет в L-стеке вызываемого блока 48-разрядный "POINTER", который указывает на фактический параметр.

Во время обращения 32-разрядный межзонный указатель передается этому "POINTER". Внутри FC доступ к фактическому параметру происходит затем посредством двойного перехода.

Установка "POINTER" в L-стеке вызывающего блока происходит перед фактическим переходом на вызываемый FC.

Косвенный доступ к сложным типам данных

Address	Declaration	Name	Type	Start value	Comment
0.0	in	Meas_Val	ARRAY[1..8]		
*4.0			REAL		
32.0	out	Sum	REAL		
36.0	out	Mean_Val	REAL		
	in out				
0.0	temp	DB_Num	WORD		

Network 1: Определение номера DB и начального адреса

```

L   P## Meas_Val           // Загрузите адрес POINTER в ACCU1,
LAR1                          // и оттуда загрузите в AR1;
L   W [AR1,P#0.0]          // Определите номер DB
T   #DB_Num                // и загрузите его во временную переменную;
OPN DB[DB_Num]             // Открыть DB
L   D [AR1,P#2.0]          // Определите указатель на область
LAR1                          // и загрузите его в AR1;

```

Network 2: Вычисление суммы

```

L   0.000000e+000          // 0 в ACCU1 (sum =0.0)
L   8                      // Счетчик в ACCU1; Sum=0 в ACCU2
BEGN: TAK                  // сумма в ACCU1, счетчик в ACCU2
ENT                          // счетчик в ACCU3
L   D[AR1,P#0.0]           // элемент массива в ACCU1
+R                          // сумма в ACCU1, счетчик в ACCU2
+AR1 P#4.0;                // Увеличение AR1 на 4 байта
TAK                          // Счетчик цикла в ACCU1, сумма в ACCU2
LOOP BEGN;                 // Уменьшить на 1 счетчик цикла и переход, если
                             // необходимо
T   #Sum                   // Запись суммы в #Sum

```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
 Datei: PRO2_15D.4



Informations- und Trainings-Center
 Wissen für Automatisierung

Косвенный доступ При передаче сложных типов данных типа ARRAY и STRUCTS, доступ к ним может осуществляться косвенно. Косвенный доступ к переданным фактическим параметрам сложного типа данных делается в два шага:

1. Сначала межзонный указатель POINTER, который был передан в локальный стек данных, определяется посредством операции:

L P ## Meas_Val в вызывающем блоке.

2. Для фактического доступа к фактическим параметрам затем необходимо

оценить информацию в POINTER, который ссылается на фактический (актуальный) операнд:

L P ## Meas_Val // возвращает межзонный область указатель на POINTER
 LAR1 // и грузит межзонный указатель в адресный регистр

L W [AR1, P # 0.0] // возвращает номер DB фактического параметра, если
 // параметр находится в DB, иначе 0

L D [AR1, P #2.0] // возвращает межзонный указатели на фактический
 // параметр

Результат затем вычисляется обычным способом.

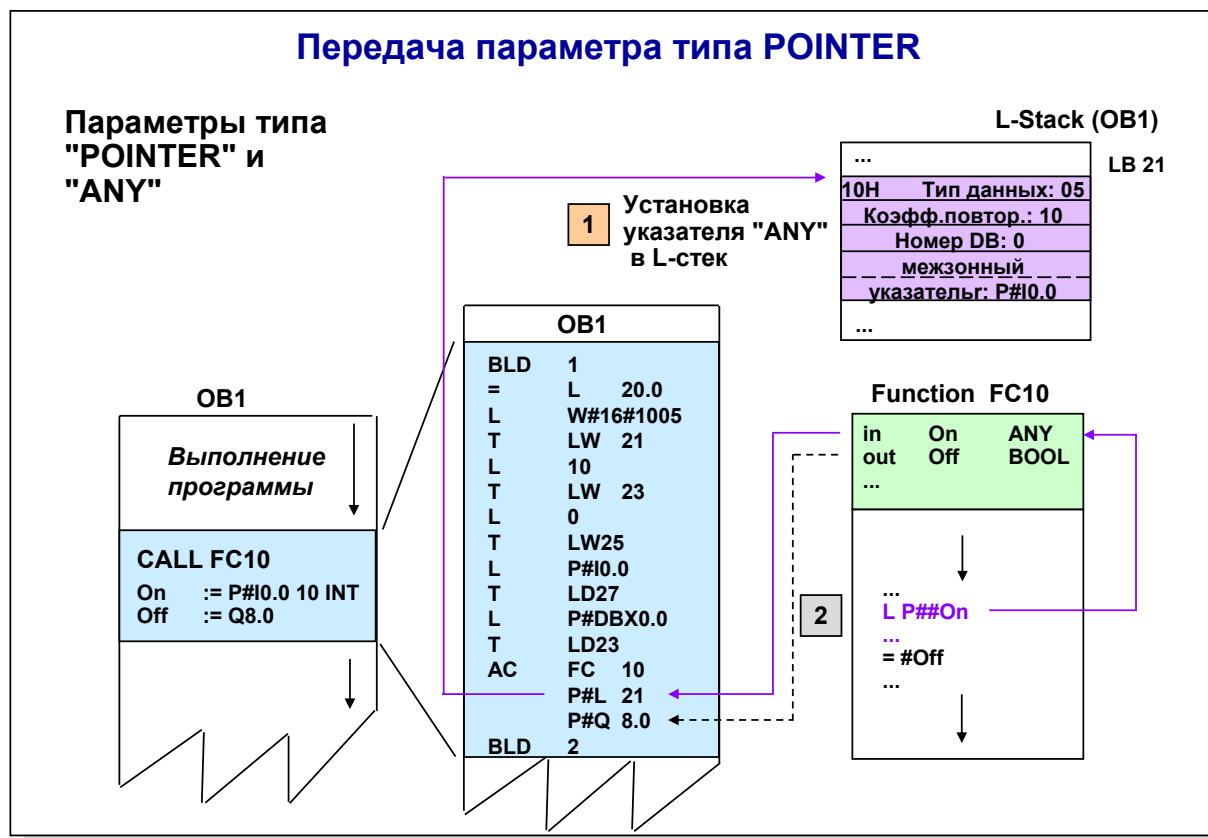
Обратите внимание Чтобы получить доступ в вышеупомянутом примере, программист должен установить содержание регистра DB и регистра AR1 так, чтобы они адресовали первый компонент параметра.

