Функции для обработки чисел

Функции для обработки чисел используются для обработки численных значений преимущественно типов INT, DINT и REAL, что значительно расширяет функциональные возможности PLC. Здесь будут рассмотрены функции для обработки чисел, используемые в STL-программах. В языке программирования SCL функции сравнения, логические функции с данными формата Word и арифметические функции выполняются с помощью операторов (см. раздел 27.4 "Выражения"); остальные функции для обработки чисел включены в язык SCL в виде стандартных функций (см. разделы 30.3 "Математические функции", 30.4 "Функции сдвига и ротации", 30.5 "Функции преобразования").

Функции сравнения формируют двоичный результат сравнения двух величин. Функции сравнения могут обрабатывать данные типов INT, DINT и REAL.

Арифметические функции позволяют выполнять в программе вычисления. Все базовые арифметические операции могут быть выполнены с данными типов INT, DINT и REAL.

Математические функции расширяют собой вычислительные возможности в программе. Математические функции добавляют к базовым арифметическим функциям такое расширение как тригонометрические функции.

Функции преобразования позволяют до и после выполнения вычислений приводить численные значения к требуемому типу данных.

Функции сдвига позволяют смещать содержимое аккумулятора влево или вправо. Функции сдвига позволяют также проверить бит, смещенный последним.

Логические функции с данными формата Word используются для маскирования численных значений для проверки отдельных битов и задания для них значений "0" или "1".

Функции для обработки чисел работают, главным образом, со значениями в блоках данных. Это могут быть блоки глобальных данных или экземплярные блоки данных, если используются статические локальные данные. В разделе 18.2 "Функции для блоков данных" показано использование блоков данных и варианты адресации данных.

За исключением аккумуляторов для хранения временных результатов очень удобны временные локальные данные.

9 Функции сравнения

Операции сравнения величин на предмет их равенства, неравенства; операции: больше чем, больше чем или равно, меньше чем, меньше чем или равно; функции сравнения в двоичных логических операциях.

10 Арифметические функции

Базовые арифметические операции; цепочки вычислений; сложение констант; декрементирование и инкрементирование.

11 Математические функции

Тригонометрические функции; обратные тригонометрические функции; возведение в квадрат; извлечение квадратного корня; экспоненты и логарифмы.

12 Функции преобразования

Преобразование данных типов INT/DINT в данные BCD типа и наоборот; преобразование данных типа DINT в данные типа REAL и наоборот с различными способами округления; нахождение обратного кода двоичного числа, инвертирование и нахождение абсолютной величины.

13 Функции сдвига

Сдвиг влево, вправо, на слово и двойное слово, сдвиг в соответствии с правилами для знаков; циклический сдвиг содержимого аккумулятора 1 влево или вправо; сдвиг и циклический сдвиг с параметрами сдвига, заданными константой или в аккумуляторе 2.

14 Логические функции с данными формата Word

Операции AND (И), OR (ИЛИ), Exclusive OR (Исключающее ИЛИ); логические операции со словом, с двойным словом, с константой или с содержимым аккумулятора 2.

9 Функции сравнения

Функции сравнения обеспечивают выполнение операции сравнения двух численных значений, одно из которых находится в аккумуляторе accumulator 1, а второе находится в аккумуляторе accumulator 2. После выполнения операции сравнения функции сравнения устанавливают RLO (результат логической операции) и биты состояния CC0 и CC1. Этот результат может в дальнейшем быть использован в двоичных логических операциях, в операциях с памятью или в операторах перехода. В таблице 9.1 Вы найдете обзор доступных пользователю функций сравнения.

Таблица 9.1
Общее представление функций сравнения

Операция сравнения	Типы данных		
	INT	DINT	REAL
Равно	==I	==D	==R
Не равно	<>I	<>D	<>R
Больше чем	>I	>D	>R
Больше чем или равно	>=I	>=D	>=R
Меньше чем	<I	<D	<R
Меньше чем или равно	<=I	<=D	<=R

В главе 15 "Биты состояния" будет показано, как функции сравнения устанавливают биты состояния CC0 и CC1.

В этом разделе будут рассмотрены функции для обработки чисел, используемые в языке программирования STL. Тогда как в языке программирования SCL выполнение функций сравнения обеспечивается с помощью соответствующих выражений (см. раздел 27.4.2 "Операторы сравнения").

Примеры, рассматриваемые в данной главе, Вы можете найти на прилагаемой дискете в библиотеке STL_Book library в разделе "Digital Functions" ("Функции для обработки чисел") в функциональном блоке FB 109 или в исходном файле Chap_9.

9.1 Общее представление функций сравнения

Вы можете программировать функции сравнения в соответствии со следующей общей схемой:

```
Загрузка адреса Address1;
Загрузка адреса Address2;
Функция сравнения;
Присвоение результата Result;
```

При выполнении первой операции загрузки Load из первого адреса число помещается в аккумулятор accumulator 1. При загрузке из второго адреса содержимое аккумулятора accumulator 1 перемещается в аккумулятор accumulator 2 (см. раздел 6.2 "Операции загрузки"). Теперь с содержимым двух аккумуляторов можно выполнить операции с помощью функций сравнения.

Функции сравнения возвращают двоичный результат (тип BOOL), который может быть назначен двоичному адресу или может быть использован в какой-либо другой двоичной проверке (опросе).

Функции сравнения не изменяют содержимого аккумуляторов. Они всегда выполняются вне всякой связи с какими-либо условиями.

В таблице 9.2 показаны примеры для различных типов данных. Инструкции сравнения выполняются в соответствии с определенными параметрами независимо от содержимого аккумуляторов.

Таблица 9.2
Примеры функций сравнения

Применение функции сравнения для данных типа INT	Меркер M99.0 сбрасывается, если значение в слове MW 92 равно 120, иначе меркер не сбрасывается	`L  MW 92;`
		`L  120;`
		`==I;`
		`R  M 99.0;`

Применение функции сравнения для данных типа DINT	Переменная "CompResult" в блоке данных "Global_DB" устанавливается, если переменная "CompVal1" меньше чем "CompVal2", иначе она сбрасывается	`L "Global_DB"."CompVal1";`
		`L "Global_DB"."CompVal2";`
		`<D;`
		`= "Global_DB"."CompResult";`

Применение функции сравнения для данных типа REAL	Если переменная #Actval больше чем или равна переменной #Calibra, то переменная #Recali устанавливается, иначе - не устанавливается.	`L  #Actval;`
		`L  #Calibra;`
		`>=R;`
		`S  #Recali;`

В случае, когда данные имеют тип INT, CPU сравнивает только расположенные справа слова (младшие слова) в аккумуляторах; содержимое слов, расположенных слева (старшие слова), в расчет не берется.

В случае, когда данные имеют тип REAL, выполняется проверка для определения корректности (в соответствии с типом данных) содержимого аккумуляторов. Если результат этой проверки отрицателен, то CPU сбрасывает RLO в состояние "0", а биты состояния CC0, CC1, OV и OS устанавливает в состояние "1".

9.2 Описание функций сравнения

Проверка на равенство чисел

Инструкция для сравнения чисел с целью определения их равенства интерпретирует содержимое аккумуляторов в соответствии с типом данных, указанным в инструкции. Инструкция призвана выявить равенство двух числовых величин. После выполнения инструкции для RLO устанавливается значение "1", в следующих случаях:

- для данных типа INT
  если содержимое младшего слова аккумулятора accumulator 2 равно содержимому младшего слова аккумулятора accumulator 1,

- для данных типа DINT
  если содержимое аккумулятора accumulator 2 равно содержимому аккумулятора accumulator 1,

- для данных типа REAL
  если содержимое аккумулятора accumulator 2 равно содержимому аккумулятора accumulator 1 при условии, что оба аккумулятора содержат корректные действительные (REAL) числа.

Если два числа формата REAL равны, но некорректны, условие равенства ("equal to") не выполняется (RLO = "0").

Проверка на неравенство чисел

Инструкция для сравнения чисел с целью определения их неравенства интерпретирует содержимое аккумуляторов в соответствии с типом данных, указанным в инструкции. Инструкция призвана выявить неравенство двух числовых величин. После выполнения инструкции для RLO устанавливается значение "1", в следующих случаях:

- для данных типа INT
  если содержимое младшего слова аккумулятора accumulator 2 не равно содержимому младшего слова аккумулятора accumulator 1,

- для данных типа DINT
  если содержимое аккумулятора accumulator 2 не равно содержимому аккумулятора accumulator 1,

- для данных типа REAL
  если содержимое аккумулятора accumulator 2 не равно содержимому аккумулятора accumulator 1 при условии, что оба аккумулятора содержат корректные действительные (REAL) числа.

Если два числа формата REAL не равны, но одно из них или оба некорректны, то условие неравенства ("not equal to") не выполняется (RLO = "0").

Проверка чисел на отношение по формуле "больше"

Инструкция для сравнения с целью определения выполнения условия "больше" интерпретирует содержимое аккумуляторов в соответствии с типом данных, указанным в инструкции. Инструкция призвана выявить, не является ли число, находящееся в аккумуляторе accumulator 2, больше числа, находящегося в аккумуляторе accumulator 1. После выполнения инструкции для RLO устанавливается значение "1", в следующих случаях:

- для данных типа INT
  если содержимое младшего слова аккумулятора accumulator 2 больше чем содержимое младшего слова аккумулятора accumulator 1,

- для данных типа DINT
  если содержимое аккумулятора accumulator 2 больше чем содержимое аккумулятора accumulator 1,

- для данных типа REAL
  если содержимое аккумулятора accumulator 2 больше чем содержимое аккумулятора accumulator 1 при условии, что оба аккумулятора содержат корректные действительные (REAL) числа.

Проверка чисел на отношение по формуле "больше или равно"

Инструкция для сравнения с целью определения выполнения условия "больше или равно" интерпретирует содержимое аккумуляторов в соответствии с типом данных, указанным в инструкции. Инструкция призвана выявить, не является ли число, находящееся в аккумуляторе accumulator 2, большим или равным числу, находящемуся в аккумуляторе accumulator 1. После выполнения инструкции для RLO устанавливается значение "1", в следующих случаях:

- для данных типа INT
  если содержимое младшего слова аккумулятора accumulator 2 больше или равно содержимому младшего слова аккумулятора accumulator 1,

- для данных типа DINT

если содержимое аккумулятора accumulator 2 больше или равно содержимому аккумулятора accumulator 1,

- для данных типа REAL
  если содержимое аккумулятора accumulator 2 больше или равно содержимому аккумулятора accumulator 1 при условии, что оба аккумулятора содержат корректные действительные (REAL) числа.

Проверка чисел на отношение по формуле "меньше"

Инструкция для сравнения с целью определения выполнения условия "меньше" интерпретирует содержимое аккумуляторов в соответствии с типом данных, указанным в инструкции. Инструкция призвана выявить, не является ли число, находящееся в аккумуляторе accumulator 2, меньше числа, находящегося в аккумуляторе accumulator 1. После выполнения инструкции для RLO устанавливается значение "1", в следующих случаях:

- для данных типа INT
  если содержимое младшего слова аккумулятора accumulator 2 меньше чем содержимое младшего слова аккумулятора accumulator 1,

- для данных типа DINT
  если содержимое аккумулятора accumulator 2 меньше чем содержимое аккумулятора accumulator 1,

- для данных типа REAL
  если содержимое аккумулятора accumulator 2 меньше чем содержимое аккумулятора accumulator 1 при условии, что оба аккумулятора содержат корректные действительные (REAL) числа.

Проверка чисел на отношение по формуле "меньше или равно"

Инструкция для сравнения с целью определения выполнения условия "меньше или равно" интерпретирует содержимое аккумуляторов в соответствии с типом данных, указанным в инструкции. Инструкция призвана выявить, не является ли число, находящееся в аккумуляторе accumulator 2, меньшим или равным числу, находящемуся в аккумуляторе accumulator 1. После выполнения инструкции для RLO устанавливается значение "1", в следующих случаях:

- для данных типа INT
  если содержимое младшего слова аккумулятора accumulator 2 меньше или равно содержимому младшего слова аккумулятора accumulator 1,

- для данных типа DINT
  если содержимое аккумулятора accumulator 2 меньше или равно содержимому аккумулятора accumulator 1,

- для данных типа REAL
  если содержимое аккумулятора accumulator 2 меньше или равно содержимому аккумулятора accumulator 1 при условии, что оба аккумулятора содержат корректные действительные (REAL) числа.

