FMS-3000

УСТРОЙСТВО
ЧПУ

Макропрограммирование

Редакция CE

ООО “Модмаш-Софт”
г. Нижний Новгород

Содержание

Введение

Язык макропрограммирования (ЯМ) системы управления FMS-3000 представляет собой расширение языка программирования управляющих программ и является подмножеством языка BASIC. ЯМ предназначен для выполнения вычислительных операций, операций ввода информации и вывода на экран графической, текстовой и числовой информации. Также ЯМ позволяет осуществить доступ к системным переменным и ячейкам программы электроавтоматики. С помощью ЯМ имеется возможность разрабатывать диалоговые управляющие программы, программы измерения (при наличии датчика касания) и т.п.

Программное обеспечение системы управления FMS-3000 позволяет выполнять программы ЯМ параллельно с отработкой управляющей программы, при условии отсутствия в тексте функций управления станком (G-, M-функций). Данная особенность ЯМ дает возможность организовать дополнительные информационные окна, систему слежения за дополнительными параметрами, режимы контроля и протоколирования процессов обработки и т.д. В отличие от управляющей программы, такие программы выполняются в фоновом режиме (в свободное от всех других задач время) и при большой загрузке могут временно приостанавливать свою работу.

Синтаксис ЯМ

В ЯМ приняты следующие соглашения:

- все переменные, операторы, функции и процедуры ЯМ записываются только строчными буквами латинского алфавита, цифрами и символом подчеркивания « _ »;
- длина имени переменной не должна превышать 255 символов;
- имя переменной не может начинаться с цифры;
- переменные, используемые в программе являются локальными и действуют только в пределах текущего файла;
- глобальные переменные текущей программы (действующие в пределах программы и подпрограмм) должны начинаться с символа подчеркивания « _ »;
- переменные-массивы являются глобальными;
- в одной строке допускается размещать только один оператор;
- выполняемая программа может иметь только одно окно вывода;
- выполняемая программа может иметь только одно диалоговое окно;
- вывод информации в неактивное окно (находящееся на заднем плане) не производится;
- номер метки программируется с буквой “N”;
- адрес перехода в операторах goto, gosub программируется только числом, без символа “N”.

Зарезервированные слова

Зарезервированные слова являются составной частью ЯМ и их нельзя использовать в качестве имени переменной.

Список зарезервированных слов:

abs	and	arc	arccos
arcsin	arctan	bar	bcolor
circle	close	closedialog	closewindow
cls			
color	compare	cos	dump
ellipse	else	end	endiff
eof	endcont	exp	for
frac	getdata	getdatacadr	getdate
geteadata	getparameter	getstring	getsystemdata
gettime	gosub	goto	goto
if	iff	input	insertcontrol
int			
length	lg	line	ln
locate	msg	next	not
open	or	paramactive	pos
print	pset	read	rect
rem	restore	return	round
seteadata	setsystemdata		
sin	sqrt	str	tan
then			
to	trunc	val	window
write	xor		

Операции макроязыка

Операции языка:

« **+** » - арифметическое сложение;
« **-** » - арифметическое вычитание или унарный минус;
« ***** » - арифметическое умножение;
« **/** » - арифметическое деление;
« **** » - остаток от деления (деление по модулю);
« **=** » - присвоение или сравнение на равенство;
« **( )** » - скобки, изменение приоритета операций;
« **<** » - сравнение на “меньше”;
« **>** » - сравнение на “больше”;
« **^** » - возведение в степень;
« **and** » - логическое умножение;
« **or** » - логическое сложение;
« **not** » - логическое отрицание (логическая инверсия);
« **xor** » - логическое “исключающее или”.

Арифметические и логические операции выполняются в порядке их приоритетов. Приоритеты операций (в порядке возрастания) следующие:

- «**+**», «**-**», «**or**», «**xor**» (низший приоритет);
- «*****»,«**/**»,«****»,«**and**»;
- «**^**»;
- унарный минус(плюс), «**not**»;
- «**(**»,«**)**» (высший приоритет).

