

# Структура и элементы AWL

## 8

### Обзор главы

В разделе	Вы найдете	на стр.
8.1	Структура команды	8–2
8.2	Значение регистров CPU в командах	8–10

## 8.1. Структура команды

### Компоненты команды

Команды, в зависимости от их структуры, относятся к одной из двух следующих основных групп (смотрите также рисунок 8–1):

- Команды, состоящие только из операции (например, NOT, смотрите по этому поводу главу 11.9).
- Команды, состоящие из операции и операнда (смотрите таблицы с 8–1 по 8–5 и таблицу 8–9).

Команды 1-ой группы	Команды 2-ой группы
Только операция	Операция и операнд

Рис. 8-1. Основные группы команд

### Операнд команды

Операнд команды задает константу или адрес, по которому операция находит значение (объект данных), с которым она может выполнить логическую операцию. Операнд может иметь символическое имя или абсолютное обозначение. Он может указывать на следующие элементы (смотрите также таблицы с 8–1 по 8–9):

- Константа, значение таймера или счетчика или строка ASCII-символов, которые должны загружаться в аккумулятор 1 (AKKU 1) (например, L +27, смотрите таблицу 8–1).
- Бит слова состояния контроллера (например, U UO, смотрите таблицу 8–2).
- Символическое имя (например, U MOTOR\_EIN, смотрите таблицу 8–3).
- Блок данных и адрес внутри области этого блока данных (например, L DB4.DB10, смотрите таблицу 8–4).
- Функция (FC), функциональный блок (FB), встроенная системная функция (SFC) или встроенный системный функциональный блок (SFB) и номер функции или функционального блока (смотрите таблицу 8–5).
- Признак операнда и адрес внутри области памяти, задаваемой признаком операнда (например, U E 1.0 или U E [AR 1,P#4.3], смотрите таблицу 8–9).

Таблицы с 8–1 по 8–9 показывают различные команды, из которых каждая состоит из одной операции и одного операнда.

## Постоянные значения

Таблица 8–1 показывает, как использовать постоянное значение в качестве операнда команды.

Таблица 8–1. Операнды, указывающие на значение или строку символов

Операция	Команда		Описание
	Операнд	= КОНСТАНТА	
L	+27		Загрузить целое число 27 в АККУ 1.
L	'ENDE'		Загрузить ASCII-символы 'ENDE' в АККУ 1.

## Бит слова состояния

Операнд AWL-операции может ссылаться на один или несколько битов слова состояния контроллера (см. главу 9.4). Операция опрашивает состояние сигнала отдельного бита слова состояния (например, U BIE) и реагирует на него, или она оценивает комбинацию из двух битов (например, U UO).

Таблица 8–2. Операнды, ссылающиеся на бит слова состояния

Операция	Команда		Описание
	Операнд	Бит слова состояния	
U	BIE		В логической операции используется "1" или "0" из бита 8 слова состояния.
U	UO		Эта операция оценивает комбинацию битов состояния A1 и A0, чтобы проверить, выполнено ли определенное условие. Так, например, сочетание "1" и "1" указывает "недопустимо", то есть одно из значений в операции с плавающей точкой не было в действительности числом с плавающей точкой.

## Символические имена

Таблица 8–3 показывает, как использовать символическое имя в качестве операнда команды. Вы должны согласовывать символические имена глобально (в таблице символов) или локально (в блоке), если Вы желаете использовать их в AWL–командах.

Таблица 8–3. Операнды, указывающие на символическое имя

Команда		Описание
Операция	Операнд	
	Символ	
U	MOTOR.EIN	Выполнить логическую операцию И с битом, имеющим символическое имя MOTOR.EIN. В этом примере символическое имя MOTOR.EIN может представлять только один бит из области памяти блока данных (D) или элемент структуры "MOTOR".
L	DRENZAHN	Загрузить в AKKU 1 значение в виде байта, слова или двойного слова с символическим именем DRENZAHN.

## Блок данных и адрес в блоке данных

Таблица 8–4 показывает, как использовать блок данных и адрес в этом блоке данных в качестве операнда команды.

Таблица 8–4. Операнды, указывающие на блок данных или адрес в блоке данных

Команда		Описание
Операция	Операнд	
	Блок данных и адрес	
L	DB4.DB10	Загрузить двойное слово DB10 из блока данных DB4 в AKKU 1.
U	DB10.DBX4.3	Выполнить логическую операцию И с битом данных DBX4.3 из блока данных DB10.