9.3 Функции сравнения в логических операциях

Функции сравнения возвращают двоичный результат логической операции RLO и, следовательно, могут быть использованы в сочетании с другими двоичными функциями. Функции сравнения устанавливают бит состояния FC, например, в логических операциях функция сравнения всегда является первичным опросом ("first check").

Функция сравнения в начале логической операции

Функция сравнения в начале логической операции всегда является первичным опросом ("first check"). RLO, возвращаемый функцией сравнения может быть непосредственно использован в логических операциях (в функциях проверки).

```
L  MW 120;
L  512;
>I;
A  Input1;
=  Output1;
```

В примере выход *Output1* устанавливается, если выполняется условие сравнения функции сравнения, а вход *Input1* имеет состояние "1".

Функция сравнения внутри логической операции

Функция сравнения внутри логической операции должна быть заключена в скобки, так как функция сравнения начинает новый логический этап (первичный опрос ["first check"]).

```
O  Input2;
O( ;
L  MW 122;
L  200;
<=I;
);
O  Input3;
=  Output2;
```

В примере выход *Output2* устанавливается, если вход *Input2* или вход *Input3* имеет состояние "1" или если выполняется условие сравнения функции сравнения.

Многократное использование функции сравнения

Так как функции сравнения не изменяют содержимого аккумуляторов, возможно многократное использование этих функций при программировании на STL.

```
L  MW 124;
L  1200;
>I;
JC GREA;
==I;
JC EQUA;
```

В примере использованы две функции сравнения для одного и того же содержимого аккумуляторов. Первая операция сравнения генерирует RLO = "1", если MW 124 больше чем 1200, поэтому выполняется переход к метке GREA. Без перезагрузки аккумуляторов вторая функция сравнения выполняет проверку условия равенства и генерирует новое значение RLO.

Функция сравнения устанавливает биты состояния, исходя из соотношения сравниваемых величин, т.е. независимо от типа проверяемого условия в операции сравнения. Вы можете использовать это, делая проверку битов состояния в соответствующих функциях перехода.

Вышеприведенный пример может быть также запрограммирован следующим образом:

```
L  MW 124;
L  1200;
>I;
JP GREA;
JZ EQUA;
```

В этом примере результат выполнения функции сравнения проверяется с помощью битов состояния CC0 и CC1. Собственно условие сравнения ("больше чем") в данном случае не влияет на установку битов состояния; могут быть использованы и другие условия сравнения, например, "меньше чем". Оператор JP проверяет не является ли первая из сравниваемых величин больше, чем вторая. Оператор JZ проверяет, не являются ли обе сравниваемые величины равными.

10 Арифметические функции

Арифметические функции обеспечивают выполнение базовых арифметических операций с двумя числовыми значениями, одно из которых находится в аккумуляторе accumulator 1, а второе находится в аккумуляторе accumulator 2. Результат арифметической операции помещается в аккумулятор accumulator 1. Биты состояния CC0, CC1, OV и OS обеспечивают информацию, касающуюся выполнения вычислений и результата (см. главу 15 "Биты состояния"). В таблице 10.1 Вы найдете обзор доступных пользователю арифметических функций.

Таблица 10.1
Обзор арифметических функций

Арифметическая функция	Типы данных		
	INT	DINT	REAL
Сложение (Addition)	+I	+D	+R
Вычитание (Subtraction)	-I	-D	-R
Умножение (Multiplication)	*I	*D	*R
Деление (Division with quotient as result)	/I	/D	/R
Остаток от деления (Division with remainder as result)	-	MOD	-

В добавление к базовым арифметическим операциям с участием числа, находящегося в аккумуляторе accumulator 2, Вы также можете прибавлять константы непосредственно к содержимому аккумулятора accumulator 1 или изменять его содержимое на некоторую фиксированную величину.

В этом разделе будут рассмотрены арифметические функции, используемые в языке программирования STL. В языке программирования SCL выполнение арифметических функций обеспечивается с помощью соответствующих выражений (см. раздел 27.4.1 "Арифметические выражения").

Примеры, рассматриваемые в данной главе, Вы можете найти на прилагаемой дискете в библиотеке STL_Book library в разделе "Digital Functions" ("Функции для обработки чисел") в функциональном блоке FB 110 или в исходном файле Chap_10.

10.1 Общее представление арифметических функций

Вы можете программировать арифметические функции в соответствии со следующей общей схемой:

```
Загрузка (load) адреса Address1;
Загрузка (load) адреса Address2;
Арифметическая функция;
Передача (transfer) результата Result;
```

При выполнении первой операции загрузки Load из первого адреса число помещается в аккумулятор accumulator 1. При загрузке из второго адреса содержимое аккумулятора accumulator 1 перемещается в аккумулятор accumulator 2 (см. раздел 6.2 "Операции загрузки"). Теперь с содержимым двух аккумуляторов можно выполнить операции с помощью арифметических функций.

Результат операции сохраняется в аккумуляторе accumulator 1.

Арифметические функции выполняются в соответствии с заданными параметрами. Они всегда выполняются вне всякой связи с какими-либо условиями.

В таблице 10.2 показаны примеры для различных типов данных.

Таблица 10.2
Примеры арифметических функций

Применение арифметических функций для данных типа INT	Значение в слове MW 100 делится на 250; целый результат хранится в слове MW 102.	`L  MW 100;`
		`L  250;`
		`/I;`
		`T  MW 102;`

Применение арифметических функций для данных типа DINT	Значения переменных "ArithVal1" и "ArithVal2" складываются, и результат заносится в переменную "ArithResult". Все переменные в блоке данных "Global_DB".	`L "Global_DB"."ArithVal1";`
		`L "Global_DB"."ArithVal2";`
		`+D;`
		`T "Global_DB"."ArithResult";`

Применение арифметических функций для данных типа REAL	Переменные #Actval и #Factor перемножаются; результат умножения передается в переменную #Display.	`L  #Actval;`
		`L  #Factor;`
		`*R;`
		`T  #Display;`

В случае, когда данные имеют тип INT, арифметические операции выполняются только для расположенных справа слов (младшие слова) в аккумуляторах; слова, расположенные слева (старшие), игнорируются.

В случае, когда данные имеют тип REAL, выполняется проверка для определения корректности (в соответствии с типом данных) содержимого аккумуляторов.

В S7-300 CPU (за исключением CPU 318) выполнение арифметических функций не изменяет содержимого аккумулятора accumulator 2; в S7-400 CPU и CPU 318 содержимое аккумулятора accumulator 2 после выполнения операции переписывается содержимым аккумулятора accumulator 3. При этом содержимое аккумулятора accumulator 4 смещается (заменяет собой прежнее значение) в аккумулятор accumulator 3 (см. рис. 10.1).

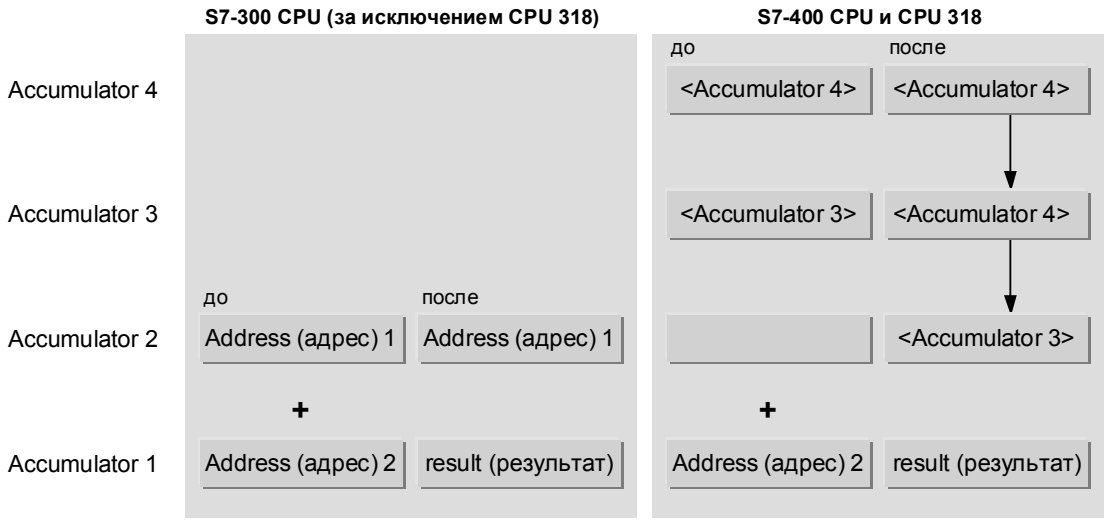

Рис. 10.1 Содержимое аккумуляторов при выполнении арифметических функций

10.2 Вычисления с данными типа INT

Сложение данных типа INT

Инструкция +I для сложения интерпретирует содержимое младших слов аккумуляторов 1 и 2 как числа целого типа (INT). Инструкция призвана выполнить сложение этих двух числовых величин и сохранить результат сложения в аккумуляторе accumulator 1.

После выполнения инструкции биты состояния CC0 и CC1 показывают, какова результирующая сумма: отрицательна, равна нулю или положительна. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

Вычитание данных типа INT

Инструкция - I для вычитания интерпретирует содержимое младших слов аккумуляторов 1 и 2 как числа целого типа (INT). Инструкция призвана выполнить вычитание числа, находящегося в аккумуляторе accumulator 1, из числа, находящегося в аккумуляторе accumulator 2 и сохранить результат

вычитания в аккумуляторе accumulator 1.

После выполнения инструкции биты состояния CC0 и CC1 показывают, какова разность: отрицательна, равна нулю или положительна. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

Умножение данных типа INT

Инструкция *I для перемножения интерпретирует содержимое младших слов аккумуляторов 1 и 2 как числа целого типа (INT). Инструкция призвана выполнить перемножение этих двух числовых величин и сохранить результат перемножения в формате числа DINT в аккумуляторе accumulator 1.

После выполнения инструкции биты состояния CC0 и CC1 показывают, каков результат перемножения чисел: отрицателен, равен нулю или положителен. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

Для последующей операции перемножения результат, находящийся в аккумуляторе accumulator 1, имеет формат числа DINT.

Деление данных типа INT

Инструкция /I для деления интерпретирует содержимое младших слов аккумуляторов 1 и 2 как числа целого типа (INT). Инструкция призвана выполнить деление числа, находящегося в аккумуляторе accumulator 2 (делимое), на число, находящееся в аккумуляторе accumulator 1 (делитель), и сохранить два результата деления: частное и остаток, оба имеющие тип INT (см. рис. 10.2).

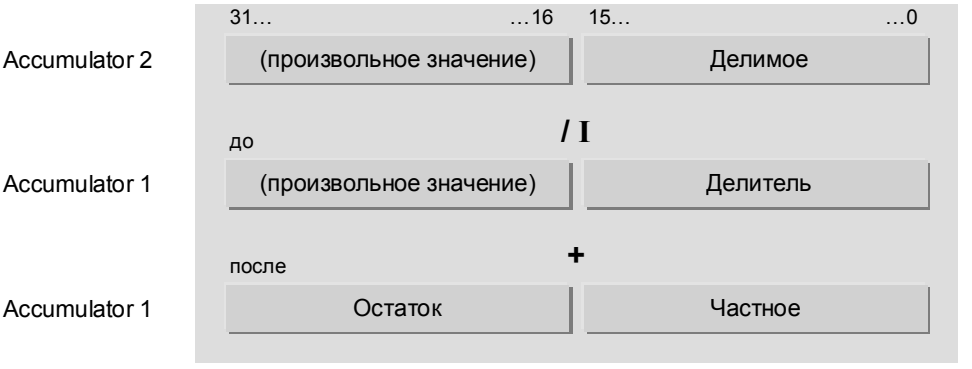

Рис. 10.2 Результат, возвращаемый арифметической функцией деления / I.

