Синтаксис и правила для исходных файлов

7

Что описывает эта глава?

Исходные файлы можно компилировать в блоки только тогда, когда соблюдаются определенные структуры и синтаксические правила. Эта глава представляет структуры для кодовых блоков и блоков данных и содержит синтаксические правила включая необходимые ключевые слова.

Защиту блоков и защиту от записи нельзя создать в инкрементном редакторе AWL, это можно сделать только при программировании в исходном файле. Эта глава описывает, как нужно вводить атрибуты блока, например, для защиты блока или защиты от записи.

Если Вы хотите быстро и просто редактировать исходный код для новых блоков, Вам лучше всего пользоваться шаблонами блоков.

Обзор главы

7.1. Общие правила ввода и структура

Правила ввода

Для создания прикладных программ в виде исходного файла действуют следующие правила:

- Синтаксис команд AWL тот же, что и в инкрементном редакторе AWL. Исключение составляют только вызовы блоков и описание массивов и структур.
- Для текстового редактора в общем безразлично **написание большими или малыми буквами.** Исключение - имена переменных. Например, имена переменных ”Motor_ein” и ”MOTOR_EIN” различны. При вводе последовательностей символов (тип данных STRING) также нужно обращать внимание на написание большими или малыми буквами.
- Конец каждой команды AWL и каждого описания переменной отмечайте точкой с запятой (;). В строке можно вводить более одной команды.
- Каждый комментарий начинайте двумя косыми чертами (//) и завершайте каждый ввод комментария нажатием клавиши RETURN.

Последовательность блоков

Что касается последовательности блоков, то при создании исходного файла Вы должны обратить внимание на следующее:

Вызываемые блоки стоят перед вызывающими. Это значит:

- Чаще всего используемый OB1, который вызывает другие блоки, стоит последним. Блоки, которые в свою очередь вызываются из блоков, вызываемых из OB1, должны стоять перед ними и т.д.
- Типы данных, определенные пользователем (UDT), стоят перед блоками, в которых они используются.
- Блоки данных с соответствующим типом данных, определенным пользователем (UDT) стоят после UDT.
- Глобальные блоки данных стоят перед всеми блоками, из которых они вызываются.
- Блоки данных, соответствующие функциональному блоку, стоят за этим функциональным блоком.

Блоки, вызываемые в исходном файле, но в нем не программируемые, к моменту компиляции файла должны уже находиться в соответствующей прикладной программе.

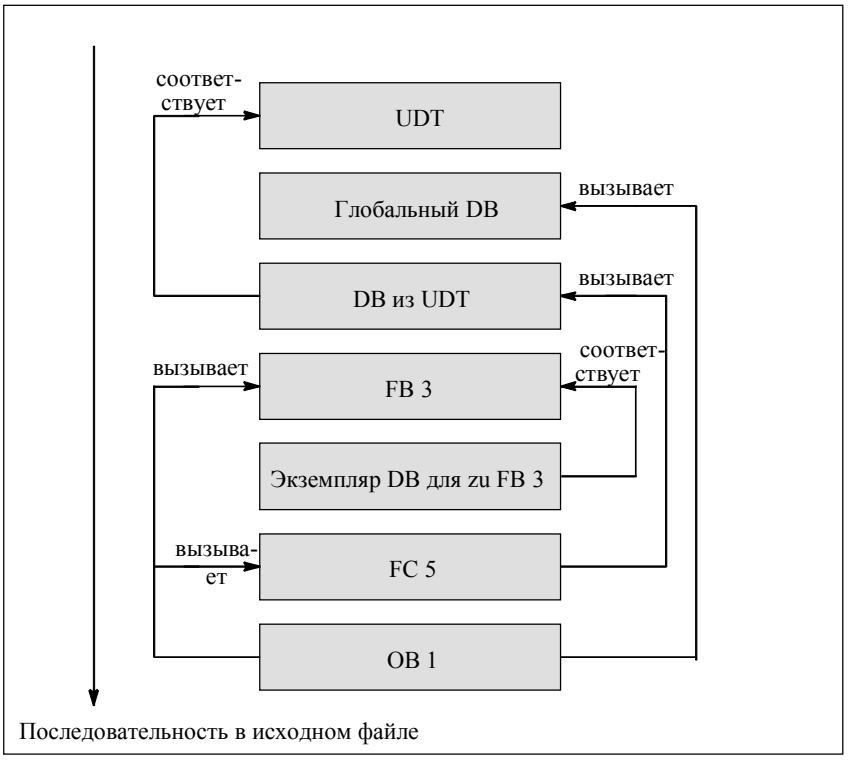

Рис. 7-1. Структура блоков исходного файла (пример)

