22 Язык программирования Структурированный Текст

22.1 Введение

ST программа представляет из себя список ST операторов. Каждый оператор заканчивается двоеточием-разделителем (:). Имена использующиеся в операторах (переменные, константы, ключевые слова) разделяются неактивными разделителями (символ пробела, EOL, табуляция), а также специальными символами (например, > "больше чем"). Комментарии могут быть включены в текст и заключены в звездочки.

Программа должна начинаться с ключевого слова `Program` за которым следует имя программы, и заканчиваться ключевым словом End_Program.

Блок объявления типов данных может начинаться с ключевого слова `Var, Var_Input, Var_Output, Var_In_Out` и заканчиваться ключевым словом `End_Var.`

Пример

```
PROGRAM Noname;
VAR
   I : integer;
   T : Timer;
END_VAR;
   TStart(T);
   I:=Test;
   TStop(T);
   Tim:=T;
   TContinue(T);
END_PROGRAM;
```

22.2 Типы операторов

Имеются следующие типы ST операторов:

- операторы воздействия на переменную (`переменная:= выражение:);`
- условные операторы `(IF,THEN,ELSE);`
- итерационные операторы `(FOR,WHILE...);`
- управляющие операторы `(RETURN...);`
- специальные операторы для связи с другими языками.

22.3 Выражения

ST выражения является комбинацией ST операторов и переменных (или констант) операндов. Круглые скобки используют для выделения части выражения и задания определенного порядка выполнения операций.

Например:

```
2+3*6     дает значение 20
(2+3)*6   дает значение 30
```

Для каждого одиночного выражения (комбинации операндов с одним ST оператором) все операнды должны иметь один тип.

Например:

```
(boo_var1 AND boo_var2)  имеет тип boolean
not(boo_var1)  имеет тип boolean
(sin(3.14) + 0.72)  имеет тип Float
(1s23 + 1.78)  является недействительным (ошибочным) выражением.
```

22.4 Список операций языка ST в порядке убывания приоритета

№	Операция	Имя
1	Парные круглые скобки	(выражение)
2	Вычисляемые функции	Идентификатор(список аргументов)
	Примеры:	`LN(A), MAX (X, Y), и т.п.`
3	Возведение в степень	**
4	Отрицание	_
5	Логическая инверсия	NOT
6	Умножение	*
7	Деление	/
8	Модуль	MOD
9	Сложение	+
10	Вычитание	-
11	Сравнение	< , > , <= , >=
12	Равенство	=
13	Неравенство	<>
14	Логическое И	&
15	Логическое исключающее ИЛИ	XOR
16	Логическое ИЛИ	OR

22.5 Арифметические операции

**	возведение в степень
*	умножение
/	деление
+	сложение
-	вычитание
_	отрицание
MOD	модуль
SQRT	извлечение квадратного корня
SIN	синус
COS	косинус
TAN	тангенс

Операция ***синус***
Обозначение sin
Назначение: синус угла в радианах
Синтаксис: <переменная> := sin<выражение>;
Операнд: выражение типа FLOAT.
Возвращаемое значение имеет тот же тип, как исходное.

Пример
a := sin (pi/2);
где
pi число **ПИ**

Операция **косинус**

Обозначение cos
Назначение: косинус угла в радианах
Синтаксис: <переменная> := cos<выражение>;
Операнд: выражение типа FLOAT.
Возвращаемое значение имеет тот же тип, как исходное.

Операция ***тангенс***
Обозначение tan
Назначение: тангенс угла в радианах
Синтаксис: <переменная> := tan<выражение>;
Операнд: выражение типа FLOAT.
Возвращаемое значение имеет тот же тип, как исходное.

Операция *возведение в степень*
Имя: **
Назначение: возведение в степень
Синтаксис: `<переменная> := <выражение>**;`
Операнд: выражение типа `INTEGER, SHORTINT, LONGINT, FLOAT, WORD, DWORD`.
Возвращаемое значение имеет тот же тип, как исходное.

Операция *извлечение квадратного корня*
Имя: `SQRT`
Назначение: извлечение квадратного корня
Синтаксис: `<переменная> := SQRT(<выражение>);`
Операнд: выражение типа `FLOAT`.
Возвращаемое значение имеет тип `FLOAT`.

Операция *отрицание*
Имя: `<переменная> := -<выражение>;`
Операнд: знаковое выражение типа `INTEGER, SHORTINT, LONGINT, FLOAT`.
Возвращаемое значение имеет тот же тип, как исходное.