## FC, FB, SFC и SFB

Таблица 8–5 показывает, как использовать функцию (FC), функциональный блок (FB), встроенную системную функцию (SFC), встроенный системный функциональный блок (SFB) и номер функции или функционального блока в качестве операнда операции.

Таблица 8–5. Операнды, указывающие на функцию, функциональный блок, системную функцию или системный функциональный блок		
Операция	Команда	Описание
	Операнд FC, FB, SFC, SFB и номер	
CALL	FB10, DB 10	Вызвать функциональный блок FB10 с экземпляром блока данных DB10.
CALL	SFC43	Вызвать системную функцию SFC43.

## Признак операнда

Некоторые операнды состоят из признака операнда и адреса внутри области памяти, заданной в признаке операнда. Признак операнда может принадлежать к следующим трем видам (смотрите таблицы с 8–6 по 8–8):

- Признак операнда, задающий область памяти и размер объекта данных в этой области следующим образом (смотрите таблицу 8–6):
  - Область памяти, где операция находит значение (объект данных), с которым она выполняет логическую операцию (например, "E" для отображения процесса на входах).
  - Размер значения (объекта данных), с которым операция должна выполнить логическую операцию (например, B для "байта", W для "слова" и D для "двойного слова").
- Признак операнда, который указывает область памяти, однако не указывает размер объекта данных в этой области (например, признак для области T (таймеры), Z (счетчики) или DB или DI (блок данных) и номер таймера, счетчика или блока данных) (смотрите таблицу 8–7).
- Признак операнда, который задает размер объекта данных без области памяти. Область памяти закодирована в адресе операнда, который следует за признаком операнда (смотрите таблицу 8–8).

Таблица 8–6. Признак операнда, задающий область памяти и размер объекта данных

Способ адресации	Операция	Область памяти	Признак операнда: Размер объекта данных (бит, если не задано иное)	Адрес операнда
Прямая	U	E		0.0
Прямая	L	E	B	10
Косвенная через память	U	E		[MD2]
Косвенная через память	L	E	B	[DID4]
Косвенная регистровая внутри области	U	E		[AR1, P#4.3]
Косвенная регистровая внутри области	T	L	D	[AR2, P#53.0]

Таблица 8–7. Признак операнда, задающий область памяти, но не задающий размер объекта данных

Способ адресации	Операция	Признак операнда: Область памяти	Номер или адрес номера
Прямая	AUF	DB	5
Прямая	SI	T	7
Косвенная через память	AUF	DB	[LW2]
Косвенная через память	S	Z	[MW44]

Таблица 8–8. Признак операнда, задающий размер объекта данных, но не задающий область памяти

Способ адресации	Операция	Размер объекта данных (бит, если не задано иное)	Адрес операнда
Косвенная регистровая с указанием области	U		[AR1, P#4.3]
Косвенная регистровая с указанием области	L	B	[AR1, P#100.0]

Таблица 8–9 Операнды, состоящие из признака операнда и адреса операнда

Команда			Описание
Операция	Операнд		
	Признак операнда	Адрес в области памяти или регистре	
U	E	1.0	Выполнить логическую операцию И с входным битом E 1.0.
U	E	[MD2]	Выполнить логическую операцию И с входным битом. Точный адрес находится в двойном меркерном слове MD2.
L	Z	1	Загрузить счетное значение счетчика в АККУ 1.

#### Работа со словом или двойным словом как с объектом данных

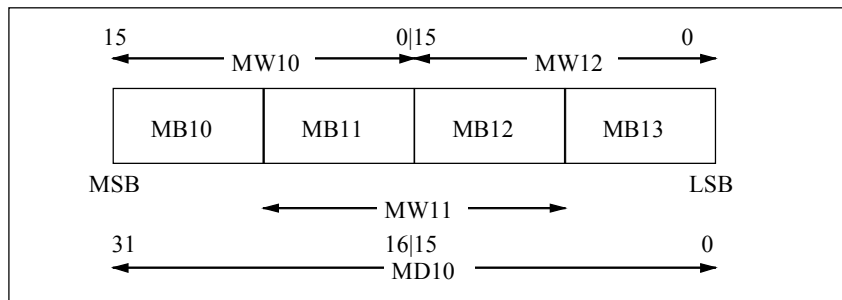
Если Вы работаете с операцией, признак операнда которой задает область памяти Вашего контроллера, и с объектом данных, который по размеру является словом или двойным словом (смотрите таблицу 8–6), то Вы должны учитывать, что на адрес памяти всегда ссылаются как на адрес

Операция: L MD10

Признак операнда      Адрес байта

Рисунок 8–3 показывает объекты данных со следующими размерами:

- Если Вы используете абсолютные операнды, которые по разрядности являются словом или двойным словом, то убедитесь, что Вам удалось избежать перекрытия байтов разных слов.