Общая структура блоков

Исходный код для блока в принципе состоит из следующих разделов:

- Начало блока с указанием блока (абсолютным или символическим)
- Заголовок блока (не обязательно)
- Комментарий к блоку (не обязательно)
- Атрибуты блока (не обязательно)
- Раздел описаний (различен в зависимости от вида блока)
- Операторная часть в кодовых блоках
  или указание текущих значений в блоках данных (не обязательно)
- Конец блока

7.2. Общий синтаксис для кодовых блоков

Структура

Кодовый блок состоит из следующих областей, которые по мере надобности обозначаются соответствующими ключевыми словами:

кодового блока

Таблица 7–1. Структура кодового блока

Структура	Ключевое слово с *примером*
Начало блока с указанием блока (абсолютным или символическим)	ORGANISATION_BLOCK *OB1* FUNCTION_BLOCK *FB6* FUNCTION *FC 1* : *int*
Заголовок блока (не обязательно)	TITLE = *Bausteintitel [Заголовок блока]*
Комментарий к блоку (не обязательно)	//*Комментарий к блоку*
Свойства блока (не обязательно)	KNOW_HOW_PROTECT AUTHOR: *Meier* FAMILY: *Motoren* NAME: *Motoreins* VERSION: *01.09*
Раздел описания переменных (Типы описаний в зависимости от вида блока) Завершение каждого типа описания	VAR_INPUT VAR_OUTPUT VAR_IN_OUT VAR VAR_TEMP : END_VAR
Операторная часть, состоящая из сетей заголовков сетей комментариев к сетям	BEGIN NETWORK *TITLE = первая сеть* // : NETWORK :
Конец блока	END_ORGANIZATION_BLOCK END_FUNCTION_BLOCK END_FUNCTION

Правила

При вводе кодового блока следует соблюдать следующие правила:

- В начале блока между ключевым словом для вида блока и указанием блока должен быть пробел. При указании символического имени блока оно может быть обозначено кавычками, чтобы гарантировать однозначность между именами локальных переменных и именами в таблице символов.
- У функций (FC), кроме того, указывается тип функции. Он может быть элементарным или составным и определяет тип данных возвращаемого значения (RET_VAL). Если никакое значение возвращаться не должно, то указывается VOID.
- Задание номера сети не допускается.

Синтаксис для вызова блока с “CALL”

Синтаксис для вызова FB и FC командой CALL слегка отличается от используемого в инкрементном редакторе AWL. В исходном файле **помощью** параметры указываются в скобках. Отдельные параметры при этом отделяются друг от друга запятыми.

Пример: CALL FC1 (param1 := E 0.0, param2 := E0.1);

Комментарии в операторной части

Чтобы при последующем редактировании в инкрементном редакторе обеспечить представление комментариев 1:1, примите во внимание следующее:

- Вызов блока: В исходных файлах при присвоении фактических параметров формальным Вы должны сохранить ту же последовательность формальных параметров, что и в описании переменных блока. Правда, в принципе последовательность параметров произвольна, однако комментарии к параметрам при компиляции исходного файла в блок могут поменяться местами.
- У команд для доступа к блокам данных, непосредственно следующих за соответствующей командой ”AUF”, при компиляции в блоки может произойти потеря комментариев к командам. Чтобы избежать этого,
  - программируйте в компактном представлении (напр., L DB5.DBW20; //комментарий)
  - или вставьте команду ”NOP” (напр., AUF DB5; //комметарий 1
NOP 0;
L DBW20; //комментарий 2).