Операция *умножение*
Имя: *
Назначение знаковое умножение
Синтаксис: `<переменная> := <выражение1> * <выражение2>;`
Операнды: выражения типа `INTEGER, SHORTINT, LONGINT, WORD, DWORD, FLOAT`. Оба операнда должны иметь одинаковый тип, число операндов может быть более двух.
Возвращаемое значение имеет тот же тип, как и операнды.

Операция *деление*
Имя: /
Назначение: знаковое деление
Синтаксис: `<переменная> := <выражение1> / <выражение2>;`
Операнды: выражения типа `INTEGER, SHORTINT, LONGINT, WORD, DWORD, FLOAT.` Оба операнда должны иметь одинаковый тип.
Возвращаемое значение имеет тот же тип, как и операнды.

Операция *модуль*
Имя MOD
Назначение модуль выражения
Синтаксис <переменная>:= MOD<выражение>;
Операнд: знаковое выражение типа INTEGER, SHORTINT, LONGINT, FLOAT.
Возвращаемое значение имеет тот же тип как исходное.

Операция *сложение*
Имя: +
Назначение: знаковое сложение
Синтаксис: <переменная> := <выражение1> + <выражение2>;
Операнды: выражения типа INTEGER, SHORTINT, LONGINT, WORD, DWORD, FLOAT
Оба операнда должны иметь одинаковый тип.
Возвращаемое значение имеет тот же тип, как и операнды.

Операция *вычитание*
Имя: -
Назначение: знаковое вычитание
Синтаксис: <переменная> := <выражение1> + <выражение2>;
Операнды: выражения типа INTEGER, SHORTINT, LONGINT, WORD, DWORD, FLOAT. Оба операнда должны иметь одинаковый тип.
Возвращаемое значение имеет тот же тип, как и операнды.

Пример
```
ares := (ax1 + ax2) * (12/ (scale-3));
a := 11/2;
```

22.6 Операции сравнения

< меньше чем
> больше чем
<= меньше или равно
>= больше или равно
= равно
<> не равно

Операция *меньше чем*
Имя: <
Назначение: "меньше чем" логическое сравнение
Синтаксис: <переменная_boolean>:=<выражение1> < <выражение2>;
Операнды: выражения типа INTEGER, SHORTINT, LONGINT, WORD, DWORD, FLOAT, TIMER. Оба выражения должны иметь одинаковый тип.
Возвращаемое значение TRUE если <выражение1> меньше чем <выражение2>.

Операция *больше чем*
Имя >
Назначение "больше чем" логическое сравнение
Синтаксис <переменная_boolean> :=<выражение1> > <выражение2>;

Операнды: выражения типа INTEGER, SHORTINT, LONGINT, WORD, DWORD, FLOAT, TIMER. Оба выражения должны иметь одинаковый тип.
Возвращаемое значение TRUE если <выражение1> больше чем <выражение2>.

Операция *меньше или равно*
Имя <=
Назначение "меньше или равно" логическое сравнение
Синтаксис <переменная_boolean>:=<выражение1> <= <выражение2>;
Операнды: выражения типа INTEGER, SHORTINT, LONGINT, WORD, DWORD, FLOAT, TIMER. Оба выражения должны иметь одинаковый тип.
Возвращаемое значение TRUE если <выражение1> меньше или равно <выражение2>.

Операция *больше или равно*
Имя >=
Назначение "больше или равно" логическое сравнение
Синтаксис <переменная_boolean>:=<выражение1> >= <выражение2>;
Операнды: выражения типа INTEGER, SHORTINT, LONGINT, WORD, DWORD, FLOAT, TIMER. Оба выражения должны иметь одинаковый тип.
Возвращаемое значение TRUE если <выражение1> больше или равно <выражение2>

Операция *равно*
Имя =
Назначение "равно" логическое сравнение
Синтаксис <переменная_boolean> :=<выражение1> = <выражение2>;
Операнды: выражения типа INTEGER, SHORTINT, LONGINT, WORD, DWORD, FLOAT, TIMER. Оба выражения должны иметь одинаковый тип
Возвращаемое значение TRUE если <выражение1> равно <выражение2>.

Операция *не равно*
Имя <>
Назначение " не равно" логическое сравнение
Синтаксис <переменная_boolean>:=<выражение1> <> <выражение2>;
Операнды: выражения типа INTEGER, SHORTINT, LONGINT, WORD, DWORD, FLOAT, TIMER. Оба выражения должны иметь одинаковый тип.
Возвращаемое значение TRUE если <выражение1> не равно <выражение2>.

