

Структура и элементы функционального плана

Обзор главы

6

В разделе	Вы найдете	на стр.
6.1	Элементы и блоки	6–2
6.2	Булева логика и таблицы истинности	6–8
6.3	Значение регистров CPU в командах	6–11

6.1 Элементы и блоки

Команды FUP

Команды FUP состоят из элементов и блоков, графически объединяемых в сегменты. Элементы и блоки можно разделить на следующие группы:

Команды как элементы

STEP 7 представляет часть команд FUP в виде отдельных элементов, которые не нуждаются ни в адресах, ни в параметрах (см. рис. 6-1)

Блок	Описание	Раздел в этом руководстве
	Отрицание двоичного ввода	8.7

Рис. 6-1. Команда FUP как элемент без адреса и параметров

Команда как блок с адресом

STEP 7 представляет некоторые из команд FUP в виде блоков, для которых Вы должны указать адрес (см. рис. 6-2). За более подробной информацией об адресации обращайтесь к главе 7.

Блок	Описание	Раздел в этом руководстве
	Присвоить	8.8

Рис. 6-2. Команда FUP как блок с адресом

Команда как блок с адресом и значением

STEP 7 представляет некоторые из команд FUP в виде блоков, для которых Вы должны указать адрес и значение (например, значение таймера или счетчика, см. рис. 6-3)

За более подробной информацией об адресации обращайтесь к главе 7.

Команда как блок с параметрами

Блок	Описание	Раздел в этом руководстве
	Таймер с задержкой включения с запоминанием	8.19

Рис. 6-3. Команда FUP как блок с адресом и значением

Команда как блок с параметрами

STEP 7 представляет некоторые из команд FUP в виде блоков с входами и выходами (см. рис. 6-4). Входы расположены слева от блока, а выходы справа. Вы указываете входные параметры и некоторые из выходных параметров. Большинство из выходных параметров предоставляется программным пакетом STEP 7. Для назначения параметров необходимо использовать специальную запись типов данных.

Параметры EN (деблокировать вход) и ENO (деблокировать выход) описаны ниже. За дальнейшей информацией о входных и выходных параметрах обращайтесь к описаниям отдельных команд в данном руководстве.

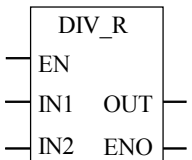
Блок	Описание	Раздел в этом руководстве
	Деление вещественных чисел	12.5

Рис. 6-4. Команда FUP как блок с входами и выходами

Параметры EN и ENO

Если параметр EN (деблокировать вход) блока FUP активизирован, то блок выполняет определенную функцию. Если эта функция выполняется блоком без ошибок, то активизируется параметр ENO (деблокировать выход). Параметры EN и ENO блока FUP относятся к типу данных BOOL и могут быть размещены в областях памяти I, Q, M, D или L (см. табл. 6-1 и 6-2).

Как действуют EN и ENO, описано ниже:

- Если EN не активизирован (его сигнальное состояние равно 0), то блоку не выполняет свою функцию и ENO не активизируется (его сигнальное состояние тоже равно 0).
- Если EN активизирован (его сигнальное состояние равно 1) и если блок выполняет свою функцию без ошибок, то ENO тоже активизируется (его сигнальное состояние тоже равно 1).
- Если EN активизирован (его сигнальное состояние равно 1) и если при исполнении блоком функции возникает ошибка, то ENO не активизируется (его сигнальное состояние остается равным 0).

Области памяти и их функции

Большинство адресов в FUP относятся к областям памяти. Следующая таблица показывает типы областей и их функции.