Пример

Далее следует сокращенный пример для организационного блока:

```
ORGANIZATION_BLOCK OB 1
TITLE =
VERSION : 0.0

VAR_TEMP
   dummy_array     : ARRAY [1..20] OF INT ;
   start_switch : BOOL ;
   dummy              : INT ;
END_VAR
BEGIN

NETWORK
TITLE =
   CLR ;
   =       ”synchron_snd_bit”;
NETWORK
TITLE =
   BLD   3;
   TDB   ;
   AUF   DI      6;
   :
END_ORGANIZATION_BLOCK
```

7.3. Общий синтаксис для блоков данных

Структура блока данных

Блок данных состоит из следующих областей, которые при необходимости вводятся соответствующими ключевыми словами:

Таблица 7–2. Структура блока данных

Структура	Ключевое слово с *примером*
Начало блока с указанием блока (абсолютным или символическим)	DATA_BLOCK *DB 26*
Заголовок блока (не обязательно)	TITLE = *Bausteintitel [Заголовок блока]*
Комментарий к блоку (не обязательно)	//*Комментарий к блоку*
Свойства блока (не обязательно)	KNOW_HOW_PROTECT AUTHOR: *Meier* FAMILY: *Motoren* NAME: *Dateneins* VERSION: *12.13*
Раздел описаний - в зависимости от DB:	
Глобальный блок данных: Описание переменных (при желании с указанием начальных значений)	STRUCT : END_STRUCT
DB из UDT: Указание UDT (абсолютное или символическое)	UDT *16*
Экземпляр DB: Указание FB (абсолютное или символическое)	FB *20*
Раздел назначений с текущими значениями	BEGIN : :
Конец блока	END_DATA_BLOCK

Правила

При вводе блоков данных Вы должны придерживаться следующих правил:

- DB 0 занят. Таким образом, Вы не можете создать DB с таким именем.
- Текущие значения Вы можете по выбору указать для одной или нескольких переменных. Переменным, которым Вы не назначили текущих значений, назначаются - если имеются - начальные значения, иначе - значения, предустановленные для соответствующего типа данных.
- Комментарии к командам в разделе назначений для текущих значений (между ”BEGIN” и ”END_DATA_BLOCK”) после компиляции в блоки в инкрементном редакторе не отображаются. Поэтому комментируйте блоки данных только в разделе описаний.

Примеры

Блок данных

```
data_block db 26
```

```
struct
    aa      :   bool := False;          //Начальные значения
    bb      :   int  := 100;
    cc      :   word;
end_struct;
begin
    aa      := false;   //Текущие значения
    bb      := 1500;    //Эти комментарии выпадают.
end_data_block
```

DB с соответствующим типом данных, определенным пользователем

```
data_block db 25
udt 54                                  //Указание UDT
begin
    Buffer := W#16#1111;
    Today := DATE#1994-7-18;
end_data_block
```

DB с соответствующим функциональным блоком

```
data_block db 34
fb 21                                   //Указание FB
begin
    b       := false;
    d       := false;
end_data_block
```

DB с соответствующим функциональным блоком, в котором описаны мультиэкземпляры

```
function_block ”BACKEN”
TITLE =
VERSION : 0.0
VAR
stat_fb : FB 1;
END_VAR
BEGIN
END_FUNCTION_BLOCK
DATA_BLOCK ”REZEPT”
TITLE =
VERSION : 0.0
”BACKEN”                                //Соответствие FB ”BACKEN”
BEGIN
stat_fb.feld[1] := 1;               //Начальные значения для
stat_fb feld[2] := 2;  // переменных FB1, который в FB
stat_fb feld[3] := 3;  //”BACKEN” описан как
                    // мультиэкземпляр.
end_data_block
```

7.4. Общий синтаксис для типов данных, определенных пользователем

Структура типа данных, определенного пользователем (UDT)

Тип данных, определенный пользователем, состоит из следующих областей, которые при необходимости вводятся соответствующими ключевыми словами:

Таблица 7−3. Структура типа данных, определенного пользователем (UDT)

Структура	Пример
Начало блока с указанием блока (абсолютным или символическим)	TYPE *UDT 14*
Раздел описаний: Задание структурного типа данных (при желании с указанием начальных значений)	STRUCT : END_STRUCT
Конец блока	END_TYPE