Команда

L Meas_Val [0]

также гарантирует, что блок данных открыт через регистратор DB, а регистр AR1 установлен на начало передаваемого ARRAY.

Передача параметра типа POINTER



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
Datei: PRO2_15D.5Informations- und Trainings-Center
Wissen für Automatisierung

Передача параметров

Если параметр типа данных "POINTER" или "ANY" передан в FC, то STL/LAD/FBD-редактор устанавливает соответствующую структуру данных в L-стеке вызывающего блока.

При вызове FC вызываемому FC передается 32-разрядный межзонный указатель, который указывает на эту структуру данных ("POINTER" или "ANY").

Внутри вызываемого FC не возможно обратиться к параметрам непосредственно из-за отсутствующей информации о типе данных, которые вызваны через этот указатель "POINTER" или "ANY".

Оценка содержания "POINTER" или "ANY" должна быть проведена пользователем с помощью элементарных команд STL внутри вызываемого FC (см. Главу 2).

Установка структуры "POINTER" или "ANY" в L-стеке вызывающего блока происходит перед фактическим переходом в вызываемый FC.

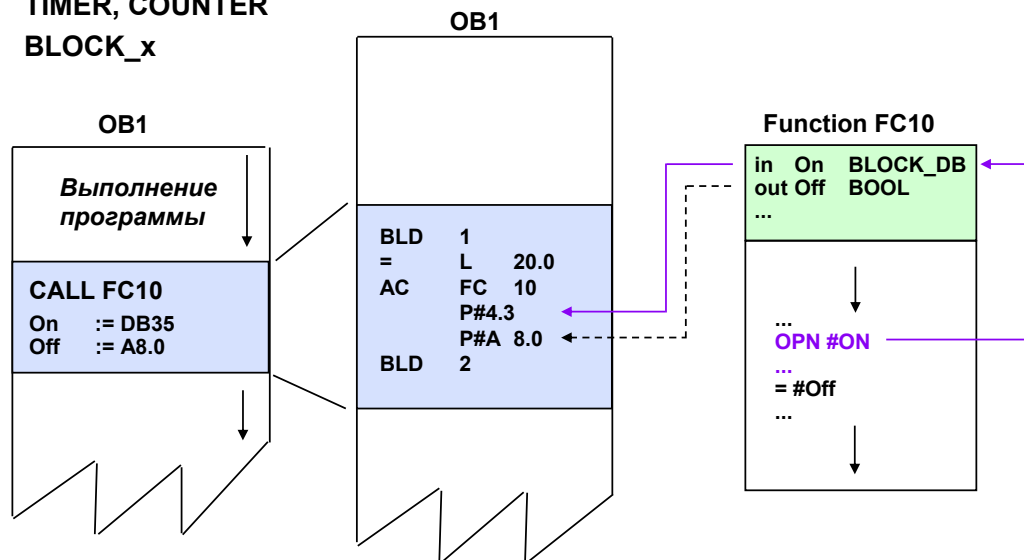
Исключение

Исключение из вышеупомянутого правила - STL/LAD/FBD-редактор, когда в блочном параметре типа данных "ANY" установлен фактический параметр типа данных "ANY", *сохраненный в L-стеке вызывающего блока*. В этом случае STL/LAD/FBD-редактор не устанавливает дополнительный указатель "ANY" в L-стеке вызывающего блока, но передает непосредственно в вызываемую FC 32-разрядный межзонный указатель на уже существующий указатель "ANY" (в L-стеке вызывающего блока). Таким образом, во времени выполнения этот указатель "ANY" может управляться вызывающим блоком и выполнять "косвенное" назначение FC с фактических параметров.

Передача параметров параметрического типа

Блочные параметры:

- TIMER, COUNTER
- BLOCK_x



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
Datei: PRO2_15D.6



Informations- und Trainings-Center
Wissen für Automatisierung

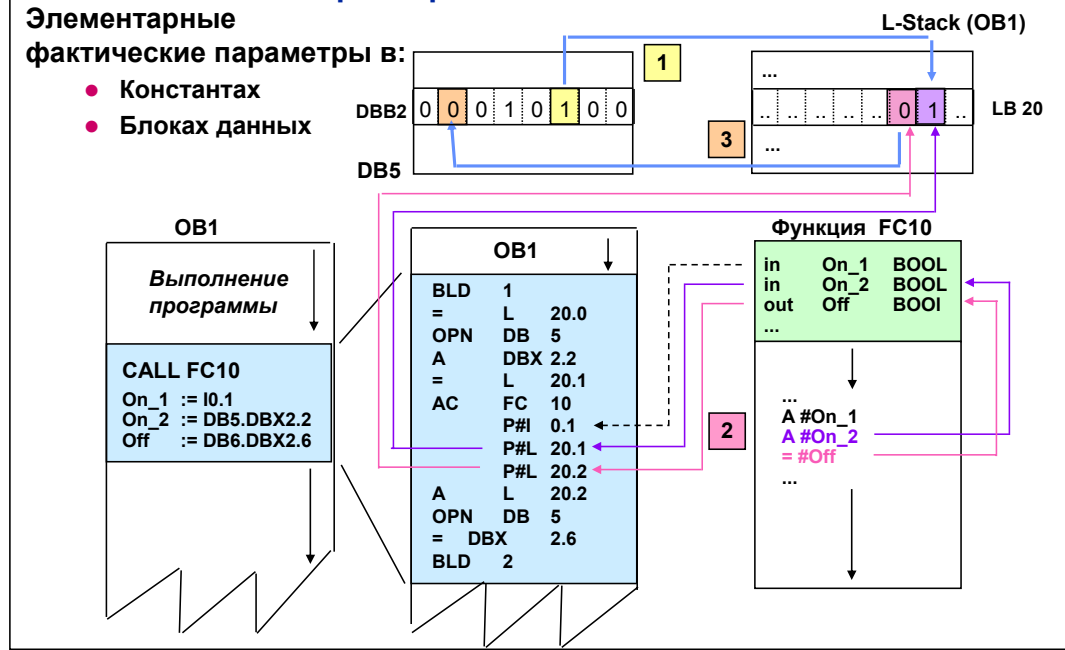
Передача параметров

Передача параметров типа: TIMER, COUNTER и BLOCK_x самая простая. В этом случае вместо 32-разрядного межзонного указателя вызываемому FC просто передается номер текущего (актуального) TIMER или COUNTER или BLOCK_x.

Специальные возможности для элементарных фактических параметров в DB и константах

Элементарные фактические параметры в:

- Константах
- Блоках данных



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
Date: PRO2_15D.7Informations- und Trainings-Center
Wissen für Automatisierung

Передача параметров

Если параметр FC типа in, out, или in_out передан с помощью константы или параметра, который находится в DB, STL/LAD/FBD-редактор сначала резервирует необходимую память в L-стеке вызывающего блока и затем копирует фактические значения параметров типа in и in_out в L-стек.

Для выходного параметра (out) происходит резервирование области памяти в L-стеке, но без инициализации.

Только после того, как это сделано, происходит переход в вызываемый FC, при этом STL/LAD/FBD-редактор сохраняет в каждом случае межзонный указатель на области памяти L-стека вызываемого FC.

После перехода обратно в вызываемый блок, результаты - параметры типа с out или in / out - будут скопированы обратно в фактические параметры.

Следствия

Этот механизм показывает, что внутри вызываемого FC входные параметры могут только быть просмотрены, и выходные параметры могут только быть записаны.

Если входной параметр записан, то, хотя соответствующее значение сохранено в L-стеке, после обработки FC, передача данных в фактические параметры не происходит.

Таким же образом, выходные параметры могут только быть написаны, а не прочитаны. При чтении выходного параметра читается - из-за отсутствующей инициализации - неопределенное значение из L-стека.

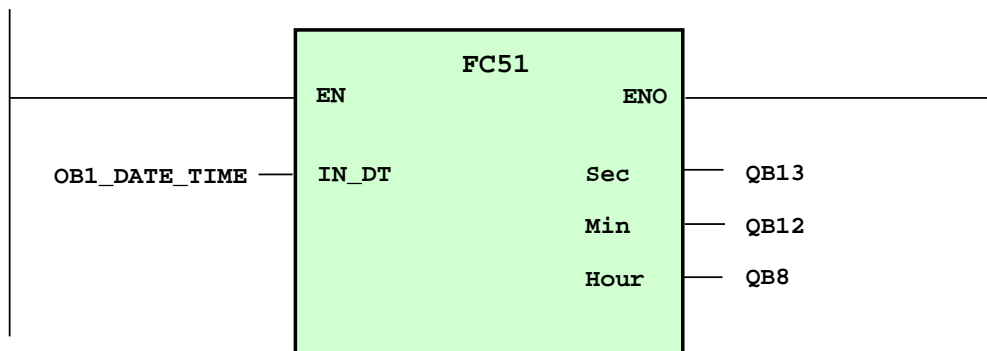
Параметры типа in_out вызывают меньше всего проблем. Они назначены со значениями фактических параметров перед обращением также как после обращения.

Важно

Выходные параметры должны быть записаны в вызываемом FC, иначе неопределенное значение из L-стека будет скопировано обратно в фактический параметр.

Если Вы не можете гарантировать, что выходные параметры будут записаны, Вы должны использовать параметры типа in_out.

Упражнение А.1: Оценка параметров типа DATE_AND_TIME в FC



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
Datei: PRO2_15D.8Informations- und Trainings-Center
Wissen für Automatisierung

Краткий обзор

Следующее упражнение должно показать, как Вы можете косвенно обращаться к входным, выходным и проходным параметрам сложного типа внутри функции.

Вы должны использовать ту же самую технологию, если Вы должны косвенно обратиться к другим сложным типам данных, типа ARRAY, STRUCT или STRING.

Постановка задачи

Создайте FC51 со следующими свойствами:

- FC51 имеет входной параметр # IN_DT с типом данных DATE_AND_TIME
- В 3-х выходных параметрах #Sec, #Min и #Hour FC51 возвращает секунды, минуты и часы - компоненты DT-параметра, переданного этому блоку.

Выполнение

1. Создайте FC51 с требуемыми свойствами.
2. Вызовите FC51 в OB1. Передайте блочному параметру # IN_DT переменной OB1_DATE_TIME из стартовой информации OB1.
3. Загрузите блоки в CPU и проверьте программу.

Вызов FB с параметрами сложных типов

Пример: Передача массивов в функциональный блок

FB17

Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Meas_1	ARRAY[1..10]		
*4.0	in		REAL		
40.0	out	Sum_1	REAL	0.000000e+000	
44.0	out	Sum_2	REAL	0.000000e+000	
48.0	in_out	Meas_2	ARRAY[1..15]		
*4.0	in_out		REAL		
54.0	stat	DB_Num	INT		

DB2 "Temperature"

Address	Name	Type	Initial
0.0		STRUCT	
+0.0	Cylinder	ARRAY[1..10]	
*4.0		REAL	
+40.0	Shaft	ARRAY[1..15]	
*4.0		REAL	
=100.0		END_STRUCT	

Назначение сложных параметров возможно только символически

Network 1:

```

CALL FB 17, DB 30
Meas_1      := "Temperature".Cylinder
Sum_1      := MD20
Sum_2      := MD30
Meas_2     := "Temperature".Shaft

```

SIMATIC S7

Siemens AG 1999. All rights reserved.