Таблица 6–1. Области памяти и их функции

Название области	Функция области памяти	Доступ к области через	
		единицы следующих размеров:	Сокращение
Отображение процесса на входах	В начале цикла операционная система читает входы с процесса и записывает значения в этой области. Программа использует эти значения при циклической обработке.	Входной бит Входной байт Входное слово Двойное входное слово	I IB IW ID
Отображение процесса на выходах	Во время цикла программа рассчитывает выходные значения и сохраняет их в этой области. В конце цикла операционная система считывает рассчитанные выходные значения из этой области и пересылает их на выходы к процессу.	Выходной бит Выходной байт Выходное слово Двойное выходное слово	Q QB QW QD
Меркеры	Эта область предоставляет место в памяти для промежуточных результатов, рассчитанных программой.	Меркерный бит Меркерный байт Меркерное слово Двойное меркерное слово	M MB MW MD
Периферийная область: внеш. входы	Эта область дает Вашей программе возможность прямого доступа к модулям ввода и вывода (периферийные входы и выходы).	Периферийный входной байт	PIB
Периферийная область: внеш. выходы		Периферийное входное слово	PIW
		Периферийное двойное входное слово	PID
		Периферийный выходной байт	PQB
		Периферийное выходное слово	PQW
		Периферийное двойное выходное слово	PQD
Таймеры	Таймеры - это функциональные элементы в FUP. Эта область предоставляет место в памяти для таймерных ячеек. В этой области датчик импульсов времени обращается к таймерным ячейкам для их актуализации путем уменьшения значения времени. Таймерные операции обращаются к этим ячейкам.	Таймер (Т)	T
Счетчики	Счетчики - это функциональные элементы в FUP. Эта область предоставляет место в памяти для счетчиков. К ней обращаются операции счета.	Счетчик (С)	C
Блок данных	В этой области содержатся данные, к которым можно обратиться из любого блока. Если Вам нужно одновременно открыть два различных блока данных, то Вы можете один из них открыть командой "OPN DB", а другой - командой "OPN DI". Нотация операндов, напр., L DBWi и L DIWi, определяет, к какому блоку данных производится обращение. Хотя командой "OPN DI" Вы можете обратиться к любому блоку данных, эта команда применяется главным образом для открытия экземпляров блоков данных, поставленных в соответствие функциональным блокам (FB) и системным функциональным блокам (SFB). Дальнейшую информацию о FB и SFB Вы найдете в Руководстве по программированию /234/ и в Руководстве пользователя /231/.	Блок данных, открытый командой "OPN DB": Бит данных Байт данных Слово данных Двойное слово данных Блок данных, открытый командой "OPN DI": Бит данных Байт данных Слово данных Двойное слово данных	DBX DBB DBW DBD DIX DIB DIW DID
Локальные данные	Эта область содержит временные данные логического блока(FB или FC). Этот тип данных называется также динамическими локальными данными. Эта область используется как буфер. Когда логический блок закрывается, эти данные теряются. Эти данные расположены в стеке локальных данных (L-stack).	Врем. локальный бит Врем. локальный байт Врем. локальное слово Врем. локальное двойное слово	L LB LW LD

В таблице 6–2 перечислены максимальные диапазоны адресов для различных областей памяти. За более подробной информацией о диапазонах адресов на Вашем CPU обращайтесь к соответствующему руководству /70/ или /101/.