После выполнения инструкции младшее слово аккумулятора accumulator 1 содержит частное от деления. Частное является целым результатом операции деления. Частное от деления равно нулю в двух случаях: если делимое равно нулю, в то время как делитель не равен нулю, или если делимое меньше чем делитель. Частное от деления отрицательно, если делитель меньше нуля.

После выполнения инструкции старшее слово аккумулятора accumulator 1 содержит остаток от деления (не путать остаток от деления с дробной частью результата операции деления!). Остаток от деления отрицателен, если делимое меньше нуля.

После выполнения инструкции биты состояния CC0 и CC1 показывают, каково частное от деления: отрицательно, равно нулю или положительно. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

В случае деления на ноль частное от деления и остаток возвращаются с нулевыми значениями, а биты состояния CC0, CC1, OV и OS устанавливаются в состояние "1".

10.3 Вычисления с данными типа DINT

Сложение данных типа DINT

Инструкция +D для сложения интерпретирует содержимое аккумуляторов 1 и 2 как числа целого типа (DINT). Инструкция призвана выполнить сложение этих двух числовых величин и сохранить результат сложения в аккумуляторе accumulator 1.

После выполнения инструкции биты состояния CC0 и CC1 показывают, какова результирующая сумма: отрицательна, равна нулю или положительна. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

Вычитание данных типа DINT

Инструкция -D для вычитания интерпретирует содержимое аккумуляторов 1 и 2 как числа целого типа (DINT). Инструкция призвана выполнить вычитание числа, находящегося в аккумуляторе accumulator 1, из числа, находящегося в аккумуляторе accumulator 2 и сохранить результат вычитания (разность) в аккумуляторе accumulator 1.

После выполнения инструкции биты состояния CC0 и CC1 показывают, какова разность: отрицательна, равна нулю или положительна. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

Умножение данных типа DINT

Инструкция *D для перемножения интерпретирует содержимое аккумуляторов 1 и 2 как числа целого типа (DINT). Инструкция призвана выполнить перемножение этих двух числовых величин и сохранить результат перемножения в формате числа в аккумуляторе accumulator 1.

После выполнения инструкции биты состояния CC0 и CC1 показывают, каков результат перемножения чисел: отрицателен, равен нулю или положителен. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

Деление данных типа DINT

Инструкция /D для деления интерпретирует содержимое аккумуляторов 1 и 2 как числа целого типа (DINT). Инструкция призвана выполнить деление числа, находящегося в аккумуляторе accumulator 2 (делимое), на число, находящееся в аккумуляторе accumulator 1 (делитель), и сохранить частное от деления в аккумуляторе accumulator 1.

Частное является целым результатом операции деления. Частное от деления равно нулю в двух случаях: если делимое равно нулю, в то время как делитель не равен нулю, или если делимое меньше чем делитель. Частное от деления отрицательно, если делитель меньше нуля.

После выполнения инструкции биты состояния CC0 и CC1 показывают, каково частное от деления: отрицательно, равно нулю или положительно. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

В случае деления на ноль частное от деления возвращается с нулевым значением, а биты состояния CC0, CC1, OV и OS устанавливаются в состояние "1".

Остаток от деления данных типа DINT

Инструкция MOD для нахождения остатка от деления интерпретирует содержимое аккумуляторов 1 и 2 как числа целого типа (DINT). Инструкция призвана выполнить деление числа, находящегося в аккумуляторе accumulator 2 (делимое), на число, находящееся в аккумуляторе accumulator 1 (делитель), и сохранить остаток от деления в аккумуляторе accumulator 1.

Остаток - это то, что остается в результате операции деления нацело. Нельзя путать остаток от деления с дробной частью результата операции деления. Остаток от деления отрицателен, если делимое меньше нуля.

После выполнения инструкции биты состояния CC0 и CC1 показывают, каков остаток от деления: отрицателен, равен нулю или положителен. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

В случае деления на ноль остаток от деления возвращается с нулевым значением, а биты состояния CC0, CC1, OV и OS устанавливаются в состояние "1".

10.4 Вычисления с данными типа REAL

Сложение данных типа REAL

Инструкция +R для сложения интерпретирует содержимое аккумуляторов 1 и 2 как числа типа (REAL). Инструкция призвана выполнить сложение этих двух числовых величин и сохранить результат сложения в аккумуляторе accumulator 1.

После выполнения инструкции биты состояния CC0 и CC1 показывают, какова результирующая сумма: отрицательна, равна нулю или положительна. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

В случае, когда не соблюдаются корректные условия для выполнения операции (одно из входных значений является некорректным действительным числом (invalid REAL number) или делается попытка сложить [-бесконечное число] и [+бесконечное число]), тогда операция +R возвращает некорректное результирующее значение в аккумулятор accumulator 1 и устанавливает биты состояния CC0, CC1, OV и OS в состояние "1".

Вычитание данных типа REAL

Инструкция -R для вычитания интерпретирует содержимое аккумуляторов 1 и 2 как числа типа (REAL). Инструкция призвана выполнить вычитание числа, находящегося в аккумуляторе accumulator 1, из числа, находящегося в аккумуляторе accumulator 2 и сохранить результат вычитания (разность) в аккумуляторе accumulator 1.

После выполнения инструкции биты состояния CC0 и CC1 показывают, какова разность: отрицательна, равна нулю или положительна. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

В случае, когда не соблюдаются корректные условия для выполнения операции (одно из входных значений является некорректным действительным числом (invalid REAL number) или делается попытка вычесть [+бесконечное число] из [+бесконечного числа]), тогда операция -R возвращает некорректное результирующее значение в аккумулятор accumulator 1 и устанавливает биты состояния CC0, CC1, OV и OS в состояние "1".

Умножение данных типа REAL

Инструкция *R для перемножения интерпретирует содержимое аккумуляторов 1 и 2 как числа типа (REAL). Инструкция призвана выполнить перемножение этих двух числовых величин и сохранить результат перемножения в формате числа в аккумуляторе accumulator 1.

После выполнения инструкции биты состояния CC0 и CC1 показывают, каков результат перемножения чисел: отрицателен, равен нулю или положителен. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

В случае, когда не соблюдаются корректные условия для выполнения операции (одно из входных значений является некорректным действительным числом (invalid REAL number) или делается попытка перемножить бесконечное число и 0), тогда операция *R возвращает некорректное результирующее значение в аккумулятор accumulator 1 и устанавливает биты состояния CC0, CC1, OV и OS в состояние "1".

Деление данных типа REAL

Инструкция /R для деления интерпретирует содержимое аккумуляторов 1 и 2 как числа типа (REAL). Инструкция призвана выполнить деление числа, находящегося в аккумуляторе accumulator 2 (делимое), на число, находящееся в аккумуляторе accumulator 1 (делитель), и сохранить частное от деления в аккумуляторе accumulator 1.

После выполнения инструкции биты состояния CC0 и CC1 показывают, каково частное от деления: отрицательно, равно нулю или положительно. Биты состояния OV и OS указывают на нарушение границ разрешенного диапазона.

В случае деления на ноль частное от деления возвращается с нулевым значением, а биты состояния CC0, CC1, OV и OS устанавливаются в состояние "1".

В случае, когда не соблюдаются корректные условия для выполнения операции (одно из входных значений является некорректным действительным числом (invalid REAL number) или делается попытка разделить бесконечное число на бесконечное число или 0 разделить на 0), тогда операция /R возвращает некорректное результирующее значение в аккумулятор accumulator 1 и устанавливает биты состояния CC0, CC1, OV и OS в состояние "1".

10.5 Последовательное выполнение арифметических функций

Вы можете запрограммировать выполнение арифметических операций одну за другой; в этом случае результат выполнения первой операции будет использоваться для обработки в последующей операции. При этом аккумуляторы используются для временного хранения данных.

Примечание:

Необходимо помнить, что последовательное выполнение операций в процессорах S7-300 CPU и S7-400 CPU происходит по-разному (S7-300 CPU имеют только 2 аккумулятора, тогда как S7-400 CPU имеют 4).

Цепочки вычислений в S7-300 CPU (кроме CPU 318)

Цепочки вычислений выполняются следующим образом: после выполнения арифметической операции следует операция загрузки, после чего следует комбинирование вновь введенного значения согласно текущей инструкции с результатом предыдущей операции.

Пример:

Result1 := Value1 + Value2 - Value3

```
L  Value1;
L  Value2;
+I;                //Value1 + Value2
```

```
L  Value3;
-I;                //Сумма - Value3
T  Result1;
```

В CPU с двумя аккумуляторами первое загруженное значение остается без изменений в аккумуляторе accumulator 2 во время выполнения арифметической функции и может быть вновь использовано без необходимости повторной его загрузки.

Пример:

Result2 := Value5 + 2 · Value6

```
L  Value6;
L  Value5;
+R;                //Value5 + Value6
+R;                //Сумма + Value6
T  Result2;
```

Пример:

Result3 := Value7 · $(Value8)^2$

```
L  Value8;
L  Value7;
*D;                //Value7 · Value8
*D;                //Произведение · Value8
T  Result3;
```

Цепочки вычислений в S7-400 CPU и CPU 318

Цепочки вычислений выполняются следующим образом: после выполнения арифметической операции следует операция загрузки, после чего следует комбинирование вновь введенного значения согласно текущей инструкции с результатом предыдущей операции. В CPU, имеющих 4 аккумулятора, значение из аккумулятора accumulator 3 может быть смещено в аккумулятор accumulator 2 сразу после первой операции вычисления. Заблаговременно с помощью инструкции ENT Вы можете сохранить промежуточный результат в аккумуляторе accumulator 3 (например, перед строкой с оператором вычисления) (см. раздел 6.4 "Функции аккумуляторов").

Пример:

Result4 := (Value1 + Value2) · (Value3 - Value4)

```
L  Value1;
L  Value2;
+I ;
L  Value3;
ENT;
```

```
L  Value4;
-I ;
*I ;
T  Result4;
```

В данном примере сначала вычисляется сумма *Value1* и *Value2.* Затем во время загрузки в аккумулятор accumulator 1 *Value3* сумма сдвигается в аккумулятор accumulator 2. Оттуда инструкция ENT копирует ее в аккумулятор accumulator 3. После того как будет загружено значение *Value4, Value3* окажется в аккумуляторе accumulator 2. После выполнения операции вычитания сумма "выбирается" из аккумулятора accumulator 3 в accumulator 2. После этого сумма и разность могут быть перемножены.

10.6 Добавление констант к содержимому аккумулятора Accumulator 1

+	B#16#bb	добавление байтовой константы (byte)
+	±w	добавление константы формата слово (word)
+	L#±d	добавление константы формата двойное слово (duobleword)

Вы можете запрограммировать добавление константы в соответствии со следующей общей схемой:

```
Загрузка адреса Address;
Сложение с константой Const;
Передача результата  Result;
```

Операция добавления константы более предпочтительна для вычислений с адресуемыми данными, потому что по сравнению с арифметическими функциями она не влияет ни на содержимое остальных аккумуляторов, ни на биты состояния.

Операция добавления константы предназначена выполнять сложение указанной в инструкции константы с содержимым аккумулятора accumulator 1. Вы можете определить константу как шестнадцатеричную константу формата byte или как десятичную константу формата word или как десятичную константу формата duobleword. Если необходимо прибавить константу формата word, используя тип DINT, добавьте к описанию константы префикс L#. Если десятичная константа по величине превышает разрешенный диапазон для типа INT, вычисление (добавление константы) автоматически выполняется как для типа данных DINT.

Вы можете записать десятичное число со знаком минус - таким образом можно выполнить операцию вычитания константы из значения в аккумуляторе. Перед добавлением байтовой константы она преобразуется в

число формата INT со знаком.