## Области памяти и их функции

AWL для S7-300/400  
C79000-G7000-C305-02

Таблица 8–10. Области памяти и их функции

Название области	Функция области памяти	Доступ к области через	
		Единицы следующих величин:	Сокр.
Отображение процесса на входах	В начале цикла операционная система считывает входные сигналы от процесса и записывает эти значения в данной области. Программа может использовать эти значения при своей циклической обработке.	Вход Входной байт Входное слово Входное двойное слово	E EB EW ED
Отображение процесса на выходах	Во время цикла программа рассчитывает выходные значения и располагает их в данной области. В конце цикла операционная система считывает рассчитанные выходные значения из этой области и передает их к выходам на процесс.	Выход Выходной байт Выходное слово Выходное двойное слово	A AB AW AD
Меркеры	Эта область предоставляет в распоряжение место в памяти для промежуточных результатов, рассчитываемых программой.	Меркер Меркерный байт Меркерное слово Меркерное двойное слово	M MB MW MD
Входы периферийной области: внеш. входы  Выходы периферийной области: внеш. выходы	Эта область дает Вашей программе возможность прямого доступа к модулям ввода и вывода (периферийные входы и выходы)	Периферийный входной байт Периферийное входное слово Периферийное входное двойное слово  Периферийный выходной байт Периферийное выходное слово Периферийное выходное двойное слово	PEB  PEW  PED PAB PAW PAD
Таймеры	Таймеры являются функциональными элементами AWL. Эта область предоставляет в распоряжение место в памяти для таймерных ячеек. В этой области генератор тактовых импульсов обращается к таймерным ячейкам для того, чтобы актуализировать их путем уменьшения значения времени. Здесь таймерные операции обращаются к таймерным ячейкам.	Таймер (T)	T
Счетчики	Счетчики являются функциональными элементами AWL. Эта область предоставляет в распоряжение место в памяти для счетчиков. Здесь к ним обращаются операции счета.	Счетчик (Z)	Z
Блок данных	В этой области содержатся данные, к которым может выполняться доступ из любого блока. Если Вы должны одновременно открыть два разных блока данных, то Вы можете открыть один из них командой "AUF DB", а другой - командой "AUF DI". Таким образом CPU сможет различать, к какому из двух блоков данных Ваша программа желает обратиться, когда оба открыты. Хотя Вы и можете с помощью команды "AUF DI" обратиться к любому произвольному блоку данных, однако эта команда преимущественно используется для открытия экземпляров блоков данных, сопоставляемых функциональным блокам (FB) и системным функциональным блокам (SFB). Дальнейшую информацию по FB и SFB возьмите из справочника программиста /234/ и справочника пользователя /231/.	Блок данных, открытый командой "AUF DB": Бит данных Байт данных Слово данных Двойное слово данных  Блок данных, открытый командой "AUF DI": Бит данных Байт данных Слово данных Двойное слово данных	DBX DBB DBW DBD   DIX DIB DIW DID



Таблица 8–10. Области памяти и их функции (продолжение)

Название области	Функция области памяти	Доступ к области через	
		Единицы следующих величин:	Сокр.
Локальные данные	Эта область содержит временные в пределах блока данные кодового блока (OB, FB или FC). Эти данные называются также динамическими локальными данными. Они служат в качестве промежуточной памяти. Когда кодовый блок закрывается, эти данные теряются. Эти данные содержатся в стеке локальных данных (L–стек).	Временные локальные данные	L
		Временный байт локальных данных	LB
		Временное слово локальных данных	LW
		Временное двойное слово локальных данных	LD

Таблица 8–11 перечисляет максимальные диапазоны адресов различных областей памяти. Эти диапазоны адресов для Вашего CPU возьмите, пожалуйста, из соответствующих технических данных CPU. Функция областей памяти пояснена в таблице 8–10.