Пример

Следующий пример показывает программирование типа данных, определенного пользователем, в исходном файле:

```
TYPE udt 54
struct
    Buffer :    word   := W#16#4D2;
    Today :    date    := DATE#1994-7-14;
    Temp       :    time_of_day       := time_of_day#11:13:2.0;
end_struct;
END_TYPE
```

7.5. Правила описания переменных

Обзор

В каждом блоке исходного файла Вы должны описать соответствующие переменные. В то время как в инкрементном редакторе Вы заполняете таблицу, здесь Вы должны работать с соответствующими ключевыми словами.

Типы описания в кодовых блоках

У кодовых блоков тип описания переменных обозначается ключевым словом, которое стоит в собственной строке. В зависимости от вида блока допустимы **только определенные типы описаний** (табл. 7–4).

Таблица 7–4. Типы описаний в кодовых блоках

Тип описания	Ключевое слово	OB	FB	FC
Входной параметр	VAR_INPUT	-	Да	Да
Выходной параметр	VAR_OUTPUT	-	Да	Да
Проходной параметр	VAR_IN_OUT	-	Да	Да
Статическая переменная	VAR	-	Да	-
Временная переменная	VAR_TEMP	Да	Да	Да
По мере надобности завершение	END_VAR			

Правила ввода

При вводе описания переменной Вы должны принимать во внимание следующее:

- Переменные - те, что имеются - должны описываться в названном порядке типов описания. Благодаря этому все переменные, имеющие один и тот же тип описания, стоят вместе.
- Ключевые слова, смотря по обстоятельствам, стоят в собственной строке или разделяются пробелом.
- Имя переменной стоит в начале команды и должно начинаться с буквы. Оно не должно совпадать ни с одним из зарезервированных ключевых слов (см. приложение C).
- Тип данных указывается за именем переменной и отделяется от него двоеточием. Допустимы элементарные, составные и определенные пользователем типы данных.
- Массивы вводятся ключевым словом ARRAY. Кроме того, в квадратных скобках указывается размерность и тип данных (см. пример).
- У структур описание переменных находится между ключевыми словами STRUCT и END_STRUCT.
- Каждое описание переменной оканчивается точкой с запятой.
- Комментарии отделяются от раздела описаний двумя косыми чертами.

Определение начальных значений

При желании Вы можете для каждой переменной определить начальное значение. Если Вы отказываетесь от этого, то в качестве начального значения принимается значение по умолчанию, определяемое типом переменной. Начальное значение указывается в описании вслед за типом данных и отделяется от него двоеточием и знаком равенства.

Примеры

Ниже приведены примеры описания переменных различного типа:

```
VAR_INPUT
   input1 :   int;     //элементарные типы данных
   input2 :   word;
   input3 :   dword;
   Feld1  :   ARRAY [1..20,1..40] of int;
END_VAR
VAR_TEMP
struktur1: STRUCT
   var1   :   bool := False;          //Структура
   var2   :   int  := 93;
   var3   :   word;
end_struct;
END_VAR
```

Комментарии в разделе описаний

Чтобы добиться однозначного 1:1 представления комментариев в инкрементном редакторе после компиляции исходного файла в блоки, Вы должны придерживаться следующих правил:

- Ограничивайте свои комментарии в описании переменной одной строкой (т.е. максимум 80 символов) Многострочные комментарии в таблице описания переменных отображаются не полностью.
- Если Вы в разделе описаний установили начальные значения переменных, комментируйте их в конце строки после точки с запятой.

  Комментарии на других местах (напр., после ключевого слова VAR_INPUT, VAR_OUTPUT, и т.д. или после END_VAR) в таблице описания переменных инкрементного редактора не отображаются (исключение: комментарии к структурам и массивам).
- Вводите комментарии для переменных типа структуры или массива после ключевого слова, а не после описания.

  Иначе комментарий не отображается в таблице описания переменных. Таким образом, у массивов вводите комментарий после ARRAY[...]; тип данных вводите в следующей строке. У структур комментарий стоит после STRUCT и/или после каждого элемента структуры, но не после END_STRUCT.