Как и для арифметических функций, с данными типа INT операция добавления байтовой константы или константы формата word влияет только на младшее слово аккумулятора accumulator 1 и не влияет на старшее слово аккумулятора.

Если разрешенный диапазон для типа INT нарушается, то бит 15 (бит знака) переписывается. Операция добавления константы формата word обрабатывает все 32 бита аккумулятора, что соответствует операции вычисления для данных типа DINT.

Указанные операции добавления константы выполняются вне всякой связи с какими-либо условиями.

Пример операции добавления констант:

```
L   AddValue1;
+   B#16#21;
T   AddResult1;
```

В примере значение переменной AddValue1 увеличивается на 33 и передается в переменную AddResult1.

```
L   AddValue2;
+   -33;
T   AddResult2;
```

В примере значение переменной AddValue2 уменьшается на 33 и сохраняется в переменной AddResult2.

```
L   AddValue3;
+   L#-1;
T   AddResult3;
```

В примере значение переменной AddValue3 уменьшается на 1 и сохраняется в переменной AddResult3. Операция вычитания как при вычислениях с данными типа DINT.

10.7 Операции декрементирования и инкрементирования

DEC	n	Декрементирование
INC	n	Инкрементирование

Вы можете запрограммировать операции декрементирования и инкрементирования в соответствии со следующей общей схемой:

```
Загрузка адреса Address;
Декрементирование;
Передача результата  Result;
```

```
Загрузка адреса Address;
Инкрементирование;
Передача результата  Result;
```

Операции декрементирования и инкрементирования меняют значение в аккумуляторе accumulator 1. Содержимое аккумулятора уменьшается (декрементируется) или увеличивается (инкрементируется) на число, определенное в параметре инструкции. Этот параметр может принимать значение в диапазоне от 0 до 255.

При этом изменяется только значение в младшем байте аккумулятора. Старший байт аккумулятора остается без изменения. Операция инкрементирования выполняется таким образом, что если значение в результате инкрементирования превышает величину 255, то расчет значения начинается с начала (с 0), а операция декрементирования выполняется так, что если значение в результате декрементирования становится меньше 0, то расчет значения начинается с максимально возможного значения (с 255).

Операции декрементирования и инкрементирования выполняются независимо от значения RLO. Эти операции выполняются при появлении соответствующих инструкций и не влияют ни на RLO, ни на биты состояния.

Примеры операций декрементирования и инкрементирования:

```
L   IncValue;
INC 5;
T   IncResult;
```

В примере значение переменной IncValue увеличивается на 5 и передается в переменную IncResult.

```
L   DecValue;
DEC 7;
T   DecResult;
```

В примере значение переменной DecValue уменьшается на 7 и сохраняется в переменной DecResult.

11 Математические функции

К математическим функциям относятся следующие функции:

- синус (SIN), косинус (COS), тангенс (TAN)
- арксинус (ASIN), арккосинус (ACOS), арктангенс (ATAN)
- возведение в квадрат (SQR), извлечение квадратного корня (SQRT)
- экспонента (EXP), логарифм (LN)

Все математические функции обрабатывают числа в формате REAL. В зависимости от результата математические функции устанавливают биты состояния CC0, CC1, OV и OS (см. главу 15 "Биты состояния").

В этой главе будут рассмотрены математические функции, используемые в языке программирования STL. В языке программирования SCL выполнение математических функций обеспечивается с помощью соответствующих стандартных функций SCL (см. раздел 30.3 "Математические выражения").

Примеры, рассматриваемые в данной главе, Вы можете найти на прилагаемой дискете в библиотеке STL_Book library в разделе "Digital Functions" ("Функции для обработки чисел") в функциональном блоке FB 111 или в исходном файле Chap_11.

11.1 Общее представление математических функций

Математические функции в качестве входного значения используют число, находящееся в аккумуляторе accumulator 1, данное число обрабатывается в соответствии с инструкцией функции и вновь сохраняется в аккумуляторе accumulator 1.

Вы можете программировать математические функции в соответствии со следующей общей схемой:

```
Загрузка (load) адреса Address;
Математическая функция;
Передача (transfer) результата Result;
```

Математические функции изменяют только содержимое аккумулятора accumulator 1; содержимое всех остальных аккумуляторов остается неизменным. Математические функции всегда выполняются вне всякой связи с какими-либо условиями.

В таблице 11.1 показаны три примера применения математических функций.

Таблица 11.1 Примеры математических функций

Применение функции SIN	Значение в двойном слове MD 110 содержит значение угла в радианах. Функция генерирует синус угла и сохраняет результат в двойном слове MD 104.	`L MD 110;`
		`SIN;`
		`T MD 104;`

Применение функции SQRT	Функция генерирует квадратный корень из значения в переменной "MathValue1" и результат сохраняется в переменной "MathRoot". Все переменные в блоке данных "Global_DB".	`L "Global_DB".MathValue1;`
		`SQRT;`
		`T "Global_DB".MathRoot;`

Применение функции EXP	Переменные #Result содержит показатель степени e и #Exponent.	`L #Exponent;`
		`EXP;`
		`T #Result;`

Математические функции выполняются в соответствии с правилами обработки чисел типа REAL, даже когда применяется абсолютная адресация и тип входного числа не описан.

В случае, когда аккумулятор accumulator 1 содержит некорректное действительное число (invalid REAL number) при выполнении математической функции, тогда функция возвращает некорректное результирующее значение (REAL) в аккумулятор accumulator 1 и устанавливает биты состояния CC0, CC1, OV и OS в состояние "1".

11.2 Тригонометрические функции

Тригонометрические функции

- SIN синус
- COS косинус
- TAN тангенс

воспринимают действительное число (REAL) в аккумуляторе accumulator 1 как значение угла, выраженное в радианах.

Обычно для измерения углов применяются две единицы измерения: градусы с диапазоном измерения от 0° до 360° и радианы с диапазоном измерения от 0 до 2π (где π = +3.141593e+00). Обе эти единицы пропорциональны друг другу и могут быть преобразованы одна в другую. Например, угол 90° в радианах имеет значение π / 2 или +1.570796e+00.

В случаях, когда угол, выраженный в радианах, имеет величину, большую, чем 2π или +6.283185e+00, то из этой величины вычитается 2π или кратное 2π, до тех пор, пока угол не станет меньше 2π.

Пример:

Расчет реактивной мощности: $P_s = U \cdot I \cdot Sin\ (\varphi)$

```
L    PHI;          //Загрузка угла фи
SIN  ;
L    Current;      //Загрузка значения тока
*R   ;
L    Voltage;      //Загрузка значения напряжения
*R   ;
T    I_Power;      //Сохранение значения реактивной
                   //мощности
```

Необходимо отметить, что угол должен быть выражен в радианах. Если угол выражается в градусах, его значение должно быть умножено на коэффициент

π / 180 = +1.745329e-02

до того, как Вы используете значение угла в тригонометрических функциях.

11.3 Обратные тригонометрические функции (Arc-функции)

Arc-функции

- ASIN арксинус
- ACOS арккосинус
- ATAN арктангенс

являются обратными по отношению к соответствующим тригонометрическим функциям, рассмотренным выше. Эти функции используют действительное число (REAL), находящееся в аккумуляторе accumulator 1, и возвращают значение угла, выраженное в радианах (см. табл. 11.2).

Если превышается разрешенный диапазон значений, то обратные тригонометрические функции возвращают некорректное действительное число и устанавливают биты состояния CC0, CC1, OV и OS в состояние "1".

Таблица 11.2 Диапазоны значений для Arc-функций

Функция	Разрешенный диапазон значений	Возвращаемое значение
ASIN	-1 … +1	$-\pi/2 \ldots +\pi/2$
ACOS	-1 … +1	$0 \ldots +\pi$
ATAN	Не ограничен	$-\pi/2 \ldots +\pi/2$

Обычно

Пример:

В прямоугольном треугольнике величина одного из катетов относится к величине гипотенузы как 0,343. Сколько градусов составляет угол между ними?

Операция Arcsin(0.343) возвращает угол, выраженный в радианах; умножение на коэффициент 360/2π (= 57.2958) позволяет определить угол в градусах (приблизительно 20°).

Программа для выполнения указанных операций:

```
L    0.343;
ASIN  ;
L    57.2958;
*R   ;
T    Angle_Degree;
```

11.4 Другие математические функции

Пользователю доступны также еще несколько математических функций:

- SQR возведение в квадрат
- SQRT извлечение квадратного корня
- EXP экспоненциальная функция по основанию e
- LN логарифмическая функция по основанию e (натуральный логрифм)

Возведение в квадрат

Функция SQR выполняет операцию возведения в квадрат содержимого аккумулятора accumulator 1.

Пример:

Расчет объема цилиндра: $V = r^2 \cdot \pi \cdot H$

```
L    Radius;          //Загрузка значения радиуса
                      //основания
SQR  ;
L    Height;          //Загрузка значения высоты
*R   ;
L    3.141592;        //Загрузка значения числа π
*R   ;
T    Volume;          //Сохранение значения объема
```

Извлечение квадратного корня

Функция SQRT выполняет операцию извлечения квадратного корня из содержимого аккумулятора accumulator 1. Если содержимое аккумулятора меньше нуля, то функция возвращают некорректное действительное число и устанавливают биты состояния CC0, CC1, OV и OS в состояние "1". Если аккумулятор содержит "-0" (минус 0), функция возвращает "-0".

Пример:

$c = SQRT(a^2 + b^2)$

```
L    #a;
SQR  ;
L    #b;
SQR  ;
+R   ;
SQRT  ;
T    #c;
```

(Если b или c объявлены локальными переменными, перед их символами должен стоять префикс #, чтобы компилятор распознал их как локальные переменные; если b или c являются глобальными переменными, они должны быть заключены в кавычки).

Экспоненциальная функция (степенная функция с основанием e)

Функция EXP выполняет операцию нахождения степени с основанием e (содержимое аккумулятора accumulator 1 берется как показатель степени для этой функции (e^{Accu1}); e = 2.718282e+00).

Пример:
любая степень может быть определена по формуле

$a^b = e^{b \cdot \ln a}$

```
L    Value_a;
LN   ;
L    Value_b;
*R   ;
EXP  ;
T    Power;        //Значение степени
```

Логарифмическая функция (натуральный логарифм)

Функция LN выполняет операцию нахождения натурального логарифма (содержимое аккумулятора accumulator 1 берется как входное значение для этой функции; при этом e = 2.718282e+00). Если содержимое аккумулятора меньше нуля или равно нулю, то функция возвращают некорректное действительное число и устанавливают биты состояния CC0, CC1, OV и OS в состояние "1".

Функция натурального логарифма является обратной по отношению к экспоненциальной функции:

если $y = e^x$, то $x = \ln(y)$.

Пример:
с помощью переходных формул можно вычислять логарифмы по любому основанию:

Базовая формула:

$$\log_b a = \frac{\log_n a}{\log_n b}$$

В базовой формуле *b* и *n* являются произвольными основаниями. Если Вы назначите n = e, то можно будет вычислить логарифм по любому основанию, используя в вычислениях натуральные логарифмы:

$$\log_b a = \frac{\ln a}{\ln b}$$

В частном случае, при расчетах логарифмов по основанию 10 (десятичных логарифмов) формула принимает вид:

$$\lg a = \frac{\ln a}{\ln 10} = 0.4342945 \cdot \ln a$$

12 Функции преобразования

Функции преобразования преобразуют (конвертируют) тип данных, находящихся в аккумуляторе accumulator 1. На рис.12.1 представлен обзор функций преобразования, рассматриваемых в данной главе.