Операторы

Исходный код содержит операторы, которые описывают выполняемые программой действия.
Приведем примеры операторов:

```
a = b + c                 ( присвоить значение )
window(20,20,200,300)     ( активизировать процедуру )
if x < 2 then goto 100    ( оператор условия )
for i=1 to 100            ( оператор цикла )
x=i*2                     ( присвоить значение )
next                      ( оператор ограничения цикла )
```

В операторах можно присваивать значение, активизировать процедуру или функцию или передавать управление на другую часть кода.

Выражения

Оператор ЯМ состоит из выражений. Выражения оператора могут состоять из операндов и операций. Обычно в выражениях выполняется сравнение либо арифметические или логические операции. Выражения ЯМ могут состоять из более простых выражений. Они могут быть достаточно сложными. Приведем некоторые примеры выражений:

```
x + y
done < > error
i < = length
 - x
a [ i + 20 ]
```

Лексемы

Лексемы - это наименьшие значащие элементы в программе ЯМ. Они образуются операндами и операциями выражений. Лексемы - это специальные символы, зарезервированные слова, идентификаторы, метки и строковые константы.

Приведем примеры лексем ЯМ:

```
print                (зарезервированное слово)
(                    (специальный символ)
=                    (специальный символ)
9                    (число)
“Это текст”          (строковая константа)
```

Типы, переменные, константы

Переменная может содержать изменяемое значение. Каждая переменная может быть только одним из двух типов: вещественный или строковый. Признаком строковой переменной является наличие символа $ в конце имени. Тип переменной определяет множество значений, которые может иметь переменная.

Например, в следующей программе используются переменные x и y, имеющие вещественный тип. Таким образом, x и y могут содержать только числа. Если в программе предпринимается попытка присвоить этим переменным значения строкового типа, то в процессе выполнения программы возникнет ошибка, и программа остановится.

```
x = 7.5        (переменной x присваивается значение )
y = 44         (переменной y присваивается значение)
x = x + y      (значение переменной x изменяется)
```

В этой программе переменной x первоначально присваивается значение 7.5; двумя операторами ниже ей присваивается новое значение: x + y. Как можно видеть, значение переменной может изменяться.

В качестве переменных могут использоваться также и массивы, или упорядоченные множества переменных. Доступ к элементам массивов выполняется через их индексы. Перед использованием массива он должен быть описан с помощью оператора dim

```
dim a[5]
```

Здесь описан массив с именем **a** и с количеством элементов 5.
Индексы массива, указываемые в квадратных скобках, начинаются с нуля. Таким образом, например, запись значения 10 в третий элемент массива **a** будет выглядеть следующим образом:

```
a[2]=10
```

Массивы могут быть как одномерными (см. выше), так и многомерными. Максимальная размерность массива – 6 индексов. При этом количество элементов в массиве не должно превышать 1000000.

Пример использования двумерного массива:

```
dim mm[3,5]
a=2
b=4
mm[a,b]=152
```

Метки

Меткой (номером кадра) является символ “N” и последовательность цифр. Начальные нули являются значащими, то есть если в операторе goto указана метка 00123, то переход будет происходить именно на метку N00123, а не на метку N123. Метки используются с операторами перехода goto и вызова подпрограммы gosub.

Справочник по функциям и процедурам макроязыка

В данном разделе описываются все функции языка.

Алгебраические и тригонометрические функции

Функция **abs**

Функция: Возвращает абсолютное значение аргумента.
Описание: *abs(x)*

Пример:

```
r = abs(-2.3)
i = abs(-157)
window(100,100,200,200)
print r
print i
end
```

Окно вывода

```
2.3
157
```

Функция **arctan**

Функция: Возвращает арктангенс аргумента.
Описание: *arctan(x)*
Примечания: Результат представляет собой главное значение арктангенса *x* (при технологическом параметре N3033, равном 0 - в радианах, при N3033=1 – в градусах).