Таблица 6–2. Области памяти и их диапазоны адресов

Название области	Доступ к области через		Максимальный диапазон адресов
	единицы следующих размеров:	Сокр.	
Отображение процесса на входах	Входной бит	I	0 от 0.0 до 65 535.7
	Входной байт	IB	от 0 до 65 535
	Входное слово	IW	от 0 до 65 534
	Двойное входное слово	ID	от 0 до 65 532
Отображение процесса на выходах	Выходной бит	Q	от 0.0 до 65 535.7
	Выходной байт	QB	от 0 до 65 535
	Выходное слово	QW	от 0 до 65 534
	Двойное выходное слово	QD	от 0 до 65 532
Меркеры	Меркерный бит	M	от 0.0 до 255.7
	Меркерный байт	MB	от 0 до 255
	Меркерное слово	MW	от 0 до 254
	Двойное меркерное слово	MD	от 0 до 252
Периферийная область внешние входы	Периферийный входной байт	PIB	от 0 до 65 535
	Периферийное входное слово	PIW	от 0 до 65 534
Периферийная область внешние выходы	Периферийное двойное входное слово	PID	от 0 до 65 532
	Периферийный выходной байт	PQB	от 0 до 65 535
	Периферийное выходное слово	PQW	от 0 до 65 534
	Периферийное двойное выходное слово	PQD	от 0 до 65 532
Таймеры	Таймер	T	от 0 до 255
Счетчики	Счетчик	C	от 0 до 255
Блок данных	Блок данных, открытый командой DB [OPN]		от 0.0 до 65 535.7 от 0 до 65 535 от 0 до 65 534 от 0 до 65 532
	Бит данных в блоке данных	DBX DBB DBW DBD	
	Байт данных		
	Слово данных		
	Двойное слово данных		
	Блок данных, открытый командой DI [OPN]		от 0.0 до 65 535.7 от 0 до 65 535 от 0 до 65 534 от 0 до 65 532
	Бит данных в экземпляре DB	DIX DIB DIW DID	
	Байт данных		
	Слово данных		
	Двойное слово данных		
Локальные данные 1)	Времен. бит локальных данных	L	от 0.0 до 65 535.7
	Времен. байт локальных данных	LB	от 0 до 65 535
	Времен. слово локальных данных	LW	от 0 до 65 534
	Времен. двойное слово локальных данных	LD	от 0 до 65 532

1) С командами FUP Вы можете использовать адрес в области памяти L только тогда, когда Вы описали его как VAR_TEMP в таблице описания переменных.

6.2 Булева логика и таблицы истинности

Булева логика

Язык программирования FUP основан на двоичной логике булевой алгебры, в которой переменные могут принимать значения “истина” (1) или “ложь” (0).

Каждая логическая команда проверяет состояние сигнала переменной на равенство 1 (истина, удовлетворяется) или 0 (ложь, не удовлетворяется) и генерирует результат. Затем команда или сохраняет результат, или использует его для выполнения булевой логической операции. Результат логической операции называется RLO (в мнемонике SIMATIC - VKE).

Для представления логики используются логические блоки, известные из булевой алгебры.

Результаты логических операций для всех возможных комбинаций логических переменных перечисляются в таблицах истинности.

Правила булевой логики иллюстрируются ниже на примере логических операций И, ИЛИ и исключающее ИЛИ.

Логическая операция И

В логической операции И опрашиваются сигнальные состояния двух или более указанных адресов. Если сигнальные состояния всех адресов равны 1, то условие удовлетворяется, и команда выдает результат 1. Если сигнальное состояние хотя бы одного адреса равно 0, то условие не удовлетворяется, и операция выдает результат 0.

Рис. 6–5 иллюстрирует логическую операцию И в языке программирования FUP.

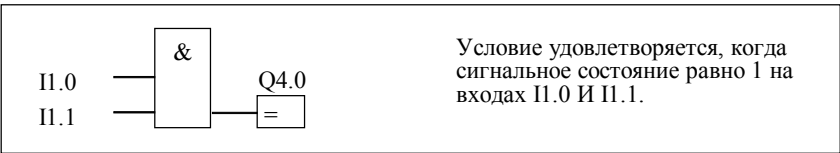


Рис. 6-5. Логическая операция И в FUP

Возможные результаты логической операции И могут быть представлены в таблице истинности (табл. 6–3). Здесь 1 означает “удовлетворяется”, а 0 означает “не удовлетворяется”.

Таблица 6–3. Таблица истинности для И

Если результат опроса сигнального состояния по адресу I1.0 равен	и результат опроса сигнального состояния по адресу I1.1 равен	то результат логической операции имеет следующее значение:
1	1	1
0	1	0
1	0	0
0	0	0

Логическая операция ИЛИ

В логической операции ИЛИ опрашиваются сигнальные состояния двух или более указанных адресов. Если сигнальное состояние хотя бы одного из адресов равно 1, то условие удовлетворяется, и команда дает результат 1. Если сигнальные состояния всех адресов равны 0, то условие не удовлетворяется, и операция выдает результат 0.