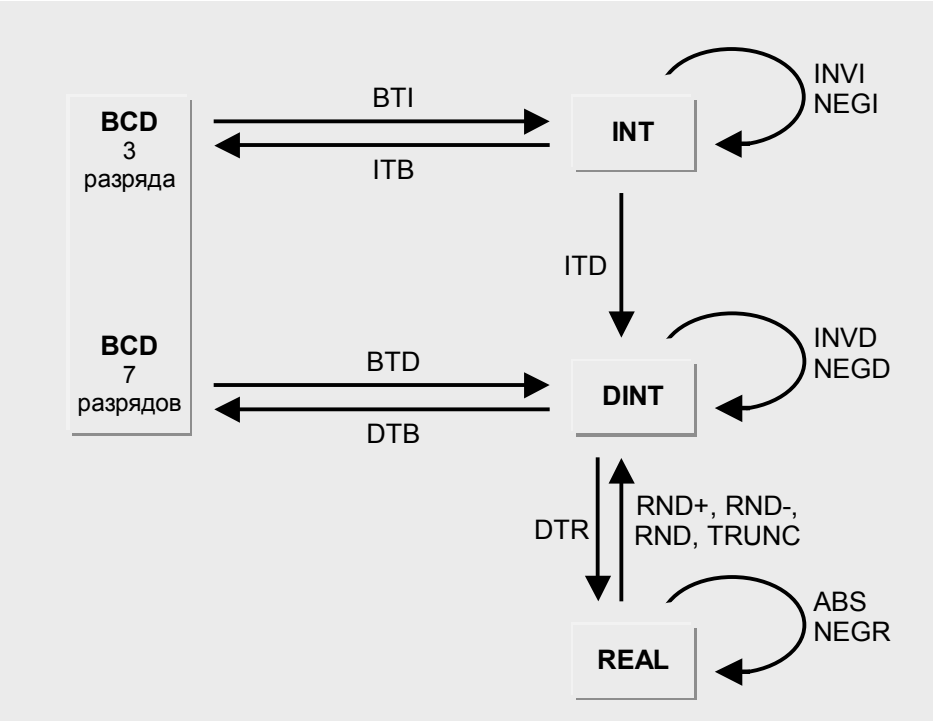

Рис.12.1 Обзор функций преобразования

В этой главе будут рассмотрены функции преобразования, используемые в языке программирования STL. В языке программирования SCL выполнение функций преобразования обеспечивается с помощью соответствующих стандартных функций SCL (см. раздел 30.5 "Функции преобразования").

Вы можете найти также подробную информацию о битовой структуре различных форматов данных в главе 24 "Типы данных", а также информацию о том, как функции преобразования устанавливают биты состояния в главе 15 "Биты состояния".

Примеры, рассматриваемые в данной главе, Вы можете найти на прилагаемой дискете в библиотеке STL_Book library в разделе "Digital Functions" ("Функции для обработки чисел") в функциональном блоке FB 112 или в исходном файле Chap_12.

12.1 Выполнение функций преобразования

Функции преобразования воздействуют только на данные, находящиеся в аккумуляторе accumulator 1. При этом отдельные функции преобразования воздействуют только на содержимое младшего слова в аккумуляторе accumulator 1 (биты с 0 по 15), другие функции воздействуют на содержимое аккумулятора в целом.

Вы можете программировать функции преобразования в соответствии со следующей общей схемой:

Загрузка (load) адреса Address;

Функция преобразования;

Передача (transfer) результата Result;

В таблице 12.1 показаны примеры применения функций преобразования для данных каждого типа. Функции преобразования конвертируют формат данных в соответствии с инструкцией функции, даже если используется абсолютная адресация и тип данных в аккумуляторе не был объявлен. Функции преобразования всегда выполняются вне всякой связи с какими-либо условиями.

Таблица 12.1 Примеры функций преобразования

Преобразование данных типа INT	Значение в двойном слове MW 120 интерпретируется как число INT. Функция сохраняет результат в двойном слове MW 122 как BCD-число.	`L MW 120;`
		`ITB;`
		`T MW 122;`

Преобразование данных типа DINT	Значение в переменной "ConvertDINT" интерпретируется как число DINT. Функция сохраняет результат в переменной "ConvertREAL" как число REAL. Все переменные в блоке данных "Global_DB".	`L "Global_DB".ConvertDINT;`
		`DTR;`
		`T "Global_DB".ConvertREAL;`

Преобразование данных типа REAL	Генерируется абсолютное значение для переменной #Display.	`L #Display;`
		`ABS;`
		`T #Display;`

Последовательное выполнение функций преобразования

Вы можете использовать содержимое аккумулятора accumulator 1 в нескольких последовательно выполняемых функциях преобразования, и в этом поэтапном процессе преобразования нет необходимости использовать дополнительные ресурсы для временного хранения конвертированных значений.

Пример:

```
L    BCD_Number;
BTI  ;               //BCD  ->  INT
ITD  ;               //INT  ->  DINT
DTR  ;               //DINT ->  REAL
T    REAL_Number;
```

В данном примере BCD-число в три шага преобразуется в число формата REAL.

12.2 Преобразование чисел форматов INT и DINT

Следующие функции обеспечивают выполнение преобразований чисел форматов INT и DINT:

- ITD преобразует INT в DINT
- ITB преобразует INT в BCD
- DTB преобразует DINT в BCD
- DTR преобразует DINT в REAL

Преобразование данных формата INT в формат DINT

Функция ITD интерпретирует содержимое младшего слова в аккумуляторе accumulator 1 (биты с 0 по 15) как число типа INT и пересылает значение бита 15 (бит знака) в старшее слово, т.е. в биты с 16 по 31.

Функция преобразования INT в DINT не устанавливает битов состояния.

Преобразование данных формата INT в формат BCD

Функция ITB интерпретирует содержимое младшего слова в аккумуляторе accumulator 1 (биты с 0 по 15) как число типа INT и преобразует его в трехразрядное BCD-число. Трехразрядное BCD-число в аккумуляторе accumulator 1 располагаются с выравниванием вправо и представляет собой значение десятичного числа. Знак располагается в битах с 12 по 15. Если все эти биты имеют значение "0", то знак положительный, если все биты имеют значение "1", то знак отрицательный. Содержимое старшего слова (биты с 16 по 31) остается без изменений.

Если исходное INT-число слишком велико, для того, чтобы выполнить преобразование в BCD-число (>999), тогда функция ITB устанавливает биты состояния OV и OS. Преобразование в таком случае не будет выполнено.

Преобразование данных формата DINT в формат BCD

Функция DTB интерпретирует содержимое аккумулятора accumulator 1 как число типа DINT и преобразует его в семиразрядное BCD-число. Семиразрядное BCD-число в аккумуляторе accumulator 1 располагаются с выравниванием вправо и представляет собой значение десятичного

числа. Знак располагается в битах с 28 по 31. Если все эти биты имеют значение "0", то знак положительный, если все биты имеют значение "1", то знак отрицательный.

Если исходное DINT-число слишком велико, для того, чтобы выполнить преобразование в BCD-число (> 9 999 999), в таком случае функция DTB устанавливает биты состояния OV и OS и преобразование в таком случае не может быть выполнено.

Преобразование данных формата DINT в формат REAL

Функция DTR интерпретирует содержимое аккумулятора accumulator 1 как число формата DINT и преобразует его в число формата REAL. Так как число формата DINT имеет большую точность чем число формата REAL, то в процессе преобразования формата числа может произойти округление, но только до следующего целого числа (как при выполнении операции RND). Если все эти биты имеют значение "0", то знак положительный, если все биты имеют значение "1", то знак отрицательный.

Функция DTR не устанавливает битов состояния.

12.3 Преобразование чисел формата BCD

Следующие функции обеспечивают выполнение преобразований чисел формата BCD:

- BTI преобразует BCD в INT
- BTD преобразует BCD в DINT

Преобразование данных формата BCD в формат INT

Функция BTI интерпретирует содержимое младшего слова в аккумуляторе accumulator 1 (биты с 0 по 15) как трехразрядное BCD-число и преобразует его в число типа INT. Три разряда числа в аккумуляторе accumulator 1 располагаются с выравниванием вправо и представляют собой значение десятичного числа. Знак располагается в битах с 12 по 15. Если все эти биты имеют значение "0", то знак положительный, если все биты имеют значение "1", то знак отрицательный. В процессе преобразования берется в расчет только состояние бита 15.

Содержимое старшего слова (биты с 16 по 31) остается без изменений.

При обнаружении "псевдотетрады" в исходном BCD-числе (численные значения с 10 по 15 или с A по F в шестнадцатеричном числе), CPU сообщает об ошибке назначения параметра ("a parameter assignment error") и вызывает организационный блок OB 121 (блок обработки синхронных ошибок). Если блок OB 121 не запрограммирован, то CPU переходит в состояние STOP.

Функция преобразования BTI не устанавливает битов состояния.

Преобразование данных формата BCD в формат DINT

Функция BTD интерпретирует содержимое аккумулятора accumulator 1 как семиразрядное BCD-число и преобразует его в число типа DINT. Семь разрядов числа в аккумуляторе accumulator 1 располагаются с выравниванием вправо и представляют собой значение десятичного числа. Биты с 28 по 31 содержат знак. Если все эти биты имеют значение "0", то знак положительный, если все биты имеют значение "1", то знак отрицательный. В процессе преобразования берется в расчет только состояние бита 31.

При обнаружении "псевдотетрады" в исходном BCD-числе (численные значения с 10 по 15 или с A по F в шестнадцатеричном числе), CPU сообщает об ошибке назначения параметра ("a parameter assignment error") и вызывает организационный блок OB 121 (блок обработки синхронных ошибок). Если блок OB 121 недоступен, то CPU переходит в состояние STOP.

Функция преобразования BTD не устанавливает битов состояния.

12.4 Функции преобразования чисел формата REAL

Существует несколько функций, обеспечивающих выполнение операций преобразования дробного числа формата REAL в число формата DINT (преобразование дробного числа в целое число). Эти функции отличаются друг от друга способом выполнения операции округления.

- RND+ преобразование с округлением "вверх" до следующего целого
- RND- преобразование с округлением "вниз" до следующего целого
- RND преобразование с округлением до следующего целого числа
- TRUNC преобразование без округления (усечение)

В таблице 12.2 показано различие действий указанных функций преобразования чисел формата REAL в числа формата DINT. Диапазон от -1 до +1 был выбран выбран для примера.

Преобразование с округлением "вверх" до следующего целого числа

Функция RND+ интерпретирует содержимое аккумулятора accumulator 1 как число типа REAL и преобразует его в число формата DINT.

Функция RND+ возвращает целое число, которое больше исходного числа или равно ему.

Если исходное число в аккумуляторе выходит за пределы допустимого для формата DINT диапазона или исходное число имеет другой формат, отличный от формата REAL, то функция RND+ устанавливает биты состояния OV и OS. Преобразование в таком случае не может быть выполнено.

Таблица 12.2 Режимы округления функции преобразования чисел типа REAL

Входное значение		Результат			
REAL	DW#16#	RND	RND+	RND-	TRUNC
1.00000001	3F80 0001	1	2	1	1
1.00000000	3F80 0000	1	1	1	1
0.99999995	3F7F FFFF	1	1	0	0
0.50000005	3F00 0001	1	1	0	0
0.50000000	3F00 0000	0	1	0	0
0.49999996	3EFF FFFF	0	1	0	0
5.877476e-39	0080 0000	0	1	0	0
0.0	0000 0000	0	0	0	0
-5.877476e-39	8080 0000	0	0	-1	0
-0.49999996	BEFF FFFF	0	0	-1	0
-0.50000000	BF00 0000	0	0	-1	0
-0.50000005	BF00 0001	-1	0	-1	0
-0.99999995	BF7F FFFF	-1	0	-1	0
-1.00000000	BF80 0000	-1	-1	-1	-1
-1.00000001	BF80 0001	-1	-1	-2	-1

Преобразование с округлением "вниз" до следующего целого числа

Функция RND- интерпретирует содержимое аккумулятора accumulator 1 как число типа REAL и преобразует его в число формата DINT.

Функция RND- возвращает целое число, которое меньше исходного числа или равно ему.

Если исходное число в аккумуляторе выходит за пределы допустимого для формата DINT диапазона или исходное число имеет другой формат, отличный от формата REAL, то функция RND- устанавливает биты состояния OV и OS. Преобразование в таком случае не будет выполнено.