 Datum: 04.11.2005
 Datei: PRO2_15D.9

 Informations- und Trainings-Center
 Wissen für Automatisierung

Сложные типы данных

Точно так, как с функциями, адреса параметров сложного типа (ARRAY, STRUCT, DATE_AND_TIME, и STRING) могут быть переданы полностью вызываемому функциональному блоку.

Для передачи должен быть передан фактический параметр того же самого типа данных, что и параметр в вызываемом функциональном блоке.

Назначение такого параметра возможно только символически.

Входные и выходные параметры

Для входных и выходных параметров сложного типа данных, соответствующие области для значений фактических параметров размечены в экземпляре DB. При вызове FB фактические значения входного параметра копируются в экземпляр DB с использованием SFC 20 (BLKMOV) ("передача по значению") прежде, чем происходит фактический переход на раздел команд FB.

Тем же самым способом, значения выходного параметра копируются из экземпляра DB обратно в фактический параметр после того, как FB был обработан.

В результате может происходить немалое количество копирований (процессорное время!) при назначении входных и выходных параметров. Этих копирований нет при работе с параметрами типа in_out.

Параметры in_out

Никакой "передачи по значению" не происходят у in_out параметров сложного типа. Шесть байтов просто зарезервировано для каждого in_out параметра в экземпляре блока данных. В эти байты записывается POINTER на фактические параметры ("передача по ссылке").

Замечание

Входные и выходные параметры сложного типа данных могут быть инициализированы в разделе объявления FB, однако это не справедливо для in_out параметров.

Входные и выходные параметры сложного типа данных могут быть не назначены при вызове FB, in_out параметры должны быть обязательно назначены. Косвенный доступ через память или регистры к параметрам типа in, out или in_out сложного типа данных отлична от таковой lkw элементарных параметров.

Косвенная адресация входных и выходных параметров

Address	Declaration	Name	Type	Start value	Comment
0.0	in	Meas_1	ARRAY[1..10]		
*4.0			REAL		
40.0	out	Sum_1	REAL	0.000000e+000	
44.0	out	Sum_2	REAL	0.000000e+000	
48.0	in_out	Meas_2	ARRAY[1..15]		
*4.0	in_out		REAL		
54.0	stat	DB_Num	INT	0	

Network 1: Определение стартового адреса Meas_1

```

LAR1 P##Meas_1      // Загрузите межзонный указатель на параметр без
                    // смещения адреса (мультиэкземпляр) в AR1
TAR2                // Загрузить смещение адреса в ACCU1
+AR1                // Добавить смещение адреса к AR1;
                    // AR1 теперь указывает на параметр в экземпляре DB
                    // экземпляр DB уже открыт

```

Network 2: Доступ к Meas_1

```

L      0.000000e+000 // 0 в ACCU1 (сумма =0.0)
L      10            // Счетчик в ACCU1; сумма =0 в ACCU2
BEGN:  TAK           // Сумма в ACCU1, счетчик в ACCU2
      ENT           // Счетчик в ACCU3
      L      D[AR1,P#0.0] // Компонент массива в ACCU1
      +R           // Сумма в ACCU1, счетчик в ACCU2
      +AR1 P#4.0;    // Увеличение AR1 на 4 байта
      TAK           // Счетчик цикла в ACCU1, сумма в ACCU2
      LOOP BEGN;    // Уменьшить счетчик цикла на 1 и переход, если это
                    // необходимо
      T      #Sum_1  // Запись суммы в #Sum_1

```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
 Datei: PRO2_15D.10



Informations- und Trainings-Center
 Wissen für Automatisierung

Косвенный доступ

Передача сложных типов данных типа ARRAY и STRUCT может с помощью косвенного доступа внутри вызываемого блока. Косвенный доступ к переданным фактическим параметрам сложного типа данных делается в два шага:

1. Сначала посредством операции:

```

LAR1 P##Meas_1 // возвращает межзонный указатель на параметр
               // без смещения адреса

```

межзонный указатель на параметр в экземпляре DB загружается в AR1. Указатель, определенный таким образом содержит идентификатор области DI и тот же самый адрес byte.bit, который также отображается редактором во время объявления параметра в первом столбце раздела описаний.

В случае мультиэкземпляра это не соответствует исполнительному адресу входного или выходного параметра в экземпляре DB. Необходимо добавить смещение адреса из AR2, который идентифицирует начало области данных образца в случае мультиэкземпляра к указателю в AR1.

```

TAR2      // Загрузка смещение адреса в ACCU1
+ AR1     // Добавление смещение адреса к AR1;

```

2. После того, как это сделано, может происходить фактический доступ к параметру ввода/вывода экземпляра DB, уже открытого при вызове FB макрокомандой CALL.

```

L D[AR1, P#0.0] // Загружают 1-ый компонент Meas_1 и т.д.

```

Обратите внимание

Косвенный доступ к параметрам in, out и in_out элементарного типа или к переменным происходит таким же образом, как случае, когда значения операндов сохранены в экземпляре.