Рис. 6–6 иллюстрирует логическую операцию ИЛИ в языке программирования FUP.

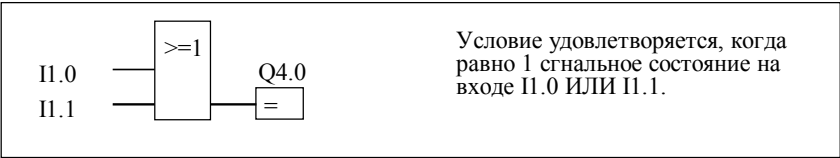


Рис. 6-6. Логическая операция ИЛИ в FUP

Возможные результаты логической операции ИЛИ могут быть представлены в таблице истинности (табл. 6–4). Здесь 1 означает “удовлетворяется”, а 0 означает “не удовлетворяется”.

Таблица 6–4. Таблица истинности для ИЛИ

Если результат опроса сигнального состояния по адресу П.0 равен	и результат опроса сигнального состояния по адресу П.1 равен	то результат логической операции имеет следующее значение:
1	0	1
0	1	1
1	1	1
0	0	0

Логическая операция исключающее ИЛИ

В логической операции исключающее ИЛИ опрашиваются сигнальные состояния двух или более указанных адресов. Если сигнальное состояние ровно одного из адресов равно 1, то условие удовлетворяется, и команда дает результат 1. Если сигнальные состояния всех адресов равны 0 или 1, то условие не удовлетворяется, и операция выдает результат 0.

Рис. 6–7 иллюстрирует логическую операцию исключающее ИЛИ в языке программирования FUP.

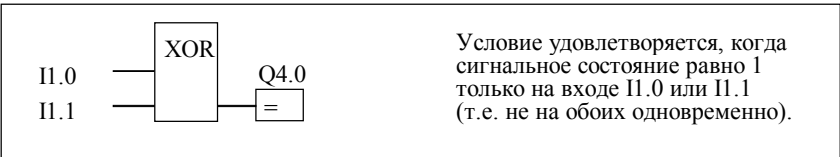


Рис. 6-7. Логическая операция исключающее ИЛИ в FUP

Возможные результаты логической операции ИЛИ могут быть представлены в таблице истинности (табл. 6–5). Здесь 1 означает “удовлетворяется”, а 0 означает “не удовлетворяется”.

Таблица 6–5. Таблица истинности для исключающего ИЛИ

Если результат опроса сигнального состояния по адресу I1.0 равен	и результат опроса сигнального состояния по адресу I1.1 равен	то результат логической операции имеет следующее значение:
1	0	1
0	1	1
1	1	0
0	0	0

6.3 Значение регистров CPU в командах

Объяснение

Регистры помогают CPU выполнять логические и арифметические операции, а также операции сдвига и преобразования. Эти регистры описаны ниже.

Аккумуляторы

Аккумуляторы - это регистры общего назначения, которые используются для обработки байтов, слов и двойных слов. Аккумуляторы имеют длину 32 бита.

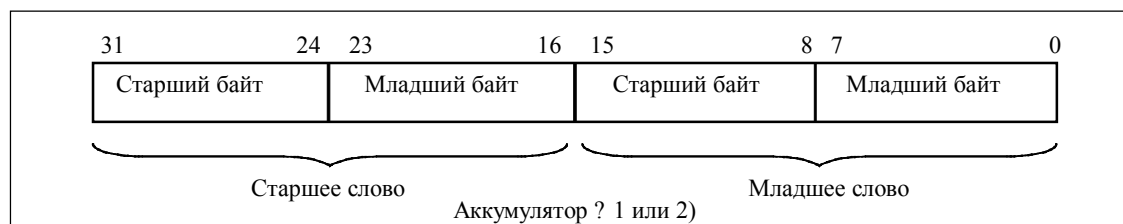


Рис. 6-8. Области аккумулятора

Слово состояния

Слово состояния содержит биты, к которым Вы можете обращаться в операндах битовых логических операций. Следующие разделы объясняют значения битов с 0 по 8.