Преобразование с округлением до ближайшего целого числа

Функция RND интерпретирует содержимое аккумулятора accumulator 1 как число типа REAL и преобразует его в число формата DINT.

Функция RND возвращает целое число, которое может быть больше или меньше исходного числа, или равно ему.

Функция RND возвращает то целое число, которое ближе к результату преобразования типа исходного числа. Если значение результата лежит точно посредине между четным и нечетным целыми числами, то четное число будет иметь более высокий приоритет, т.е. возвращается четное.

Если исходное число в аккумуляторе выходит за пределы допустимого для формата DINT диапазона или исходное число имеет другой формат, отличный от REAL, то функция RND устанавливает биты состояния OV и OS. Преобразование в таком случае не будет выполнено.

Преобразование без округления

Функция TRUNC интерпретирует содержимое аккумулятора accumulator 1 как число типа REAL и преобразует его в число формата DINT.

Функция TRUNC возвращает целую часть обрабатываемого числа, то есть, дробная часть числа отбрасывается ("усекается").

Если исходное число выходит за пределы допустимого для формата DINT диапазона или исходное число имеет другой формат, отличный от REAL, то функция TRUNC устанавливает биты состояния OV и OS. Преобразование в таком случае не будет выполнено.

12.5 Другие функции преобразования чисел

Существует еще несколько функций, обеспечивающих выполнение операций преобразования чисел:

- INVI нахождение обратного кода двоичного числа формата INT
- INVD нахождение обратного кода двоичного числа формата DINT
- NEGI инвертирование числа формата INT
- NEGD инвертирование числа формата DINT
- NEGD инвертирование числа формата REAL
- ABS нахождение абсолютного значения числа формата REAL

Нахождение обратного кода двоичного числа формата INT

Функция INVI инвертирует содержимое младшего слова в аккумуляторе accumulator 1 (биты с 0 по 15) бит за битом. Функция каждый двоичный ноль заменяет двоичной единицей и наоборот, каждую единицу заменяет нулем. Содержимое старшего слова (биты с 16 по 31) остается без изменений.

Функция преобразования INVI не устанавливает битов состояния.

Нахождение обратного кода двоичного числа формата DINT

Функция INVD инвертирует содержимое в аккумуляторе accumulator 1 бит за битом. Функция каждый двоичный ноль заменяет двоичной единицей и наоборот, каждую единицу заменяет нулем.

Функция преобразования INVD не устанавливает битов состояния.

Инвертирование числа формата INT

Функция NEGI интерпретирует содержимое младшего слова в аккумуляторе accumulator 1 (биты с 0 по 15) как число типа INT и изменяет его знак.

Эта операция идентична умножению на -1. При этом биты старшего слова в аккумуляторе accumulator 1 (биты с 16 по 31) не изменяют своего состояния.

Функция преобразования NEGI устанавливает следующие биты состояния: CC0, CC1, OV и OS.

Инвертирование числа формата DINT

Функция NEGD интерпретирует содержимое аккумулятора accumulator 1 как число типа DINT и изменяет его знак.

Эта операция идентична умножению на -1. При этом биты старшего слова в аккумуляторе accumulator 1 (биты с 16 по 31) не изменяют своего состояния.

Функция преобразования NEGD устанавливает следующие биты состояния: CC0, CC1, OV и OS.

Инвертирование числа формата REAL

Функция NEGR интерпретирует содержимое аккумулятора accumulator 1 как число формата REAL и умножает это число на -1 (функция меняет знак мантиссы, даже если число в аккумуляторе является некорректным действительным числом).

Функция NEGR не устанавливает битов состояния.

Нахождение абсолютного значения числа формата REAL

Функция ABS интерпретирует содержимое аккумулятора accumulator 1 как число формата REAL и формирует для этого числа абсолютное значение, устанавливая знак числа в значение "0" (функция устанавливает знак мантиссы в значение "0", даже если число в аккумуляторе является некорректным действительным числом).

Функция ABS не устанавливает битов состояния.

13 Функции сдвига

Функции сдвига сдвигают влево или вправо содержимое аккумулятора accumulator 1 бит за битом.

В таблице 13.1 представлен обзор доступных пользователю функций сдвига.

Таблица 13.1 Обзор функций сдвига

Функции сдвига	Word (Слово)		Duobleword (Двойное слово)	
	Число поз. как параметр	Число поз. в Accum2	Число поз. как параметр	Число поз. в Accum2
Сдвиг влево (Shift left)	SLW n	SLW	SLD n	SLD
Сдвиг вправо (Shift right)	SRW n	SRW	SRD n	SRD
Сдвиг со знаком (Shift with sign)	SSI n	SSI	SSD n	SSD
Циклический сдвиг влево (Rotate left)	-	-	RLD n	RLD
Циклический сдвиг вправо (Rotate right)	-	-	RRD n	RRD
Циклический сдвиг влево через бит CC1 (Rotate left through CC1)	-	-	RLDA [1)]	-
Циклический сдвиг вправо через бит CC1 (Rotate right through CC1)	-	-	RRDA [1)]	-

[1)] Без параметров, как при смещении только одного бита

В этой главе будут рассмотрены функции сдвига, используемые в языке программирования STL. В языке программирования SCL выполнение функций сдвига обеспечивается с помощью соответствующих стандартных функций SCL (см. раздел 30.4 "Сдвиг и циклический сдвиг").

Примеры, рассматриваемые в данной главе, Вы можете найти на прилагаемой дискете в библиотеке STL_Book library в разделе "Digital Functions" ("Функции для обработки чисел") в функциональном блоке FB 113 или в исходном файле Chap_13.

13.1 Выполнение функций сдвига

Функции сдвига позволяют побитно сдвигать влево или вправо данные, находящиеся в аккумуляторе accumulator 1. При этом в зависимости от функции сдвига аккумулятор содержит либо слово (word), либо двойное слово (duobleword). Биты, сдвигаемые за пределы слова (двойного слова), либо теряются (при операциях сдвига [shift operations]), либо переносятся на другую сторону слова или двойного слова (при операциях циклического сдвига [rotate operations]). Функции сдвига не влияют на содержимое других аккумуляторов.

Функции сдвига всегда выполняются вне всякой связи с какими-либо условиями. Функции сдвига воздействуют только на содержимое аккумулятора accumulator 1. При этом функции не влияют на результат логической операции (RLO).

Параметры для функции сдвига могут быть заданы следующими двумя путями:

- С числом позиций, заданным в аккумуляторе 2
- С числом позиций, заданным в инструкции с помощью параметра

Вы можете программировать операции двумя способами в соответствии со следующими общими схемами:

Загрузка (load) числа позиций Number_of_positions;

Загрузка (load) адреса Address;

Функция сдвига;

Передача (transfer) результата Result;

Загрузка (load) адреса Address;

Функция сдвига с параметром "число позиций" (Number_of_positions);

Передача (transfer) результата Result;

Функции сдвига устанавливают бит состояния CC0 в "0", а бит CC1 устанавливают в состояние, в котором находился последний перемещенный бит (см. рис. 13.1). Биты состояния проверяются с помощью двоичного опроса или в операциях перехода в соответствии с описанием в главе 15 "Биты состояния" и в главе 16 "Функции перехода".

В таблице 13.2 показаны несколько примеров применения функций сдвига для данных типов Word и Duobleword.

Функции сдвига для данных типа Word воздействуют только на содержимое младшего слова в аккумуляторе accumulator 1 (биты с 0 по 15). Содержимое старшего слова аккумулятора остается неизменным.

Функция циклического сдвига с битом состояния CC1 смещает содержимое аккумулятора accumulator 1 на одну разрядную позицию

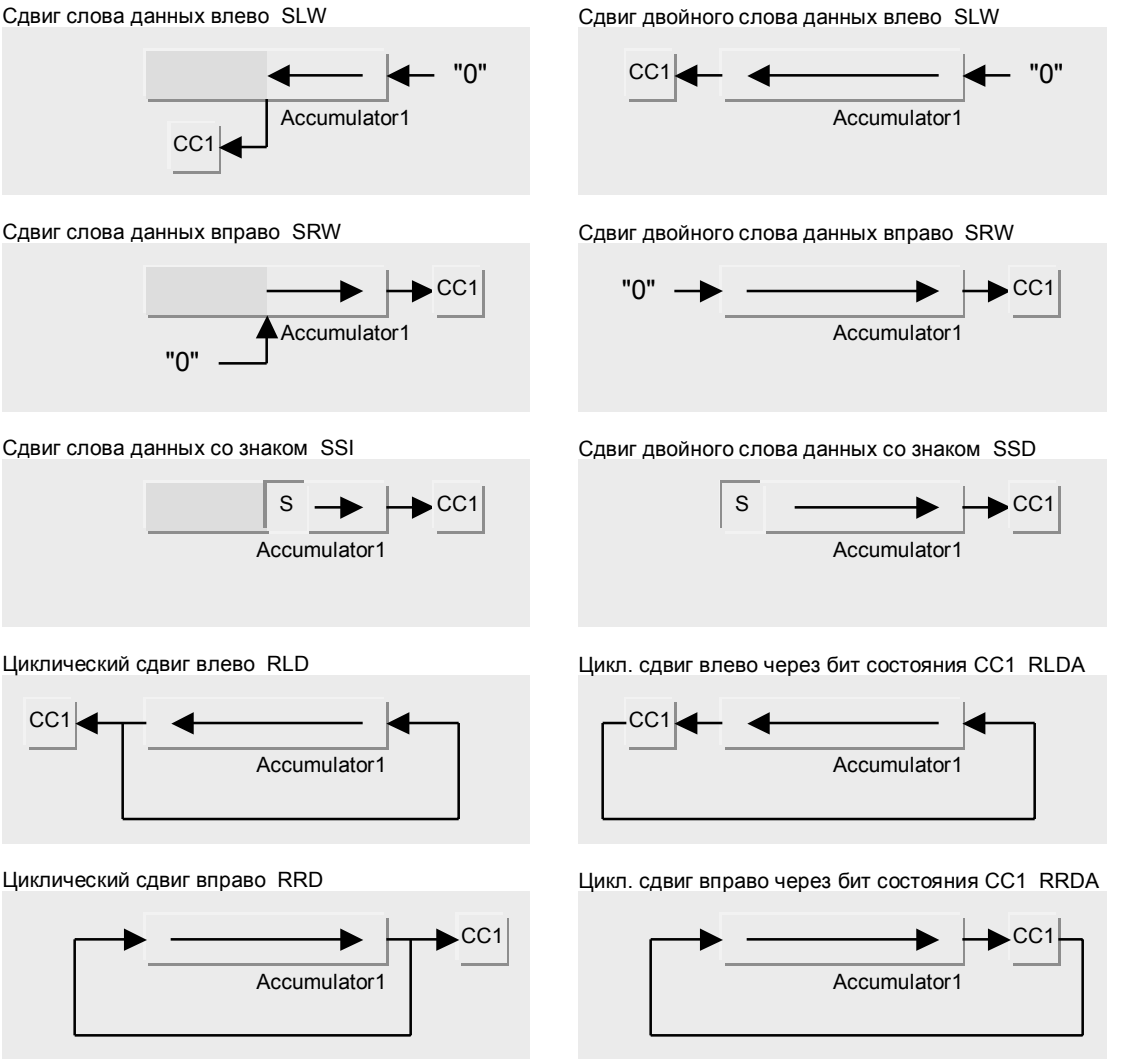

Рис.13.1 Принцип работы функций сдвига

Таблица 13.2 Примеры применения функций сдвига

Сдвиг данных формата Word (слово)	Значение в слове MW 130 сдвигается на 4 позиции влево и сохраняется в слове MW 132. Здесь число позиций задано параметром в инструкции.	`L   MW 130;`
		`SLW 4;`
		`T   MW 132;`

Сдвиг данных формата Duobleword (двойное слово)	Значение в переменной "ShiftOn" сдвигается вправо на "ShiftPos" позиций и сохраняется в переменной "ShiftOff". Здесь число позиций задано посредством аккумулятора accumulator 2.	`L   "Global_DB".ShiftPos;`
		`L   "Global_DB".ShiftOn;`
		`SRD ;`
		`T   "Global_DB".ShiftOff;`

Сдвиг со знаком	Сдвиг переменной #Actval со знаком вправо на 2 позиции и передача в переменную #Display.	`L   #Actval;`
		`SSI 2;`
		`T   #Display;`

Последовательное выполнение функций сдвига

Вы можете выполнять операцию сдвига содержимого аккумулятора accumulator 1 любое необходимое число раз.