Косвенная адресация in_out - параметров

Address	Declaration	Name	Type	Start value	Comment
0.0	in	Meas_1	ARRAY[1..10]		
*4.0			REAL		
40.0	out		REAL		
44.0	out	Sum_2	REAL	0.000000e+000	
48.0	in_out	Meas_2	ARRAY[1..15]		
*4.0	in_out		REAL		
54.0	stat		INT		

Network 3: Определение начального адреса Meas_2

```

LAR1 P##Meas_2      // Загрузка межзонного указателя на POINTER
TAR2                // Загрузка смещения адреса в ACCU1, прибавление к AR1;
+AR1                // AR1-новый указатель на POINTER в экземпляр DB
L   W [AR1,P#0.0]   // Загрузка номера DB из POINTER в ACCU1
T   #DB_Num         // Запись номера DB (или 0) в статическую переменную
OPN DB [#DB_Num]    // Открытие DB
L   D [AR1,P#2.0]   // Загрузка межзонного указателя на параметр
LAR1                // Загрузка указателя в AR1, AR1 указатель на параметр

```

Network 4: Доступ к Meas_2

```

L   0.000000e+000   // 0 в ACCU1 (сумма =0.0)
L   15              // Счетчик в ACCU1; сумма=0 в ACCU2
BEGN: TAK           // Сумма в ACCU1, счетчик в ACCU2
ENT                // Счетчик в ACCU3
L   D[AR1,P#0.0]    // Элемент массива в ACCU1
+R                // Сумма в ACCU1, счетчик в ACCU2
...                // ...

```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
 Datei: PRO2_15D.11



Informations- und Trainings-Center
 Wissen für Automatisierung

Косвенный доступ

Косвенный доступ к in_out параметрам сложного типа имеет отличия по сравнению с доступом к входным и выходным параметрам.

При передаче in/out параметров сложного типа значение не копируется, а только POINTER на параметр заносится в экземпляр DB.

Фактический доступ происходит в три шага:

1. Сначала, посредством операции:

LAR1 P ## Meas_2 // возвращает межзонный указатель на POINTER
 межзонный указатель на POINTER для параметра загружается в регистр AR1. Как и в предыдущем случае, адрес byte.bit указателя в AR1 не идентифицирует исполнительный адрес POINTER в экземпляре DB. В случае мультиэкземпляра смещение из AR2 должно еще быть добавлено к указателю в регистре AR1:

```

TAR2    // загрузочный адрес в ACCU1, добавляется к AR1;
+ AR1    // AR1 теперь указывает на POINTER в экземпляре DB;

```

2. На следующем шаге оценивается информация в POINTER, в случае необходимости открывается DB, в котором фактический параметр размещен и межзонный указатель на фактические операнды загружен в регистр AR1:

```

L W [AR1, P # 0.0] // Загружается номер DB из POINTER в ACCU1
T #DB_Num         // Передача номер DB (или 0) в переменную
OPN DB [# DB_NUM] // Открытие DB
L D [AR1, P #2.0] // Загружается межзонный указатель на параметр
LAR1            // Загрузочный указатель в AR1, AR1 указывает на параметр

```

3. После того, как это сделано, может происходить доступ к фактическому параметру :

```

L D [AR1, P # 0.0] // Загружается 1-ый компонент Meas_2 и т.д.

```

Передача параметров

Глубина вложения:

- S7-300: max. 8 S7-400: max. 16



Передача зависит от блока, типа данных и вида параметра:

Вызов	FC выз. FC	FB выз. FC	FC выз. FB	FB выз. FB
Тип данных	Е С Р	Е С Р	Е С Р	Е С Р
Input -> Input	x - -	x x -	x - x	x x x
Output -> Output	x - -	x x -	x - -	x x -
in/out -> Input	x - -	x - -	x - -	x - -
in/out -> Output	x - -	x - -	x - -	x - -
in/out -> in/out	x - -	x - -	x - -	x - -

Е: Элементарный тип данных

С: Комплексный тип данных

Р: Параметрический тип (Timer, Counter, Block_x)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
Datei: PRO2_15D.12



Informations- und Trainings-Center
Wissen für Automatisierung

Краткий обзор

"Передача" блочных параметров - специальная форма доступа или назначения блочных параметров. "Передача" - это механизм, который снабжает формальный параметр вызываемого блока фактическим параметром из вызывающего блока.

Ограничения на тип параметра

Как общее правило, фактический параметр должен иметь тот же самый тип данных что и формальный параметр. Кроме того, входные параметры вызываемого блока могут быть установлены только во входных параметрах вызываемого блока, а выходные параметры - только в выходных параметрах.

In_out параметр вызываемого блока может в принципе быть установлен в параметры любого типа (in, out и in_out) вызываемого блока.