7.6. Задание свойств блока

Обзор

При желании Вы можете ввести свойства для блоков Вашего исходного файла. Особенность задания атрибутов блока состоит в том, что это можно сделать только при программировании в текстовом редакторе. Все другие свойства блока, как, например, потребность в памяти или метка времени, которые видны в инкрементном редакторе, вводятся с помощью STEP 7.

Автор, семейство, имя и версия

Имя, семейство, версию и автора блока можно задать с помощью ключевых слов. При этом действуют следующие правила:

- Свойства блока задаются **перед разделом описания переменных**.
- Каждое свойство блока стоит в отдельной строке или же ключевые слова разделяются пробелами.
- В конце строки нет (!) точки с запятой.
- Вы можете задать одно, несколько или ни одного свойства блока.
- Если свойства задаются, то они должны быть указаны в последовательности, приведенной в таблице 7–5.

Таблица 7–5. Свойства блока

Ключевое слово	Значение
AUTHOR :	Имя автора: название фирмы, отдела или другое имя (максимум 8 символов)
FAMILY :	Название семейства блоков: например, Motoren. При программировании на языке КОР заданное название семейства указывается дополнительно к абсолютному имени блока при вводе элементов КОР. Это может облегчить Вам выбор блока. (максимум 8 символов).
NAME :	Имя, которое дополнительно указывается при вводе блока как элемента КОР, с тем, чтобы Вы идентифицировали его не только с помощью абсолютного имени. (максимум 8 символов)
VERSION :	Номер версии блока (первое число = 0..15, второе число = 0..15)

Защита блока

Вы можете организовать защиту для кодовых блоков и блоков данных, указав ключевое слово KNOW_HOW_PROTECT. Защита блока имеет нижеперечисленные последствия:

- Если ВЫ просматриваете скомпилированный блок в инкрементном редакторе AWL или КОР, операторная часть блока не видна.
- В описании переменных блока отображаются только входные, выходные и проходные параметры (типы описания var_input, var_output и var_in_out). Внутренние переменные, имеющие типы описания var и var_temp остаются скрытыми.
- Скомпилированный блок, правда, можно декомпилировать в исходный файл, но только как блок без операторной части.

Ключевое слово KNOW_HOW_PROTECT должно вводиться перед всеми другими атрибутами блока.

Защита блоков данных от записи

В исходных файлах можно организовать защиту блоков данных от записи, так что при исполнении программы сохраненные в них данные не могут быть заменены. Для этого введите ключевое слово READ_ONLY. Оно должно стоять в непосредственно перед описаниями в собственной строке.

Атрибут ”Unlinked”

Атрибут ”Unlinked” (“Не связан”) может быть только у блоков данных. Он означает, что блок данных не загружается из загрузочной памяти в рабочую память CPU. Обращение к блокам данных, находящимся в загрузочной памяти, осуществляется через SFC, которые копируют в рабочую память только содержимое DB. Благодаря этому достигается лучшее использование рабочей памяти, которая, таким образом, содержит только данные, существенные для исполнения программы.

Резюме

В таблице 7–6 приведен обзор того, какие свойства и атрибуты в каких блоках можно задавать. Обратите внимание на последовательность.

Таблица 7–6. Обзор атрибутов блока

Атрибут	Кодовые блоки (OB, FB, FC)	Блоки данных	UDT
KNOW_HOW_PROTECT	да	да	нет
AUTHOR	да	да	нет
FAMILY	да	да	нет
NAME	да	да	нет
VERSION	да	да	нет
UNLINKED	нет	да	нет
READ_ONLY	нет	да	нет

Пример

```
FUNCTION_BLOCK FB 6
TITLE =
KNOW_HOW_PROTECT        //Защита блока
FAMILY : Regler
VERSION : 2.05
//Описание переменных
//Команды
END_FUNCTION_BLOCK


DATA_BLOCK DB 16
TITLE =
KNOW_HOW_PROTECT
AUTHOR : Siemens
VERSION : 11.03
READ_ONLY               //Защита от записи
//Описание переменных
//Назначения
END_DATA_BLOCK
```