Пример:

```
 x=1
 r = 180 / 3.14159 * arctan(x)
window(100,100,200,200)
print "Арктангенс 1 =", r , "град"
end
```

Окно вывода

```
Арктангенс 1 = 45 град
```

Функция **tan**

Функция: Возвращает тангенс аргумента.
Описание: *tan(x)*
Примечания: Результат представляет собой значение тангенса *x*, заданного в радианах при N3033=0, в градусах – при N3033=1.

Пример:

```
x = 45 / 180 * 3.14159
r = tan(x)
window(100,100,200,200)
print “Тангенс 45 =”, r
end
```

Окно вывода

Тангенс 45 = 1

Функция sin

Функция: Возвращает синус аргумента.

Описание: *sin(x)*

Примечания: Результат представляет собой значение синуса *x*, заданного в радианах при N3033=0, в градусах – при N3033=1.

Пример:

```
x = 30 *3.14159 / 180
r = sin ( x )
window (100,100,200,200)
print “Синус 30 = “, r
end
```

Окно вывода

Синус 30 = 0.5

Функция cos

Функция: Возвращает косинус аргумента.

Описание: *cos(x)*

Примечания: Результат представляет собой значение косинуса *x*, заданного в радианах при N3033=0, в градусах – при N3033=1.

Пример:

```
x = 60 *3.14159 / 180
r = cos ( x )
window (100,100,200,200)
print “Косинус 60 = “, r
end
```

Окно вывода

Косинус 60 = 0.5

Функция arcsin

Функция: Возвращает арксинус аргумента.

Описание: *arcsin(x)*

Примечания: Результат представляет собой значение арксинуса *x* в радианах при N3033=0, в градусах – при N3033=1.

Пример:

```
x = - 0.5
r = 180 / 3.14159 * arcsin ( x )
window (100,100,200,200)
print “Арксинус –0.5 = “, r ,”град”
end
```

Окно вывода

Арксинус –0.5= - 30 град

Функция **arccos**

Функция: Возвращает арккосинус аргумента.
Описание: *arccos(x)*
Примечания: Результат представляет собой значение арккосинуса *x* в радианах при N3033=0, в градусах – при N3033=1.
.

Пример:

```
x =  0.5
r = 180 / 3.14159 * arccos ( x )
window (100,100,200,200)
print “Арккосинус 0.5 = “, r ,”град”
end
```

Окно вывода

Арккосинус 0.5= 60 град

Функция **exp**

Функция: Возвращает экспоненциальное значение аргумента.
Описание: *exp (x)*
Примечания: Параметр *x* является выражением вещественного типа. Результатом будет экспонента *x*, то есть значение *e* возводится в степень *x* (*e* - основание натурального логарифма).

Пример:

```
x = 4
a = exp ( x )
window (100,100,200,200)
print “e^4 = “, a
end
```

Окно вывода

e^4 = 54.598

Функция **lg**

Функция: Возвращает десятичный логарифм аргумента.
Описание: *lg (x)*
Примечания: Параметр *x* является выражением вещественного типа. Результатом будет десятичный логарифм выражения *x*.

Пример:

```
x = 100
a = lg ( x )
window (100,100,200,200)
print “lg(100) = “, a
end
```

Окно вывода

lg(100) = 2

Функция **ln**

Функция: Возвращает натуральный логарифм аргумента.

Описание: *ln (x)*

Примечания: Параметр *x* является выражением вещественного типа. Результатом будет натуральный логарифм выражения *x*.

Пример:

```
x = 2.718
a = ln ( x )
window (100,100,200,200)
print “ln(2.718) = “, a
end
```

Окно вывода

ln(2.718) = 1

Функция **frac**

Функция: Возвращает дробную часть аргумента.

Описание: *frac (x)*

Примечания: Параметр *x* является выражением вещественного типа. Результатом является дробная часть *x*, то есть
frac (x) = x – int (x).