Таблица 8–11. Области памяти и их диапазоны адресов

Название области	Доступ к области через		Максимальный диапазон адресов
	Единицы следующих величин:	Сокращ.	
Отображение процесса на входах	Вход	E	c 0.0 по 65 535.7
	Входной байт	EB	c 0 по 65 535
	Входное слово	EW	c 0 по 65 534
	Входное двойное слово	ED	c 0 по 65 532
Отображение процесса на выходах	Выход	A	c 0.0 по 65 535.7
	Выходной байт	AB	c 0 по 65 535
	Выходное слово	AW	c 0 по 65 534
	Выходное двойное слово	AD	c 0 по 65 532
Меркеры	Меркер	M	c 0.0 по 65 535.7
	Меркерный байт	MB	c 0 по 65 535
	Меркерное слово	MW	c 0 по 65 534
	Меркерное двойное слово	MD	c 0 по 65 532
Входы периферийной области:	Периферийный входной байт	PEB	c 0 по 65 535
	Периферийное входное слово	PEW	c 0 по 65 534
	Периферийное входное двойное слово	PED	c 0 по 65 532
	Периферийный выходной байт	PAB	c 0 по 65 535
Выходы периферийной области: Внешние выходы	Периферийное выходное слово	PAW	c 0 по 65 534
	Периферийное выходное двойное слово	PAD	c 0 по 65 532
Таймер	Таймер	T	c 0 по 65 535
Счетчик	Счетчик	Z	c 0 по 65 535
Блок данных	Блок данных, открытый командой "AUF DB": Бит данных в блоке данных Байт данных Слово данных Двойное слово данных	DBX DBB DBW DBD	c 0.0 по 65 535.7 c 0 по 65 535 c 0 по 65 534 c 0 по 65 532
	Блок данных, открытый командой "AUF DI": Бит данных в экземпляре DB Байт данных Слово данных Двойное слово данных	DIX DIB DIW DID	c 0.0 по 65 535.7 c 0 по 65 535 c 0 по 65 534 c 0 по 65 532
Локальные данные	Бит локальных данных	L	c 0.0 по 65 535.7
	Байт локальных данных	LB	c 0 по 65 535
	Слово локальных данных	LW	c 0 по 65 534
	Двойное слово локальных данных	LD	c 0 по 65 532

## 8.2. Значение регистров CPU в командах

**Аккумуляторы** Аккумуляторы (AKKU) являются универсальными регистрами для обработки байтов, слов и двойных слов. Они имеют разрядность 32 бита. Вы можете загружать константы и значения из памяти в качестве операндов в аккумулятор и там их связывать. Вы можете также передать результат операции из AKKU 1 по адресу операнда. Рисунок 8–4 показывает области аккумулятора.

Механизм стека для управления аккумулятором выглядит следующим образом:

- Операция загрузки всегда воздействует на AKKU 1 и сохраняет старое содержимое в AKKU 2.
- Операция передачи не изменяет аккумуляторы (за исключением операций TAR1 и TAR2).
- Операция ТАК обменивает содержимое AKKU 1 и 2.

Информацию об управлении аккумуляторами при арифметических операциях возьмите из главы 15.1.

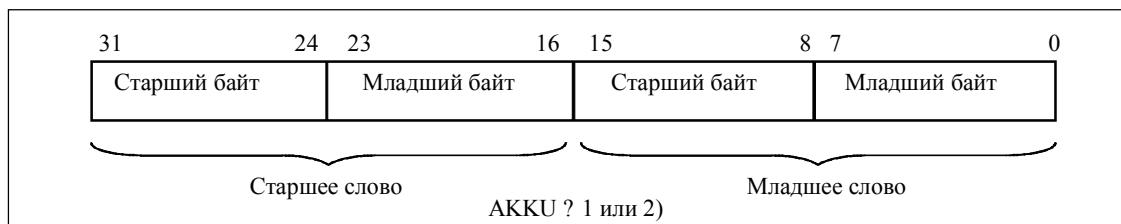


Рис. 8-4. Области аккумулятора

### Стек скобок

Стек скобок является областью памяти, которую используют логические операции скобочных выражений U(, O(, X(, UN(, ON( и XN(. Он имеет разрядность, равную байту. Эти операции запоминают текущий результат логической операции (VKE) в стеке скобок и начинают новую цепь логических операций.