Пример
```
bt : = (123 > 12);      * bt : true*
bt : = (1.0 >= 1.0);    * bt : true*
bt : = (1s < 0s2);      * bt : false*
bt : = (12 <> 12);      * ошибочное выражение*
```

22.7 Логические операции

NOT логическая инверсия
AND логическое И
OR логическое ИЛИ

XOR логическое исключающее ИЛИ

Операция NOT
Имя NOT
Назначение логическая инверсия выражения
Синтаксис <переменная> := NOT (<выражение>);
Операнды: выражение типа boolean
Возвращаемое значение TRUE если выражение FALSE
FALSE если выражение TRUE

Операция AND
Имя AND или &
Назначение "логическое И"
Синтаксис
<переменная> := <выражение1> AND <выражение2>;
<переменная> := <выражение1> & <выражение2>;
Операнды: выражения типа BOOLEAN.
AND оператор может иметь более двух операторов.
Возвращаемое значение "Логическое И" над операндами.

Пример
```
alarm1 := (curr_level > max_level);
run_mode := auto_mode AND not (alarm);
IF run_mode & cmd101 THEN
     bo12 := bi101 & bx10;
END_IF;
```

Операция XOR
Имя XOR
Назначение "исключающее ИЛИ"
Синтаксис: <переменная> := <выражение1> XOR <выражение2>;
Операнды: выражения типа BOOLEAN
XOR оператор может иметь более двух операндов.
Возвращаемое значение "исключающее ИЛИ" над операндами.
Пример
Следующие две строки имеют одинаковый результат
```
result := (bx10 & NOT(bx23)) OR (NOT(bx10) & bx23);
result := bx10 XOR bx23;
```

Операция OR
Имя: OR
Назначение: "логическое ИЛИ"
Синтаксис: <переменная> := <выражение1> OR <выражение2>;
Операнды: выражения типа BOOLEAN
OR оператор может иметь более двух операндов.
Возвращаемое значение " логическое ИЛИ" над операндами.
Пример
```
special_mode :=auto_mode OR alarm1;
IF special_mode OR cmd101 THEN
     Bo12 :=bi101 OR bx10;
```

```
END_IF;
```

22.8 Управляющие операторы

ASSIGN	присвоить
RETURN	выход
IF/THEN/ELSE	если/тогда /иначе
WHILE	пока
REPEAT	повторить
FOR	до того как
CASE	поливариантное ветвление
EXIT	выход из цикла

Оператор ASSIGN

Имя: :=

Назначение: присвоить переменной значение выражения

Синтаксис: <переменная>:= <выражение>;

Операнды: переменная должна быть внутренней или выходной. Переменная и выражение должны иметь одинаковый тип.

Пример:

```
(*переменная  -переменная *)
bo23 := bo10;
(*переменная -выражение *)
bo23 := NOT (bo10);
bo56 :=b34 OR alarm100 & (level >= over_value);
```

Оператор RETURN

Имя: RETURN

Назначение: заканчивает выполнение текущей программы

Синтаксис: RETURN

Пример:

```
* Программный счетчик*
     IF not (Enable) THEN
          Q := false;
          CV :=0;
          RETURN; (* окончание программы*)
     END_IF;
     IF R THEN
          CV :=0;
     ELSE
          IF CU and (CV < PV) THEN
               CV := CV + 1;
     END_IF;
     Q := (CV >= PV);
```

Оператор IF... THEN ... ELSE ... END_IF

Имя: IF...THEN...ELSE...END_IF

Назначение: выполняет один или два списка ST операторов. Выбор осуществляется согласно значению <двоичное_выражение>

Синтаксис:

```
IF <двоичное_выражение> THEN
          <оператор>;
          <оператор>;
...
ELSE
          <оператор>;
          <оператор>;
...
END_IF;
```

Пример:

```
     IF manual AND NOT (alarm) THEN
          Level := manual_level;
          Bx126 := bi12 OR bi45;
     ELSE
          Level := (lv100*100) / scale;
     END_IF;
```

Примечание. Оператор ELSE является дополнительным. Если оператор ELSE отсутствует, тогда отсутствуют инструкции, соответствующие условию FALSE.

Оператор WHILE

Имя: WHILE...DO...END_WHILE

Назначение: выполняет список ST операторов. Условие продолжения проверяется до начала выполнения. Выполнение списка операторов продолжается пока <двоичное_выражение> = TRUE.

Синтаксис:

```
WHILE <двоичное_выражение> DO
          <список операторов>;
               ...
END_WHILE;
```

Пример:

```
J := 1;
WHILE J <= 100 & words[j] <> 'KEY' DO
  J := J +1;
END_WHILE ;
```

Примечание. Не рекомендуется использовать в операторе WHILE входные переменные, т.к. программа считывает их один раз в цикле и не обновляет их в течении действия оператора WHILE. Это может привести к зависанию программы.