Пример:

```
f = frac(123.456)
window (100,100,200,200)
print f
end
```

Окно вывода

0.456

Функция **round**

Функция: Округляет значение вещественного типа до значения целого типа.

Описание: *round (x)*

Примечания: Параметр *x* представляет собой выражение вещественного типа. Функция *round* возвращает значение, которое является значением *x*, округленным до ближайшего целого числа. Если значение *x* находится точно посередине между двумя целыми числами, то результатом будет число с большим абсолютным значением.

Пример:

```
r = round(123.456)
window (100,100,200,200)
print r
end
```

Окно вывода

```
123
```

Функция **int**

Функция: Возвращает целую часть аргумента.

Описание: *int(x)*

Примечания: Параметр *x* представляет собой выражение вещественного типа. Результатом будет целая часть *x*, то есть дробная часть x отбрасывается (округляется в сторону нуля).

Пример:

```
i = int (123.756)
window (100,100,200,200)
print i
end
```

Окно вывода

```
123
```

Функция **sqrt**

Функция: Возвращает квадратный корень аргумента.

Описание: *sqrt (x)*

Примечания: Параметр *x* представляет собой выражение вещественного типа. Результатом является квадратный корень *x*.

Пример:

```
i = 9
window (100,100,200,200)
print sqrt(9)
end
```

Окно вывода

```
3
```

Процедура **end**

Функция: Заканчивает выполнение программы.
Описание: *end*
Примечания:
1. Процедура end может быть расположена в любом месте программы. Появление этой процедуры останавливает выполнение программы даже в том случае, когда ниже есть какой-либо код или процедура появилась в подпрограмме.
2. Программы, которые отрабатываются в автоматическом режиме, должны останавливаться по командам M2, M30.

Оператор повторения **for next**

Функция: Оператор повторения *for* позволяет циклически выполнять группу операторов, расположенных ниже до ближайшего оператора *next*.
Описание: *for <выражение начала цикла> to <выражение конца цикла>.*
Примечания: *<Выражение начала цикла>* представляет собой оператор присвоения управляющей переменной начального значения.
<Выражение конца цикла> представляет собой либо число, либо выражение, результатом которого является число. В процессе выполнения управляющая переменная увеличивается на 1 в каждом цикле, до тех пор, пока ее значение не превысит значение *<Выражения конца цикла>.* Операторы цикла могут быть вложенными. Когда начинает выполняться оператор *for*, начальное и конечное значения определяются один раз, и эти значения сохраняются на протяжении всего выполнения оператора *for*.

Пример:

```
for d=10 to 124
a=d * 2
next
```

Пример:

```
x =12
y = 35
for d = x * 20 to y * 40
for j = 1 to 10
a = j * d+123
next
next
```

Условный оператор **iff then else endiff**

Функция: В условном операторе, в зависимости от заданных условий выполняется или не выполняется некоторая группа операторов.

Описание: *iff <выражение> then*
<группа операторов>
else
<группа операторов>
endiff

Ограничения: Группа операторов не может содержать более 50 операторов. Нельзя пропускать операторы else и endiff.

Пример:

```
iff a>b then
c=1
b=2
else
c=10
b=20
endiff
```

Условный оператор **if then**

Функция: В условном операторе, в зависимости от заданных условий выполняется или не выполняется некоторый оператор.

Описание: *if <выражение> then <оператор>*

Пример:

```
if a>b then c=1
if c=1 then goto 20
```

Оператор **gosub**

Функция: Вызывает подпрограмму.

Описание: *gosub n (gosub имя_файла n)*

Примечания: Параметр *n* представляет собой выражение и определяет номер строки, на которую происходит переход. Номер строки обозначается символом "N" и номером.

Ограничения: Подпрограмма должна располагаться обязательно в текущем файле. Расширенный вариант позволяет вызвать подпрограмму, назодящуюся в произвольном файле *имя_файла. имя_файла* задается с помощью строковой переменной или полного имени файла в кавычках.