Пример:

```
L    Value1;
SSD  4;
SLD  2;
T    Result1;
```

В данном примере в результате выполнения программы содержимое аккумулятора будет сдвинуто со знаком вправо на 2 позиции при этом два правых бита будут сброшены в состояние "0".

13.2 Операции сдвига

Сдвиг слова данных влево

SLW n	Сдвиг слова данных влево на n разрядов
SLW	Сдвиг слова данных влево на количество разрядов, указанное в аккумуляторе accumulator 2

Функция сдвига SLW позволяет бит за битом сдвигать влево данные, находящиеся в младшем слове аккумулятора accumulator 1 (т.е. в битах с 0 по 15). При этом разряды, освобождаемые при сдвиге содержимого указанных битов заполняются нулями. Биты, находящиеся в старшем слове аккумулятора остаются без изменения; нет также переноса данных в бит 16.

Число позиций, на которое выполняется операция сдвига, может быть указано в параметре инструкции SLW или может быть загружено в аккумулятор accumulator 2 в виде положительного числа формата INT, выровненного вправо. Если число позиций равно нулю, то операция сдвига не выполняется (нет операции, т.е. NOP); если число позиций больше 15, то после выполнения операции сдвига SLW все биты младшего слова аккумулятора будут иметь значение "0".

Если содержимое аккумуляторе accumulator 1 (младшего слова) интерпретируется как целое число формата INT, то сдвиг влево эквивалентен умножению на число, равное степенной функции с основанием 2 и показателем степени, равным числу позиций, на которые производится сдвиг.

Сдвиг двойного слова данных влево

SLD n	Сдвиг двойного слова данных влево на n разрядов
SLD	Сдвиг двойного слова данных влево на количество разрядов, указанное в аккумуляторе accumulator 2

Функция сдвига SLD позволяет бит за битом сдвигать влево данные, находящиеся во всех битах аккумулятора accumulator 1. При этом разряды, освобождаемые при сдвиге содержимого аккумулятора

заполняются нулями.

Число позиций, на которое выполняется операция сдвига, может быть указано в параметре инструкции SLD или может быть загружено в аккумулятор accumulator 2, в виде положительного числа формата INT, выровненного вправо. Если число позиций равно нулю, то операция сдвига не выполняется (нет операции, т.е. NOP); если число позиций больше 31, то после выполнения операции сдвига SLD все биты аккумулятора accumulator 1 будут иметь значение "0".

Если содержимое аккумуляторе accumulator 1 интерпретируется как целое число формата DINT, то сдвиг влево эквивалентен умножению на число, равное степенной функции с основанием 2 и показателем степени, равным числу позиций, на которые производится сдвиг.

Сдвиг слова данных вправо

SRW	n	Сдвиг слова данных вправо на n разрядов
SRW		Сдвиг слова данных вправо на количество разрядов, указанное в аккумуляторе accumulator 2

Функция сдвига SRW позволяет бит за битом сдвигать вправо данные, находящиеся в младшем слове аккумулятора accumulator 1 (т.е. в битах с 0 по 15). При этом разряды, освобождаемые при сдвиге содержимого указанных битов заполняются нулями. Биты, находящиеся в старшем слове (биты с 16 по 31) остаются без изменения.

Число позиций, на которое выполняется операция сдвига, может быть указано в параметре инструкции SRW или может быть загружено в аккумулятор accumulator 2 в виде положительного числа формата INT, выровненного вправо. Если число позиций равно нулю, то операция сдвига не выполняется (нет операции, т.е. NOP); если число позиций больше 15, то после выполнения операции сдвига SRW все биты младшего слова аккумулятора будут иметь значение "0".

Если содержимое аккумуляторе accumulator 1 (младшего слова) интерпретируется как целое число формата INT, то сдвиг вправо эквивалентен делению на число, равное степенной функции с основанием 2 и показателем степени, равным числу позиций, на которые производится сдвиг. Так как при сдвиге вправо производится заполнение битов нулевым значением, начиная со старшего бита в слове, то результатом всегда будет положительное число. Результат упомянутого деления соответствует округлению целой части.

Сдвиг двойного слова данных вправо

SRD	n	Сдвиг двойного слова данных вправо на n разрядов
SRD		Сдвиг двойного слова данных вправо на количество разрядов, указанное в аккумуляторе accumulator 2

Функция сдвига SRD позволяет бит за битом сдвигать вправо данные, находящиеся во всех битах аккумулятора accumulator 1. При этом разряды, освобождаемые при сдвиге содержимого этого аккумулятора заполняются нулями.

Число позиций, на которое выполняется операция сдвига, может быть указано в параметре инструкции SRD или может быть загружено в аккумулятор accumulator 2, в виде положительного числа формата INT,

выровненного вправо. Если число позиций равно нулю, то операция сдвига не выполняется (нет операции, т.е. NOP); если число позиций больше 31, то после выполнения операции сдвига SRD все биты аккумулятора accumulator 1 будут иметь значение "0".

Если содержимое аккумуляторе accumulator 1 интерпретируется как целое число формата DINT, то сдвиг вправо эквивалентен делению на число, равное степенной функции с основанием 2 и показателем степени, равным числу позиций, на которые производится сдвиг. Так как при сдвиге вправо производится заполнение битов нулевым значением, начиная со старшего бита в слове, то результатом всегда будет положительное число. Результат упомянутого деления соответствует операции округления целой части.

Сдвиг слова данных со знаком

SSI n	Сдвиг слова данных со знаком на n разрядов
SSI	Сдвиг слова данных со знаком на количество разрядов, указанное в аккумуляторе accumulator 2

Функция сдвига SSI позволяет бит за битом сдвигать вправо данные, находящиеся в младшем слове аккумулятора accumulator 1 (т.е. в битах с 0 по 15). При этом разряды, освобождаемые при сдвиге содержимого указанных битов заполняются значением бита 15 (бита, содержащего знак числа формата INT). Иначе говоря, этом разряды заполняются значением "0", если число положительное и значением "1", если число отрицательное.

Функция сдвига SSI не влияет на содержимое битов с 16 по 31.

Число позиций, на которое выполняется операция сдвига, может быть указано в параметре инструкции SSI или может быть загружено в аккумулятор accumulator 2 в виде положительного числа формата INT, выровненного вправо. Если число позиций равно нулю, то операция сдвига не выполняется (нет операции, т.е. NOP); если число позиций больше 15, то после выполнения операции сдвига SSI все биты младшего слова аккумулятора будут иметь значение, соответствующее знаку исходного числа в аккумуляторе.

Если содержимое аккумуляторе accumulator 1 (младшего слова) интерпретируется как целое число формата INT, то сдвиг вправо эквивалентен делению на число, равное степенной функции с основанием 2 и показателем степени, равным числу позиций, на которые производится сдвиг. Результат упомянутого деления соответствует округлению целой части.

Сдвиг двойного слова данных со знаком SSD

SSD n	Сдвиг слова данных со знаком на n разрядов
SSD	Сдвиг слова данных со знаком на количество разрядов, указанное в аккумуляторе accumulator 2

Функция сдвига SSD позволяет бит за битом сдвигать вправо данные, находящиеся во всех битах аккумулятора accumulator 1. При этом разряды, освобождаемые при сдвиге содержимого указанных битов заполняются значением бита 31 (бита, содержащего знак числа формата DINT). Иначе говоря, этом разряды заполняются значением "0", если число положительное и значением "1", если число отрицательное.

Число позиций, на которое выполняется операция сдвига, может быть указано в параметре инструкции SSD или может быть загружено в аккумулятор accumulator 2, в виде положительного числа формата INT, выровненного вправо. Если число позиций равно нулю, то операция сдвига не выполняется (нет операции, т.е. NOP); если число позиций больше 31, то после выполнения операции сдвига SSD все биты аккумулятора accumulator 1 будут иметь значение, соответствующее знаку исходного числа в аккумуляторе.

Если содержимое аккумуляторе accumulator 1 интерпретируется как целое число формата DINT, то сдвиг вправо эквивалентен делению на число, равное степенной функции с основанием 2 и показателем степени, равным числу позиций, на которые производится сдвиг. Результат упомянутого деления соответствует операции округления целой части.

13.3 Операции циклического сдвига

Циклический сдвиг влево

RLD n	Циклический сдвиг влево на n разрядов
RLD	Циклический сдвиг влево на на количество разрядов, указанное в аккумуляторе accumulator 2

Операция циклического сдвига RLD позволяет бит за битом сдвигать влево данные, находящиеся во всех битах аккумулятора accumulator 1. При этом вновь освобожденный при сдвиге содержимого разряд заполняется значением бита, который был "вытолкнут" из аккумулятора последним.

Число позиций, на которое выполняется операция циклического сдвига, может быть указано в параметре инструкции RLD или может быть загружено в аккумулятор accumulator 2, в виде положительного числа формата INT, выровненного вправо. Если число позиций равно нулю, то операция циклического сдвига не выполняется (нет операции, т.е. NOP); если число позиций равно 32, то содержимое аккумулятора accumulator 1 не изменяется, а бит состояния CC1 принимает состояние бита, перемещенного последним (бит 0). Если указанное число позиций равно 33, то содержимое аккумулятора accumulator 1 смещается на 1 разряд, если указанное число позиций равно 34, то содержимое смещается на 2 позиции и так далее.

Циклический сдвиг вправо

RRD n	Циклический сдвиг вправо на n разрядов
RRD	Циклический сдвиг вправо на количество разрядов, указанное в аккумуляторе accumulator 2

Операция циклического сдвига RRD позволяет бит за битом сдвигать вправо данные, находящиеся во всех битах аккумулятора accumulator 1. При этом вновь освобожденный при сдвиге содержимого разряд заполняется значением бита, который был "вытолкнут" из аккумулятора последним.

Число позиций, на которое выполняется операция циклического сдвига, может быть указано в параметре инструкции RRD или может быть загружено в аккумулятор accumulator 2, в виде положительного числа формата INT, выровненного вправо. Если число позиций равно нулю, то операция циклического сдвига не выполняется (нет операции, т.е. NOP); если число позиций равно 32, то содержимое аккумулятора accumulator 1 не изменяется, а бит состояния CC1 принимает состояние бита, перемещенного последним (бит 31). Если указанное число позиций равно 33, то содержимое аккумулятора accumulator 1 смещается на 1 разряд, если указанное число позиций равно 34, то содержимое смещается на 2 позиции и так далее.

Циклический сдвиг влево через бит состояния CC1

RLDA Циклический сдвиг влево через бит состояния CC1 на 1 разряд

Операция циклического сдвига RLDA позволяет сдвигать на 1 бит влево данные, находящиеся во всех битах аккумулятора accumulator 1. При этом вновь освобожденный при сдвиге содержимого разряд (бит 0) заполняется значением из бита состояния CC1, а бит состояния CC1, в свою очередь принимает состояние бита, который был "вытолкнут" из аккумулятора последним (бит 31). При этом бит состояния CC0 устанавливается в состояние "0".

Циклический сдвиг вправо через бит состояния CC1

RRDA Циклический сдвиг вправо через бит состояния CC1 на 1 разряд

Операция циклического сдвига RRDA позволяет сдвигать на 1 бит вправо данные, находящиеся во всех битах аккумулятора accumulator 1. При этом вновь освобожденный при сдвиге содержимого разряд (бит 31) заполняется значением из бита состояния CC1, а бит состояния CC1, в свою очередь принимает состояние бита, который был "вытолкнут" из аккумулятора последним (бит 0). При этом бит состояния CC0 устанавливается в состояние "0".