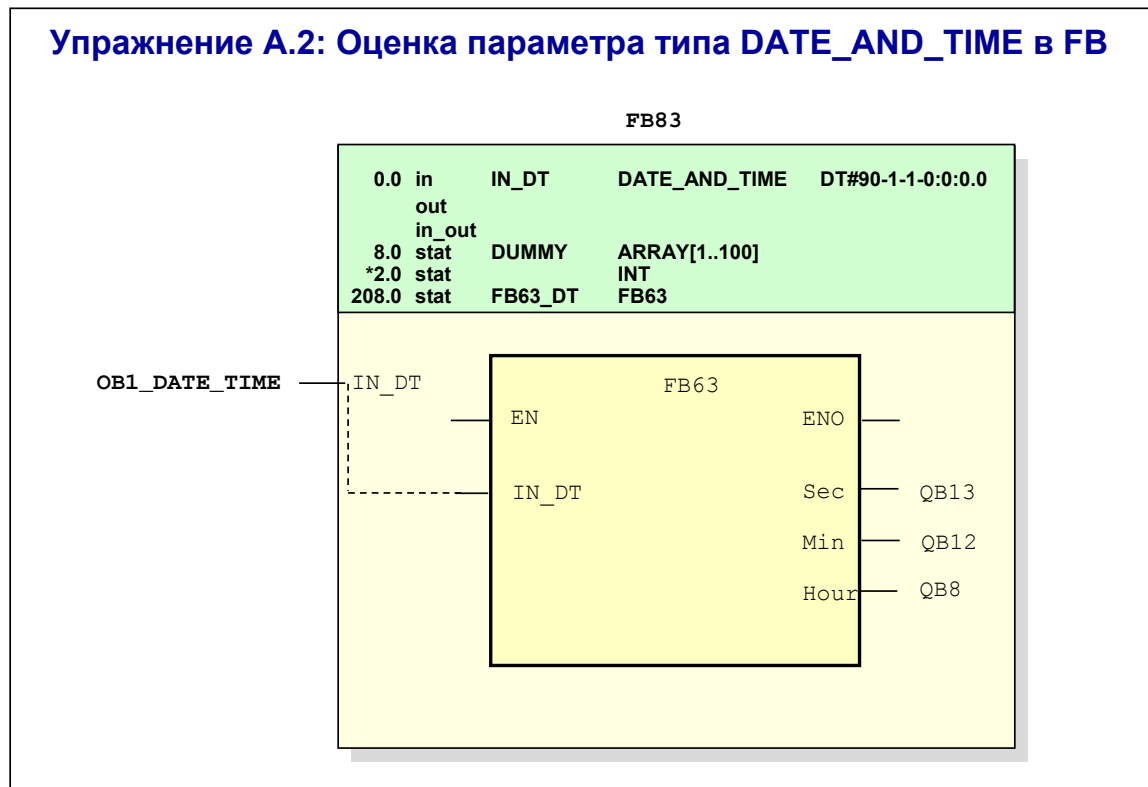
Ограничения на тип данных

В отношении типов данных накладываются ограничения в зависимости различных способов хранения в памяти блочных параметров при вызове FC или FB. Параметры элементарного типа данных могут передаваться без ограничений. Сложные типы данных во входных и выходных параметрах могут передаваться только, если вызываемый блок - FB. Блочные параметры с параметрическими типами: TIMER, COUNTER и BLOCK_x могут передаваться только из входного параметра в входной параметр, если вызываемый блок - FB.

Обратите внимание

Передача параметров типа: TIMER, COUNTER и BLOCK_x может быть выполнена в FC с помощью косвенной адресации. Номер нужного таймера, счетчика или блока может быть, например, передан как параметр типа данных WORD из вызываемого блока вызываемому блоку. Внутри последнего (вызываемого блока), этот параметр может затем быть перемещен во временную переменную и через эту переменную соответствующий таймер, счетчик или блок могут быть затем обработаны.

Упражнение A.2: Оценка параметра типа DATE_AND_TIME в FB



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
Date: PRO2_15D.13Informations- und Trainings-Center
Wissen für Automatisierung

Краткий обзор

Следующее упражнение должно показать, как Вы можете косвенно обращаться к входному или выходному параметру сложного типа данных внутри мультиэкземпляра функционального блока .
Вы должны использовать ту же самую технологию, если Вам нужно косвенно обратиться к другим сложным типам данных - ARRAY, STRUCT или STRING.

Постановка задачи

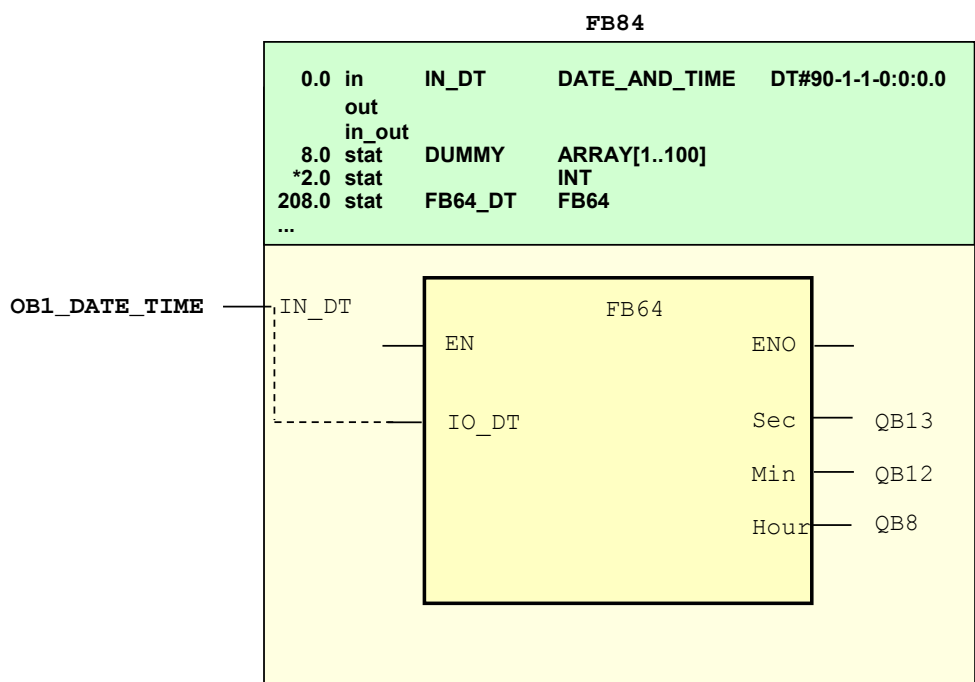
Создайте FB63 со следующими параметрами:

- FB63 имеет входной параметр **#IN_DT** типа: DATE_AND_TIME
- В 3-х выходных параметрах **#Sec**, **#Min** и **#Hour**, FB63 возвращает секунды, минуты и часы - компоненты DT-параметра, переданного ему.