$2^{15} \dots$	$\dots 2^9$	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
		BR	CC1	CC0	OV	OS	OR	STA	RLO	FC

Рис. 6-9. Структура слова состояния

Изменение битов в слове состояния

Значение	Смысл
0	устанавливает состояние сигнала в 0
1	устанавливает состояние сигнала в 1
x	изменяет состояние
-	состояние остается неизменным

Первичный опрос

Бит 0 слова состояния называется битом первичного опроса (бит FC, см. рис. 6–9). В начале сегмента FUP сигнальное состояние бита FC всегда равно 0, если только предыдущий сегмент не был завершен блоком SAVE.

Каждая логическая операция опрашивает сигнальное состояние бита FC, а также сигнальное состояние адреса, к которому обращается команда. Сигнальное состояние бита FC определяет последовательность выполнения цепи логических сопряжений. Если бит FC равен 0 (в начале сегмента FUP), то команда сохраняет результат в бите результата логической операции (RLO) слова состояния и устанавливает бит FC в 1. Это называется первичным опросом. Результат “1” или “0”, сохраняемый после первичного опроса в бите RLO, называют результатом первичного опроса.

Если сигнальное состояние бита FC равно 1, то операция логически сопрягает результат опроса состояния сигнала по обрабатываемому адресу с RLO, сформированным после первичного опроса, и сохраняет результат в бите RLO.

Логическая цепочка, составленная из команд FUP, всегда заканчивается операцией над выходом (например, установить выход, сбросить выход, присвоить значение) или командой перехода в зависимости от результата логической операции (RLO). Эти команды сбрасывают бит FC в 0.

Результат логической операции

Бит 1 слова состояния называется битом результата логической операции (бит RLO, см. рис. 6–9). Этот бит сохраняет результат цепи логических операций или операций сравнения. Сигнальное состояние бита RLO дает информацию о потоке сигнала.

Первая команда в сегменте FUP опрашивает сигнальное состояние адреса и выдает результат 1 или 0. Эта команда сохраняет результат опроса в бите RLO. Вторая команда в цепи логических операций также опрашивает сигнальное состояние некоторого адреса и формирует результат. Теперь команда комбинирует этот результат со значением бита RLO слова состояния в соответствии с правилами булевой логики (см. выше Первичный опрос). Этот результат логической операции сохраняется в бите RLO слова состояния, заменяя предыдущее значение в бите RLO. Каждая следующая команда в цепи логических операций комбинирует два значения: результат опроса сигнала по указанному адресу и текущее значение RLO.

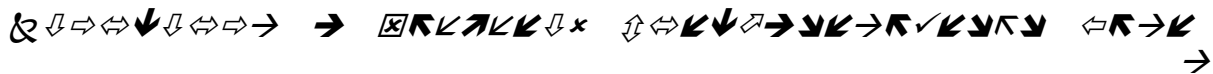
Вы можете, например, присвоить RLO во время первичного опроса состояние некоторого меркерного бита или запустить команду перехода.

Бит состояния

Бит 2 слова состояния называется битом состояния (бит STA, см. рис. 6–9). Бит состояния сохраняет значение того бита, к которому производится обращение. Состояние логической операции, выполняющей чтение из памяти, всегда совпадает со значением бита, который опрашивается этой командой (бита, над которым выполняется логическая операция). Состояние битовой операции, которая записывает в память (установить выход, сбросить выход или присвоить), совпадает со значением бита, в который эта команда осуществляет запись. Если запись не производится, то значение совпадает со значением бита, к которому обращается команда. Бит состояния не имеет значения для битовых операций, которые не обращаются к памяти. Эти команды устанавливают бит состояния в 1 (STA=1). Бит состояния не опрашивается командами. Он интерпретируется только во время тестирования программы (статус программы).