14 Логические функции для слов данных (Word Logic)

Логические функции для слов данных (Word Logic) позволяют побитно комбинировать значение, находящееся в аккумуляторе accumulator 1, с константой или с содержимым аккумулятора accumulator 2. Результат выполнения операции сохраняется в аккумуляторе accumulator 1. Логические функции могут выполняться как для данных формата Word (слов), так и для данных фомата Duobleword (двойных слов).

Пользователю доступны следующие логические операции для слов данных:

- AND логическая операция "И",
- OR логическая операция "ИЛИ",
- Exclusive OR логическая операция "Исключающее ИЛИ".

В этой главе будут рассмотрены логические функции для слов данных (Word Logic), используемые в языке программирования STL. В языке программирования SCL выполнение логических функций обеспечивается с помощью программирования соответствующих логических выражений (см. раздел 27.4.3 "Логические операции").

Информацию о том, каким образом логические функции устанавливают биты состояния Вы можете получить из главы 15 "Биты состояния". Примеры, рассматриваемые в данной главе, Вы можете найти на прилагаемой дискете в библиотеке STL_Book library в разделе "Digital Functions" ("Функции для обработки чисел") в функциональном блоке FB 114 или в исходном файле Chap_14.

14.1 Выполнение логических операций для слов данных

Вы можете программировать операции двумя способами в соответствии со следующими общими схемами:

```
Загрузка (load) адреса Address1;
Загрузка (load) адреса Address2;
Логическая операцияции для слова данных без константы;
Передача (transfer) результата Result;
```

```
Загрузка (load) адреса Address;
Логическая операцияции для слова данных с константой;
Передача (transfer) результата Result;
```

Логические функции для слов данных всегда выполняются вне всякой связи с какими-либо условиями. При этом эти функции не влияют на результат логической операции (RLO).

Формирование результата логической операции для слов данных

Логические функции для слов данных формируют результат операции бит за битом таким же образом, как и функции двоичной логики, описанные в главе 4 "Двоичные логические операции" (табл.14.1).

Таблица 14.1 Формирование результата логической операции для слов данных

Содержимое бита в аккумуляторе 2 или бита в константе	0	0	1	1
Содержимое бита в аккумуляторе 1	0	1	0	1
Результат AW, AD	0	0	0	1
Результат OW, OD	0	1	1	1
Результат XOW, XOD	0	1	1	0

Логическая функция комбинирует бит 0 аккумулятора accumulator 1 с битом 0 аккумулятора accumulator 2 или константы, указанной в инструкции. Результат сохраняется в бите 0 аккумулятора accumulator 1. Таким же образом эта же логическая операция выполняется для битов 2, битов 3 и так далее вплоть до старшего бита операндов (до бита 15 в операциях для слов и до бита 31 в операциях для двойных слов). Содержимое аккумулятора accumulator 2 при этом остается неизменным.

Логические функции для слов данных с содержимым accumulator 2

Перед инструкцией логической функции для слов данных должны следовать две операции загрузки, для того, чтобы можно было комбинировать загружаемые значения. После выполнения логической функции результат сохраняется в аккумуляторе accumulator 1.

Пример:

```
L     MW 142;        //Адрес 1
L     MW 144;        //Адрес 2
AW    ;              //Логическая операция
T     MW 146;        //Результат
```

В этом примере логическая функция выполняется с данными формата слово (Word).

Логические функции для слов данных с константой

Перед инструкцией логической функции для слов данных должна следовать операция загрузки, для того, чтобы можно было комбинировать загружаемое значение с константой, которая в свою очередь задается в операторе функции. Результат выполнения логической функции сохраняется в аккумуляторе accumulator 1.

Пример:

```
L     MW 148;
AW    W#16#807F;
T     MW 150;
L     MD 152;
OD    DW#16#8000_F000;
T     MD 156;
```

В этом примере логическая функция выполняется с данными формата двойное слово (Duobleword).

Выполнение логических операций с данными формата слово (Word)

Логические функции, выполняемые с данными формата слово (Word), воздействуют только на данные, находящиеся в младшем слове аккумулятора accumulator 1 (т.е. в битах с 0 по 15). При этом биты, находящиеся в старшем слове аккумулятора (биты с 16 по 31), остаются без изменения (см. рис. 14.1).

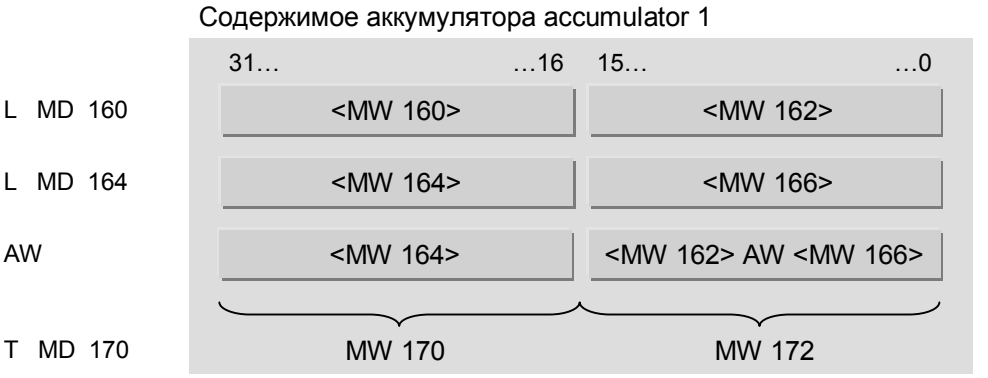

Рис.14.1 Выполнение логических операций с данными формата слово (Word)

Последовательное выполнение логических операций

После завершения логической операции для слов данных Вы можете немедленно перейти к выполнению следующей логической операции (загрузив предварительно данные посредством операции load из соответствующего адреса или задав константу как параметр в инструкции) без необходимости сохранения промежуточного результата (например, в области локальных данных). Аккумуляторы сами служат для временного хранения данных.

Пример:

```
L     Value1;
L     Value2;
AW    ;
L     Value3;
OW    ;
T     Result1;
```

В вышеуказанном примере результат выполнения инструкции AW содержится в аккумуляторе accumulator 1. Во время загрузки значения Value3 этот результат смещается в аккумулятор accumulator 2. Теперь оба значения могут быть обработаны в соответствии с инструкцией OW.

Пример:

```
L     Value4;
L     Value5;
XOW   ;
AW    W#16#FFF0;
T     Result2;
```

В данном примере результат выполнения операции XOW содержится в аккумуляторе accumulator 1. Биты с 0 по 3 аккумулятора сбрасываются в состояние "0" при выполнении операции AW.

В таблице 14.2 показаны примеры по одному для каждого типа логических операций.

Таблица 14.2 Примеры применения логических функций для слов данных

Логическая операция AND (логическое И)	Четыре старших бита слова MW 138 сбрасываются в"0"; результат сохраняется в слове MW 140.	`L   MW 138;`
		`AW  W#16#0FFF;`
		`T   MW 140;`

Логическая операция OR (логическое ИЛИ)	Переменные "WlogicVal1" и "WlogicVal1" побитно обрабатываются функцией OR (ИЛИ); результат сохраняется в переменной "WlogicReslt".	`L   "Global_DB".WlogicVal1;`
		`L   "Global_DB".WlogicVal2;`
		`OD ;`
		`T`
		`"Global_DB".WlogicReslt;`

Логическая операция Exclusive OR (Исключающее ИЛИ)	Переменные #Input и #Mask обрабатываются функцией Exclusive OR (Исключающее ИЛИ); результат сохраняется в переменной #Buffer.	`L   #Input;`
		`L   #Mask;`
		`XOW ;`
		`T   #Buffer;`

14.2 Описание логических операций для слов данных

Операция AND (И) для слов данных

AW		Логическая операция AND (И) для данных формата Word (слово) в accum1 и accum2
AW	W#16#	Логическая операция AND (И) для данных формата Word (слово) в accum1 и константе
AD		Логическая операция AND (И) для данных формата Duobleword (двойное слово) в accum1 и accum2
AD	DW#16#	Логическая операция AND (И) для данных формата Duobleword (двойное слово) в accum1 и константе

Логическая функция для обработки двух чисел в соответствии с логикой AND (И) позволяет последовательно бит за битом комбинировать данные, находящиеся в аккумуляторе accumulator 1, с соответствующими битами аккумулятора accumulator 2 или с соответствующими битами константы, указанной в инструкции функции. Бит, находящийся в слове результата, будет установлен в состояние "1" только в случае, если оба сравниваемых бита исходных данных имеют значение "1".

Так как те биты в аккумуляторе accumulator 2 или в константе, которые имеют значение "0", сбрасывают в "0" соответствующие биты слова результата вне зависимости от состояния соответствующих битов в аккумуляторе accumulator 1, то говорят, что эти биты "маскированы" ("masked"). Этот, так называемый, эффект маскирования битов и является основным назначением логических AND (И) операций для слов данных.

Операция OR (ИЛИ) для слов данных

OW		Логическая операция OR (ИЛИ) для данных формата Word (слово) в accum1 и accum2
OW	W#16#	Логическая операция OR (ИЛИ) для данных формата Word (слово) в accum1 и константе
OD		Логическая операция OR (ИЛИ) для данных формата Duobleword (двойное слово) в accum1 и accum2
OD	DW#16#	Логическая операция OR (ИЛИ) для данных формата Duobleword (двойное слово) в accum1 и константе

Логическая функция для обработки двух чисел в соответствии с логикой OR (ИЛИ) позволяет последовательно бит за битом комбинировать данные, находящиеся в аккумуляторе accumulator 1, с соответствующими битами аккумулятора accumulator 2 или с соответствующими битами константы, указанной в инструкции функции. Бит, находящийся в слове результата, будет сброшен в состояние "0" только в случае, если оба сравниваемых бита исходных данных имеют значение "0".

Так как те биты в аккумуляторе accumulator 2 или в константе, которые имеют значение "1", устанавливают в "1" соответствующие биты слова результата вне зависимости от состояния соответствующих битов в аккумуляторе accumulator 1, то говорят, что эти биты "маскированы" ("masked"). Этот, так называемый, эффект маскирования битов и является основным назначением логических OR (ИЛИ) операций для слов данных.

Операция Exclusive OR (Исключающее ИЛИ) для слов данных

XOW		Операция Exclusive OR (Исключающее ИЛИ) для данных формата Word (слово) в accum1 и accum2
XOW	W#16#	Операция Exclusive OR (Исключающее ИЛИ) для данных формата Word (слово) в accum1 и константе
XOD		Операция Exclusive OR (Исключающее ИЛИ) для данных Duobleword (двойное слово) в accum1 и accum2
XOD	DW#16#	Операция Exclusive OR (Исключающее ИЛИ) для Duobleword (двойное слово) в accum1 и константе

Логическая функция для обработки двух чисел в соответствии с логикой Exclusive OR (Исключающее ИЛИ) позволяет последовательно бит за битом комбинировать данные, находящиеся в аккумуляторе accumulator 1, с соответствующими битами аккумулятора accumulator 2 или с соответствующими битами константы, указанной в инструкции функции. Бит, находящийся в слове результата, будет установлен в состояние "1" только в случае, если только один из сравниваемых битов исходных данных имеет значение "1". Если некоторый бит в аккумуляторе accumulator 2 или в константе, имеет значение "1", то соответствующий бит слова результата в соответствии с логикой функции будет иметь инвертированное значение по отношению к предыдущему значению этого бита в аккумуляторе accumulator 1.

В результате те биты в аккумуляторе accumulator 2 или в константе, которые имеют значение "1", устанавливают в "1" соответствующие биты слова результата вне зависимости от состояния соответствующих битов в аккумуляторе accumulator 1, то говорят, что эти биты "маскированы" ("masked"). Этот, так называемый, эффект маскирования битов и является основным назначением логических Exclusive OR (Исключающее ИЛИ) операций для слов данных.