Выполнение

1. Создайте FB63 с требуемыми свойствами.
2. Чтобы проверить, является ли созданный FB63 способным работать в модели мультиэкземпляров, вызовите экземпляр FB63 в FB более высокого уровня
Создайте FB более высокого уровня - FB83. Сначала объявите входной параметр **#IN_DT** типа DT в FB83. Затем объявите статическую переменную **#DUMMY** типа ARRAY[1 .. 100] INT и экземпляр FB63 с именем **#FB63_DT**.
3. Вызовите экземпляр **#FB63_DT** внутри FB83 и передайте входной параметр **#IN_DT** этого экземпляра во входной параметр **#IN_DT** FB83. Обеспечить выходной параметр экземпляра FB63_DT выходами - байты QB8, QB12 и QB13.
4. Вызовите FB83 с экземпляром DB83 в OB1. Назначьте входному параметру **#IN_DT** переменную **OB1_DATE_TIME** из стартовой информации OB1.
5. Загрузите блоки в CPU и проверьте вашу программу.

Упражнение A.3: Оценка In_Out параметров в FB



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.11.2005
Datei: PRO2_15D.14Informations- und Trainings-Center
Wissen für Automatisierung

Краткий обзор

Следующее упражнение должно показать, как Вы можете косвенно обращаться к in_out параметрам сложного типа данных внутри мультиэкземпляра функционального блока.

Вы должны использовать ту же самую технологию, если Вы хотите косвенно обратиться к другим сложным типам данных, типа ARRAY, STRUCTS или STRING.

Постановка задачи

Создайте FB64 со следующими свойствами:

- FB64 имеет in_out параметр **#IO_DT** типа DT
- В 3-х выходных параметрах **#Sec**, **#Min** и **#Hour** FB64 возвращает секунды, минуты и часы - компоненты DT - параметра, переданного ему.

Выполнение

1. Создайте FB64 с требуемыми свойствами.
2. Создайте FB84. Сначала объявите входной параметр **#IN_DT** типа DT в FB84. Затем объявите статическую переменную **#DUMMY** типа ARRAY [1 .. 100] INT и экземпляр FB64 с именем **#FB64_DT**
3. Вызовите экземпляр **#FB64_DT** внутри FB84 и назначьте in_out параметру **#IO_DT** этого экземпляра входной параметр **#IN_DT FB84**. Обратите внимание, что на прямая передача входного параметра для in_out параметра не разрешается. Обеспечить выходной параметр экземпляра FB64_DT выходами - байты QB8, QB12 и QB13, как в предыдущей задаче.
4. Вызовите FB84 с экземпляром DB84 в OB1. Назначьте входному параметру **#IN_DT FB84** переменную **OB1_DATE_TIME** из стартовой информации OB1.
5. Загрузите блоки в CPU и проверьте программу.

Решение к упражнению А.1: Доступ к параметрам типа DT в FC

```
FUNCTION FC 51 : VOID
TITLE =
VERSION : 0.1
VAR_INPUT
    IN_DT : DATE_AND_TIME ;
END_VAR
VAR_OUTPUT
    Sec : BYTE ;
    Min : BYTE ;
    Hour : BYTE ;
END_VAR
VAR_TEMP
    DB_Num : WORD ;
END_VAR
BEGIN
NETWORK
TITLE =
// В случае параметров in, out, или in_out сложного типа данных,
// межзонный указатель на переменную POINTER (6 байтов), который установлен в
// L-стеке вызывающего блока макрокомандой CALL,
// перемещен вызываемой функции для передачи параметров.
// Содержание переменной POINTER указывает на фактические операнды. Для
// косвенного доступа создан межзонный указатель на этот POINTER .
// На следующей стадии, содержание переменной POINTER прочитано и организован
// доступ к фактическим операндам через эту информацию.
    L   P##IN_DT; // Загрузка межзонного указателя из POINTER в ACCU1
    LAR1 ;        // Загрузка указателя в AR1, в AR1 новый указатель на POINTER
    L   W [AR1,P#0.0]; // Загрузка номера DB из POINTER в ACCU1
    T   #DB_Num;      // Перенос номера DB (или 0) во временную переменную
    OPN DB [#DB_Num]; // Открытие DB
    L   D [AR1,P#2.0]; // Загрузка межзонного указателя на DT-переменную из
                        // POINTER
    LAR1 ;            // Загрузка указателя в AR1, в AR1 новый указатель на DT
                        // переменную
    L   B [AR1,P#3.0]; // Загрузка компонента "час" из DT переменной
    T   #Hour;         // Перенос в выходной параметр #Hour
    L   B [AR1,P#4.0]; // Загрузка компонента "минуты" из DT переменной
    T   #Min;          // Перенос в выходной параметр #Min
    L   B [AR1,P#5.0]; // Загрузка компонента "секунды" из DT переменной
    T   #Sec;          // Перенос в выходной параметр #Sec
    SET ;
    SAVE ;             // Установка BR-бита в 1
END_FUNCTION

ORGANIZATION_BLOCK OB1
TITLE =
VERSION : 0.1
VAR_TEMP
    OB1_TEMP: ARRAY[1..20] OF BYTE; //Стартовая информация OB1
BEGIN
NETWORK
TITLE =
    CALL FC 51 (
        IN_DT := #OB1_DATE_TIME,
        Sec   := QB 13,
        Min   := QB 12,
        Hour   := QB 8);
END_ORGANIZATION_BLOCK
```