Оператор REPEAT

Имя: REPEAT …UNTIL...END_REPEAT

Назначение: выполняет список ST операторов. Условие продолжения проверяется после выполнения

Синтаксис:

```
REPEAT
     <оператор>;
     <оператор>;
     UNTIL<двоичное_выражение>;
END_REPEAT;
```

Примечание. Не рекомендуется использовать в операторе REAPET входные переменные, т.к. программа считывает их один раз в цикле и не обновляет их в течении действия оператора REPEAT. Это может привести к зависанию программы.

Оператор EXIT
Имя: EXIT
Назначение: безусловный выход из цикла
Синтаксис: EXIT
Операнды: (нет)

Оператор FOR
Имя: FOR...TO...BY...DO...END_FOR
Назначение: выполняет ограниченное число итераций, используя как индекс переменную перчисляемого типа.
Синтаксис:

```
FOR <index> := <mini> TO <maxi> BY <step> DO
          <оператор>;
          <оператор>;
END_FOR;
```

Операнды:

Index	переменная типа INTEGER, увеличивающаяся в каждой итерации
Mini	начальное значение индекса (перед первой итерацией)
maxi	максимально допустимое значение индекса
step	величина прирастания индекса в каждой итерации

Пример:

```
J := 101 ;
FOR I := 1 TO 100 BY 2 DO
  IF WORDS[I] = 'KEY' THEN
    J := I ;
    EXIT;
  END_IF;
END_FOR;
```

Примечание. Оператор [BY step] является дополнительным. Если он опущен, то приращение индекса равно 1. Не рекомендуется использовать в операторе FOR входные переменные, т.к . программа считывает их один раз в цикле и не обновляет их в течении действия оператора FOR. Это может привести к зависанию программы.

Оператор CASE…OF
Имя: CASE…OF
Назначение: поливариантное ветвление по многим направлениям
Синтаксис:

```
CASE <порядковая_переменная> OF
     Значение_1:
<Список операторов>;
Значение_2:
          <Список операторов>;
     Значение_M:
          <Список операторов>;
```

```
     ELSE
          <Список операторов>;
END_CASE;
```

Операнды:

<порядковая_переменная> переменная типа INTEGER, SHORTINT, LONGINT, WORD, DWORD, FLOAT, TIMER.

Пример

```
TW := BCD_TO_INT(THUMBWHEEL);
TW_ERROR := 0;
CASE TW OF
  1,5:  DISPLAY := OVEN_TEMP;
  2:    DISPLAY := MOTOR_SPEED;
  3:    DISPLAY := GROSS - TARE;
  4,6..10: DISPLAY := STATUS(TW - 4);
ELSE  DISPLAY := 0 ;
      TW_ERROR := 1;
END_CASE;
%QW100 := INT_TO_BCD(DISPLAY);
```

22.9 Операции управления таймером

TSTART	пуск таймера
TSTOP	останов таймера
TCONTINUE	продолжить счет таймера

Оператор TSTART

Имя: TSTART

Назначение: Очищает таймерную переменную (записывая в нее значение 0) и запускает счет времени

Синтаксис: TSTART (<таймерная переменная>);

Операнды: любя неактивная таймерная переменная

Оператор TSTOP

Имя: TSTOP

Назначение: Останавливает таймерную переменную. Значение переменной после останова не модифицируется.

Синтаксис: TSTOP (<таймерная переменная>);

Операнды: любя активная таймерная переменная

Оператор TCONTINUE

Имя: TCONTINUE

Назначение: Продолжает инкрементирование указанной таймерной переменной, счет которой был остановлен оператором TSTOP .

Синтаксис: TCONTINUE (<таймерная переменная>);

Операнды: любя неактивная таймерная переменная

22.10 Операции управления программами

GSTART	включает программу в цикл контроллера
GSTOP	исключает программу из цикла контроллера

Оператор GSTART

Имя GSTART
Назначение: Включает указанную программу в цикл контроллера. Начиная с этого момента, программа исполняется в каждом цикле контроллера.
Синтаксис: GSTART (<имя_программы>);
Операнды: любая неактивная программа

Оператор GSTOP
Имя: GSTOP
Назначение: Исключает указанную программу из цикла контроллера. Начиная с этого момента, программа не исполняется в цикле контроллера.
Синтаксис: GSTOP (<имя_программы>);