Стек скобок может принимать до семи записей. Запись в стеке скобок состоит из битов VKE, BIE и OR слова состояния и функционального кода, задающего то, какая логическая операция должна выполняться (U, UN, O, ON, X или XN).

Операция ) завершает скобочное выражение и выполняет следующие функции:

- Извлекает запись из стека скобок.
- Извлекает биты OR и BIE.
- Определяет новый VKE путем логического сопряжения текущего VKE (то есть VKE скобочного выражения) с VKE стековой записи согласно функциональному коду (смотрите главу 11.4).

Рисунок 8–5 показывает структуру записи в стеке скобок. Вслед за рисунком следует объяснение битов байта стека скобок.

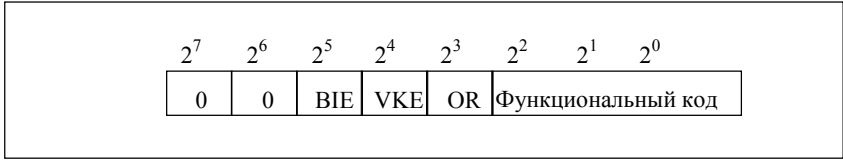


Рис. 8-5. Структура записи в стеке скобок

Байт стека скобок содержит следующие биты (смотрите также рис. 8–5):

- не назначенные биты (биты 7 и 6 с состоянием сигнала ”0”),
- запоминаемый двоичный результат (BIE),
- запоминаемый результат логической операции (VKE),
- запоминаемый OR–бит в функциях U( и UN(. В любой другой функции запоминается ноль,
- функциональный код (в битах 2, 1 и 0).

**Функциональный код**

Операция ) на основании функционального кода определяет функцию, которая должна использоваться для логического сопряжения скобочного VKE с VKE из скобочной записи. Таблица 8–12 перечисляет комбинации битов функционального кода для каждого вида функции.

Таблица 8–12. Функциональный код байта стека скобок

Функция операции	Функциональ- ный код 2	Функциональ- ный код 1	Функциональ- ный код 0
U(	0	0	0
UN(	0	0	1
O(	0	1	0
ON(	0	1	1
X(	1	0	0
XN(	1	0	1

## Стек скобок с записями и указателем

Стек скобок, а также указатель стека скобок должны запоминаться в стеке прерываний или извлекаться из стека прерываний, когда сменяются уровни. Число в указателе стека скобок указывает, сколько записей имеется в стеке скобок (смотрите рисунок 8–6).

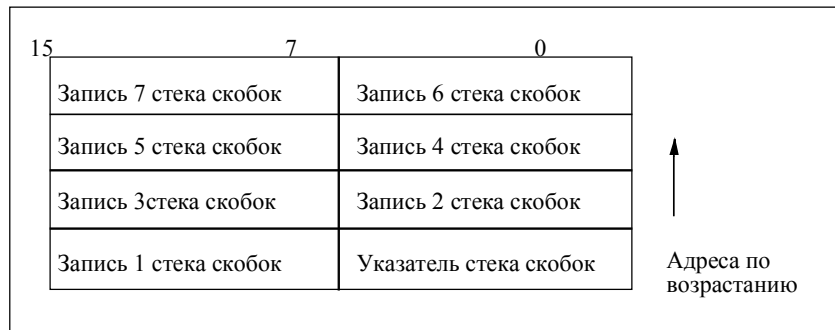


Рис. 8-6. Структура стека скобок с записями и указателем

## Слово состояния

Слово состояния содержит биты, к которым Вы можете обратиться в операндах операций логического сопряжения битов и слов. Рисунок 2–2 показывает структуру слово состояния. Следующая глава объясняет значение битов с 0 по 8.

$2^{15} \dots$	$\dots 2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
		BIE	A1	A0	OV	OS	OR	STA	VKE	/ER

Рис. 8-7. Структура слова состояния

## Первичный опрос

Бит 0 слово состояния называется битом первичного опроса (/ER–бит: смотрите рисунок 2–2). Состояние сигнала "0" в /ER–бите указывает, что следующая логическая операция в Вашей программе начинает новую цепь логических операций. (Косая черта перед аббревиатурой ER указывает, что /ER–бит берется с отрицанием).

Каждая логическая операция опрашивает состояние сигнала /ER–бита и операнда, к которому происходит обращение. Если /ER–бит равен "0", то операция запоминает результат опроса состояния сигнала в VKE–бите слова состояния (о VKE–бите: смотрите следующую главу) и устанавливает /ER–бит в 1. Такой процесс называется первичным опросом (смотрите рисунок 8–8 и главу 11.6).