Решение к упражнению А.2: Доступ к параметрам типа DT в FB

```
FUNCTION_BLOCK FB 63
TITLE =
VERSION : 0.1
VAR_INPUT
    IN_DT : DATE_AND_TIME ;
END_VAR
VAR_OUTPUT
    Sec : BYTE ;
    Min : BYTE ;
    Hour : BYTE ;
END_VAR
BEGIN
NETWORK
TITLE =
// В случае входного или выходного параметра сложного типа данных, значение
// комплексной переменной копируется из или в экземпляр DB.
// Для косвенного доступа создается сначала межзонный указатель, включая адрес
// смещения в AR2, который необходим в случае мультитемпляров.
//
    LAR1 P##IN_DT; // Загрузка межзонного указателя на #IN_DT без
                    // добавления смещения адреса из AR2
    TAR2 ;          // Перенос the смещения адреса в ACCU1 (смещение адреса в AR2
                    // создает макрокоманда CALL
    +AR1 ;          // Загрузка содержимого ACCU1 (смещение адреса AR2) в AR1,
                    // в AR1новый указатель на #IN_DT
    L   B [AR1,P#3.0]; // Загрузка компонента "часы" из DT-переменной
    T   #Hour;         // Перенос в выходной параметр #Hour
    L   B [AR1,P#4.0]; // Загрузка компонента "минуты" из DT-переменной
    T   #Min;          // Перенос в выходной параметр #Min
    L   B [AR1,P#5.0]; // Загрузка компонента "секунды" из DT-переменной
    T   #Sec;          // Перенос в выходной параметр #Sec
    SET ;
    SAVE ;             // Установка BR-бита в 1
END_FUNCTION_BLOCK

FUNCTION_BLOCK FB 83
TITLE =
VERSION : 0.1

VAR_INPUT
    IN_DT : DATE_AND_TIME ;
END_VAR
VAR
    DUMMY : ARRAY [1 .. 100] OF INT ;
    FB63_DT : FB 63;
END_VAR
BEGIN
NETWORK
TITLE =

    CALL #FB63_DT (
        IN_DT      := #IN_DT,
        Sec        := QB 13,
        Min        := QB 12,
        Hour       := QB 8);

END_FUNCTION_BLOCK
```

Решение к упражнению А.3: Доступ к I/O параметрам в FB (Part 1)

```
FUNCTION_BLOCK FB 64
TITLE =
VERSION : 0.1
VAR_OUTPUT
    Sec : BYTE ;
    Min : BYTE ;
    Hour : BYTE ;
END_VAR
VAR_IN_OUT
    IO_DT : DATE_AND_TIME ;
END_VAR
VAR_TEMP
    DB_Num : WORD ;
END_VAR
BEGIN
NETWORK
TITLE =

// В случае in_out параметра сложного типа данных, значение комплексной
// переменной не копируется в экземпляр DB, а вместо этого в экземпляр DB
// записывается POINTER на фактический операнд. Для косвенного доступа создается
// сначала создается межзонный указатель на этот POINTER, включая адрес смещают
// AR2, который необходим для мультиэкземпляров. После того, как это сделано, доступ к
// фактическим операндам осуществляется обычным способом.

LAR1 P##IO_DT; // Загрузка межзонного указателя в POINTER

// без смещения адреса
TAR2 ; // Перенос смещения адреса из AR2 в ACCU1
+AR1 ; // Добавить смещения адреса к AR1, в AR1 новый указатель
// на POINTER

L W [AR1,P#0.0]; // Загрузка номера DB из POINTER в ACCU1
T #DB_Num; // Перенос номера DB (или 0) во временную переменную
OPN DB [#DB_Num]; // Открытие DB
L D [AR1,P#2.0]; // Загрузка межзонного указателя на DT- перем. из POINTER
LAR1 ; // Загрузка указателя в AR1, в AR1 указатель на DT- перем.
L B [AR1,P#3.0]; // Загрузка компонента "часы" из DT- перем.
T #Hour; // Перенос в выходной параметр #Hour
L B [AR1,P#4.0]; // Загрузка компонента "минуты" из DT- перем.
T #Min; // Перенос в выходной параметр #Min
L B [AR1,P#5.0]; // Загрузка компонента "секунды" из DT- перем.
T #Sec; // Перенос в выходной параметр #Sec
SET ;
SAVE ; // Установка BR-бита в 1

END_FUNCTION_BLOCK
```

Решение к упражнению А.3: Доступ к I/O параметрам в FB (Part 2)

```
FUNCTION_BLOCK FB 84
TITLE =
VERSION : 0.1
VAR_INPUT
    IN_DT : DATE_AND_TIME ;
END_VAR
VAR
    DUMMY : ARRAY [1 .. 100] OF INT ;
    FB64_DT : FB 64;
END_VAR
VAR_TEMP
    DT_TEMP : DATE_AND_TIME ;
    I_Ret_VAL : INT ;
END_VAR
BEGIN
NETWORK
TITLE =

    CALL SFC 20 (
        SRCBLK    := #IN_DT,
        RET_VAL    := #I_Ret_VAL,
        DSTBLK     := #DT_TEMP);

    CALL #FB64_DT (
        Sec        := QB    13,
        Min        := QB    12,
        Hour       := QB    8,
        IO_DT      := #DT_TEMP);

END_FUNCTION_BLOCK

ORGANIZATION_BLOCK OB1
TITLE =
VERSION : 0.1
VAR_TEMP
    OB1_EV_CLASS : BYTE ; //Биты 0-3 = 1 (приход. событ.), биты 4-7 = 1 (класс события 1)
    OB1_SCAN_1 : BYTE ;    //1 (Первый свободный цикл после полного рестарта),
                           // 3 (Не первый свободный цикл)
    OB1_PRIORITY : BYTE ;  //1 (Приоритет)
    OB1_OB_NUMBR : BYTE ;  //1 (Организационный блок 1, OB1)
    OB1_RESERVED_1 : BYTE ; //Зарезервировано для системы
    OB1_RESERVED_2 : BYTE ; // Зарезервировано для системы
    OB1_PREV_CYCLE : INT ;  //Время цикла предыд. OB1 (ms)
    OB1_MIN_CYCLE : INT ;   //Минимальное время цикла OB1 (ms)
    OB1_MAX_CYCLE : INT ;   //Максимальное время цикла OB1 (ms)
    OB1_DATE_TIME : DATE_AND_TIME ; //Дата и время запуска OB1
END_VAR
BEGIN
NETWORK
TITLE =

    CALL FB 84 , DB 84 (
        IN_DT := #OB1_DATE_TIME);

END_ORGANIZATION_BLOCK
```