Бит OR

Бит 3 слова состояния называется битом OR (см. рис. 6–9). Бит OR необходим для выполнения логической операции И перед ИЛИ. Логическая операция И может содержать команды И и И-НЕ. Бит OR показывает этим командам, что ранее выполненная логическая операция И дала значение 1, так что результат логической операции ИЛИ уже определен. Любые другие команды, обрабатывающие биты, сбрасывают бит OR.



Бит переполнения

Бит 5 слова состояния называется битом переполнения (бит OV, см. рис. 6–9). Бит OV указывает на наличие ошибки. Он устанавливается арифметической операцией или операцией сравнения чисел с плавающей точкой после возникновения ошибки (переполнение, недопустимая команда, недопустимое число с плавающей точкой). Этот бит устанавливается или сбрасывается в соответствии с результатом выполнения арифметической операции или операции сравнения (ошибка).

Бит переполнения с сохранением

Бит 4 слова состояния называется битом переполнения с сохранением (бит OS, см. рис. 6–9). Бит OS устанавливается вместе с битом OV, когда возникает ошибка. Так как бит OS не меняется, когда арифметические операции выполняются без ошибок (в отличие от бита OV), он показывает, произошла или нет ошибка в ранее выполненных операциях. Бит OS сбрасывают следующие команды: JOS (перейти, если бит переполнения с сохранением = 1, должна программироваться на AWL), вызовы блоков и команды конца блока.

CC1 и CC0

Биты 7 и 6 слова состояния называются кодом условия 1 и кодом условия 0 (CC1 и CC0, см. рис. 6–9). Биты CC1 и CC0 предоставляют информацию о следующих результатах или битах:

- результат арифметической операции
- результат операции сравнения
- результат цифровой операции
- биты, которые были выдвинуты из операнда операций сдвига или циклического сдвига.

В таблицах 6–6 и 6–7 представлены значения битов CC1 и CC0 после выполнения программой определенных операций.

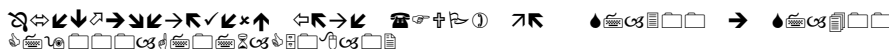
Таблица 6–6. CC1 и CC0 после арифметических операций, без переполнения

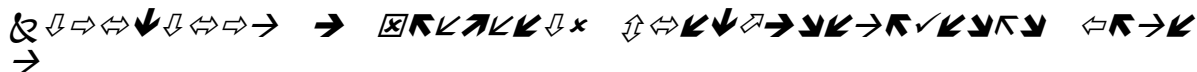
CC1	CC0	Объяснение
0	0	Результат = 0
0	1	Результат < 0
1	0	Результат > 0

Таблица 6–7. CC1 и CC0 после арифметических операций над целыми числами, с переполнением

CC1	CC0	Объяснение
0	0	Переполнение в отрицательной области при сложении целых и двойных целых чисел
0	1	Переполнение в отрицательной области при умножении целых и двойных целых чисел Переполнение в положительной области при сложении целых чисел, вычитании целых чисел, сложении двойных целых чисел, вычитании двойных целых чисел, получении дополнения до 2 целого числа и получении дополнения до 2 двойного целого числа
1	0	Переполнение в положительной области при умножении целых чисел и двойных целых чисел, делении целых чисел и двойных целых чисел Переполнение в отрицательной области при сложении целых чисел, вычитании целых чисел, сложении двойных целых чисел, вычитании двойных целых чисел
1	1	Деление на 0 при делении целых чисел, при делении двойных целых чисел и при получении остатка от деления двойных целых чисел.