Если состояние сигнала /ER–бита равно "1", то операция выполняет логическое сопряжение результата своего опроса состояния сигнала на контакте, к которому она обращается, со значением, которое запоминалось в предыдущем VKE–бите (смотрите рисунок 8–8).

Цепь логических операций всегда заканчивается операцией вывода (S, R или =, смотрите главы 11.7 и 11.8), операцией перехода, которая ссылается на результат логической операции (SPB, смотрите главу 22) или скобочным выражением U(, O(, X( UN(, ON( или XN( (смотрите главу 11.4). Эти операции сбрасывают /ER–бит в "0" (смотрите рисунок 8–8).

**Результат  
логической  
операции**

Бит 1 слова состояния называется VKE-битом (VKE обозначает “результат логической операции”, смотрите рисунок 8–7). Этот бит запоминает результат логической операции или операции сравнения.

Вторая операция в цепи логических операций опрашивает, в частности, состояние сигнала контакта и порождает результат ”1” или ”0”. Операция теперь логически связывает этот результат по правилам булевой логики со значением, которое записано в VKE-бите (смотрите выше ”первичный опрос” и главу 11). Результат этой логической операции запоминается в VKE-бите и заменяет предыдущее значение VKE-бита. Каждая последующая операция в цепи выполняет логическое сопряжение с использованием двух значений: результата опроса сигнала на контакте и текущего VKE.

Вы можете устанавливать абсолютно в “1” с помощью команды SET или сбрасывать VKE абсолютно в “0” с помощью команды CLR. С помощью логической операции при первичном опросе Вы можете присваивать VKE состояние содержимого битовой ячейки памяти. С помощью VKE Вы можете запускать операции перехода.


Программа на AWL	Состояние сигнала на входе (E) или выходе (A)	Результат опроса	VKE-бит	/ER- бит	Объяснение
				0	/ER-Bit = 0 указывает, что следующая операция начинает цепь логич. операц.
U E 1.0	1	1	1		Результат первичного опроса сохраняется в VKE-бите. /ER-бит устанавлив. в “1”
UN E 1.1	0	1	1	1	Результат опроса сопрягается с предыдущим VKE в соответствии с табл. истин. для И. /ER-бит остается равным “1”.
= A 4.0	1			0	VKE присваивается выходной катушке. /ER-бит сбрасывается в “0”.

Рис. 8-8. Влияние состояния сигнала /ER-бита на логические операции

**Бит состояния**

Бит состояния (STA) запоминает значение бита, к которому происходит обращение. Состояние логической операции, обращающейся к памяти для чтения (U, UN, O, ON, X или XN), всегда равно значению бита, опрашиваемого данной операцией (бита, с которым она выполняет свое логическое сопряжение). Состояние логической операции, обращающейся к памяти для записи (S, R, =), равно значению бита, в который операция производит запись. Если запись не происходит, то оно равно значению бита, к которому происходит обращение. Бит состояния не имеет значения для логических операций, не обращающихся к памяти. Эти операции устанавливают бит состояния в”1” (STA = 1). Бит состояния не опрашивается операциями. Он используется только при тестировании программы (состояние программы).

**OR–бит**

**OR–бит** (ODER) требуется, когда Вы с использованием операции **O** выполняете логическое сопряжение по **И** перед логической операцией **ИЛИ**. Логическое сопряжение по **И** может содержать следующие операции: **U**, **UN**, **U(**, **UN(**, **)** и **NOT**. **OR–бит** показывает этим операциям, что ранее выполненное логическое сопряжение по **И** доставило значение “1”, чем предвосхищается результат логической операции **ИЛИ**. Любая другая операция, обрабатывающая бит, сбрасывает **OR–бит** (смотрите главу 11.4).

**OV–бит**

**OV–бит** (переполнение) указывает на ошибку. Он устанавливается арифметической операцией или операцией сравнения над числами с плавающей точкой после того, как возникла ошибка (переполнение, недопустимая операция, недопустимое число с плавающей точкой). Этот бит устанавливается соответственно результату ближайшей арифметической операции или операции сравнения

**OS–бит**

**OS–бит** (переполнение с запоминанием) устанавливается вместе с **OV–битом**, когда появляется ошибка. Так как **OS–бит** остается установленным также и после устранения ошибки, он запоминает состояние **OV–бита** и показывает, появлялась ли ошибка в одной из ранее выполнявшихся операций. Следующие операции сбрасывают **OS–бит**: **SPS** (переход, если **OS = 1**), вызов блока, окончание блока.