Пример:

```
a=123
gosub a
.......
end
N123 print "Это подпрограмма"
return

gosub name$  123
gosub “\hdd\lib\program\4567.prg”  123
```

Оператор **return**

Функция: Производит выход из подпрограммы.

Описание: *return*

Примечания: Выход происходит на последний выполненный оператор вызова под-программы *gosub*.

Ограничения: Подпрограмма должна располагаться обязательно в текущем файле.

Пример:

```
gosub 123
.......
end
N123 print "Это подпрограмма"
return
```

Оператор **goto**

Функция: Вызывает переход на заданную строку.

Описание: *goto n*

Примечания: Параметр *n* представляет собой выражение и определяет номер строки, на которую происходит переход. Номер строки обозначается символом "N" и номером.

Пример:

```
goto 123
N10 print "Текст"
 lab=500
goto lab
N123 goto 10
N500 print "TEXT"
```

Оператор getstring

Функция: Вызывает чтение очередного кадра, находящегося в программе ниже вызова подпрограммы, содержащей оператор *getstring*.

Описание: *a$=getstring*

В строковую переменную a$ пересылается текст следующего кадра.

Примечания:

- оператор может использоваться только в подпрограмме, при этом каждый раз происходит чтение следующего кадра, находящегося в программе, из которой была вызвана данная подпрограмма;
- после использования оператора точка возврата из подпрограммы по функции M99 сдвигается на следующий кадр;
- для восстановления точки возврата в исходное состояние необходимо использовать оператор *restore.*

Пример:

```
N100P50
N101G1X100F200
N102Y100
N103Z100
N104X0Y0Z0

Подпрограмма 50:
a$=getstring  - в переменной a$ текст кадра N101
b$=getstring  - в переменной b$ текст кадра N102
c$=getstring  - в переменной c$ текст кадра N103
M99 – возврат в программу на кадр N104
```

Оператор restore

Функция: Восстанавливает точку возврата из подпрограммы в исходное состояние, то есть на кадр, находящийся после обращения к подпрограмме.
Описание: restore
Примечания:

- оператор может использоваться только в подпрограмме.

Пример:
N100P50
N101G1X100F200
N102Y100
N103Z100
N104X0Y0Z0

Подпрограмма 50:
a$=getstring - в переменной a$ текст кадра N101
b$=getstring - в переменной b$ текст кадра N102
c$=getstring - в переменной c$ текст кадра N103
restore
M99 – возврат в программу на кадр N101

Оператор endcont

Функция: Пустой оператор, не вызывающий никаких действий.
Описание: endcont
Примечания:

- оператор может использоваться для ограничения фрагмента программы, обрабатываемой с помощью оператора getstring. При реальной отработке этого фрагмента данный оператор просто пропускается.

Пример:
N100P50
N101G1X100F200
N102Y100
N103Z100
endcont
N104X0Y0Z0

Подпрограмма 50:
N200a$=getstring - в переменной a$ текст кадров N101-N103
If compare(a$,"endcont")<>0 then goto 200 – проверка конца фрагмента программы
restore – восстановление точки возврата из подпрограммы
M99 – возврат в программу на кадр N101

Функции графического вывода

Соответствие значений номеров цветов

Значение	***Цвет***
0	черный
1	синий
2	зеленый
3	бирюзовый
4	красный
5	малиновый
6	коричневый
7	светло-серый
8	темно-серый
9	светло-голубой
10	светло-зеленый
11	светло-бирюзовый
12	светло-красный
13	светло-малиновый
14	желтый
15	белый

Процедура **arc**

Функция: Вычерчивает дугу окружности от начального угла до конечного угла.

Описание: *arc(X,Y, start_angle, end_angle, radiusX,radiusY,colour)*

Примечания: Рисует дугу эллипса с центром (*x,y*) и радиусами "*radiusX*" и "*radiusY*". Дуга рисуется от начального угла ("*start_angle*") до конечного угла ("*end_angle*").