Таблица 6–8. CC1 и CC0 после арифметических операций над числами с плавающей точкой, с переполнением





CC1	CC0	Объяснение
0	0	Ступенчатая потеря значимости
0	1	Переполнение отрицательной области
1	0	Переполнение положительной области
1	1	Недопустимое число с плавающей точкой

Таблица 6–9. CC1 и CC0 после операций сравнения

CC1	CC0	Объяснение
0	0	IN2 = IN1
0	1	IN2 < IN1
1	0	IN2 > IN1
1	1	IN1 или IN2 недопустимое число с плавающей точкой

Таблица 6–10. CC1 и CC0 после операций сдвига и циклического сдвига

CC1	CC0	Объяснение
		Последний сдвинутый бит = 0
		Последний сдвинутый бит = 1

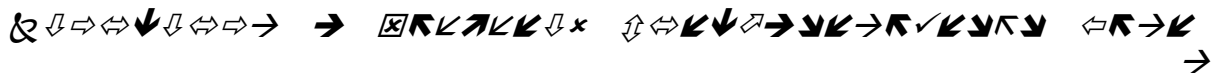
Таблица 6–11. CC1 и CC0 после поразрядных логических операций со словами

CC1	CC0	Объяснение
0	0	Результат = 0
1	0	Результат <> 0

Бит двоичного результата

Бит 8 слова состояния называется битом двоичного результата (бит BR, см. рис. 6–9). Бит BR образует связь между обработкой битов и слов. Этот бит позволяет эффективно интерпретировать результат операции над словами как двоичный результат и включать этот результат в цепочку логических операций. Бит BR представляет собой бит внутренней памяти, в котором может быть сохранен результат логической операции RLO перед выполнением операции над словами, изменяющей RLO, так что старое значение RLO снова доступно после операции, когда возобновляется прерванная последовательность битовых операций.

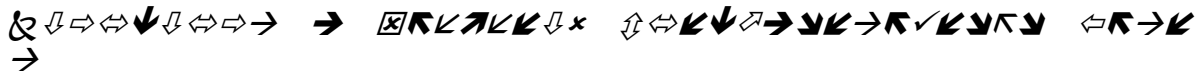
С помощью бита BR Вы можете, например, запрограммировать функциональный блок (FB) или функцию (FC) на языке AWL и вызвать FB или FC в FUP.



Если Вы хотите написать функциональный блок или функцию, которые Вы хотите вызвать в FUP, независимо от того, пишете Вы FB или FC на AWL или на FUP, Вы должны принять во внимание бит BR. Бит BR соответствует параметру ENO (деблокировать выход) блока FUP. RLO сохраняется в бите BR с помощью команды SAVE (в AWL) или блока FUP SAVE в соответствии со следующими критериями:

- Сохранить RLO, равный 1, в бите BR, когда FB или FC обрабатывается без ошибок.
- Сохранить RLO, равный 0, в бите BR, если при обработке FB или FC возникает ошибка.

Запрограммируйте эти команды в конце FB или FC, так чтобы они были последними командами, выполняемыми в блоке.



Предупреждение

Бит BR может быть сброшен в 0 непреднамеренно.

Если Вы пишете FB или FC в FUP и не обрабатываете бит BR, как это описано выше, FB или FC может переписать бит BR другого FB или другой FC.

Во избежание этой ошибки сохраняйте RLO в конце FB или FC, как описано выше.

Значение EN/ENO

Параметры EN (деблокировать вход) и ENO (деблокировать выход) блока FUP функционируют следующим образом:

- Если EN не активизирован (т.е. состояние сигнала равно "0"), то блок свою функцию не выполняет, и ENO не активизируется (т.е. состояние сигнала тоже равно "0").
- Если EN активизирован (т.е. состояние сигнала равно "1") и соответствующий блок выполняет свою функцию без ошибок, то ENO тоже активизируется (т.е. состояние сигнала тоже равно "1").
- Если EN активизирован (т.е. состояние сигнала равно "1") и при обработке функции возникает ошибка, то ENO не активизируется (т.е. состояние сигнала равно "0").

Если Вы в своей программе вызываете системный функциональный блок (SFB) или системную функцию (SFC), то SFB или SFC указывает через сигнальное состояние бита BR, выполнил ли CPU эту функцию без ошибок или с ошибкой:

- Если при исполнении возникает ошибка, то бит BR устанавливается в 0.
- Если функция была обработана без ошибок, то бит BR равен 1.