**A1 и A0**

Биты **A1** и **A0** (индикаторные биты) информируют о следующих результатах или битах:

- Результат арифметической операции.
- Результат операции сравнения.
- Результат цифровой операции.
- Биты, которые подверглись сдвигу вследствие операции сдвига или циклического сдвига.

Таблицы с 8–13 по 8–18 перечисляют значение **A1** и **A0** после того, как Ваша программа выполнила определенные операции.

Таблица 8–13. **A1** и **A0** после арифметических операций без переполнения

<b>A1</b>	<b>A0</b>	<b>Объяснение</b>
0	0	Результат = 0
0	1	Результат < 0
1	0	Результат > 0

Таблица 8–14. **A1** и **A0** после арифметических операций (арифметика с фиксированной точкой) с переполнением

<b>A1</b>	<b>A0</b>	<b>Объяснение</b>
0	0	Отрицательный выход из области значений при +I и +D
0	1	Отрицательный выход из области значений при *I и *D Положительный выход из области значений при +I, –I, +D, –D, NEGI и NEGD
1	0	Положительный выход из области при *I, *D, /I и /D Отрицательный выход из области при +I, –I, +D и –D
1	1	Деление на 0 в /I, /D и MOD

Таблица 8–15. **A1** и **A0** после арифметических операций (арифметика с плавающей точкой) с переполнением

<b>A1</b>	<b>A0</b>	<b>Объяснение</b>
-----------	-----------	-------------------

0	0	Потеря значимости
0	1	Отрицательный выход из области значений
1	0	Положительный выход из области значений
1	1	Недействительное число с плавающей точкой

Таблица 8–16. A1 и A0 после операций сравнения

A1	A0	Объяснение
0	0	AKKU 2 = AKKU 1
0	1	AKKU 2 < AKKU 1
1	0	AKKU 2 > AKKU 1
1	1	AKKU 1 или AKKU 2 не является действительным числом с плавающей точкой

Таблица 8–17 . A1 и A0 после операций сдвига и циклического сдвига

A1	A0	Объяснение
0	0	Последний сдвинутый бит = 0
1	0	Последний сдвинутый бит = 1

Таблица 8–18. A1 и A0 после цифровых логических операций

A1	A0	Объяснение
0	0	Результат = 0
1	0	Результат $\neq$ 0

## **ВІЕ–бит**

ВІЕ–бит является связующим звеном между обработкой битов и слов. Он эффективным образом дает возможность двоичной интерпретации результата операции над словами и его включения в двоичную цепь логических операций. При таком способе рассмотрения ВІЕ представляет внутримашинный меркер, в котором перед операцией над словами, изменяющей VKE, сохраняется VKE, чтобы после этой операции он снова имелся в распоряжении для продолжения прерванной битовой цепи.

ВІЕ–бит дает Вам, в частности, возможность программировать функциональный блок (FB) или функцию (FC) в AWL и вызывать этот FB или FC в KOP (по KOP смотрите справочник /233/).

Если Вы желаете записывать функциональный блок или функцию в KOP или AWL и вызывать в KOP, то Вы должны управлять ВІЕ–битом. ВІЕ–бит соответствует разрешающему выходу (ENO) для KOP–блока. С помощью операции SAVE (в AWL, смотрите главу 11.9) или с помощью “катушки” Вы сохраняете VKE в ВІЕ–бите по следующим критериям:

- Запоминайте VKE со значением 1 в ВІЕ–бите, когда FB или FC был обработан без ошибки.
- Запоминайте VKE со значением 0 в ВІЕ–бите, когда FB или FC был обработан с ошибкой.

Программируйте эти операции в конце FB или FC так, чтобы это были последние операции, которые выполняются в блоке.

Если Вы вызываете в Вашей программе системный функциональный блок (SFB) или системную функцию (SFC), то SFB или SFC через состояние сигнала ВІЕ–бита показывает, выполнил ли CPU функцию без ошибки или с ошибкой:

- Если при обработке возникла ошибка, то ВІЕ–бит равен ”0”.
- Если функция была обработана без ошибки, то ВІЕ–бит равен ”1”.