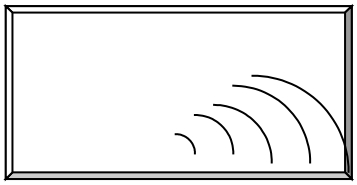
Начальный угол, равный 0 и конечный угол, равный 359, задают вычерчивание полной окружности.

Ограничения: Предварительно должна быть выполнена процедура *window*. Если процедура *window* не будет выполнена, то процедура *arc* игнорируется и программа продолжит свою работу.

Пример:

```
window(20,20,400,400)
for r = 1 to 5
arc(100, 100, 0, 89, r*10, r*10,1)
next
end
```

Окно вывода



В приведенном примере рисуется 5 дуг окружности, с радиусами от 10 до 50 с центром в точке 100,100 синим цветом

Процедура **bar**

Функция: Рисует закрашенный прямоугольник.

Описание: *bar(x1, y1, x2, y2 ,colour)*

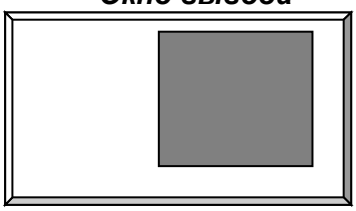
Примечания: Рисуется закрашенный прямоугольник. Координаты верхнего левого угла задаются точкой *x1* и *y1*, координата правого нижнего угла задаются точкой *x2* и *y2*. Цвет прямоугольника задается параметром *colour*.

Ограничения: Предварительно должна быть выполнена функция *window*. Если процедура *window* не будет выполнена, то процедура *bar* игнорируется и программа продолжит свою работу.

Пример:

```
window(20,20,400,400)
X=100
Y=100
bar(x, y,x+200,y+200,3)
end
```

Окно вывода



В приведенном примере рисуется закрашенный квадрат со стороной 200 пикселей бирюзовым цветом.

Процедура **rect**

Функция: Рисует прямоугольник.

Описание: *rect (x1 , y1 , x2 , y2 , colour)*

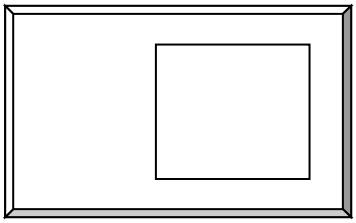
Примечания: Рисуется прямоугольник. Координаты верхнего левого угла задаются точкой *x1* и *y1*, координата правого нижнего угла задаются точкой *x2* и *y2*. Цвет прямоугольника задается параметром *colour*.

Ограничения: Предварительно должна быть выполнена функция *window*. Если процедура *window* не будет выполнена, то процедура *rect* игнорируется и программа продолжит свою работу.

Пример:

```
window(20,20,400,400)
x=100
y=100
rect(x, y,x+200,y+200,5)
end
```

Окно вывода



В приведенном примере рисуется квадрат со стороной 200 пикселей малиновым цветом.

Процедура **line**

Функция: Рисует линию.

Описание: *line (x1 , y1 , x2 , y2 , colour)*

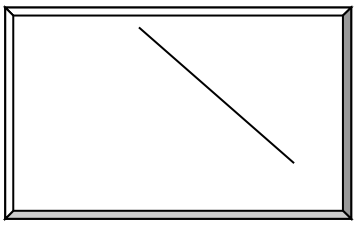
Примечание: Рисуется линия. Координаты верхнего левого угла задаются точкой *x1* и *y1*, координата правого нижнего угла задаются точкой *x2* и *y2*. Цвет линии задается параметром *colour*.

Ограничения: Предварительно должна быть выполнена функция *window*. Если процедура *window* не будет выполнена, то процедура *line* игнорируется и программа продолжит свою работу.

Пример:

```
window(20,20,400,400)
x=100
y=100
line(x, y,x+200,y+200,7)
end
```

Окно вывода



В приведенном примере рисуется линия от точки (100,100) до точки (300,300) светло-серым цветом.

Процедура circle

Функция: Рисует окружность.

Описание: *rect (x , y , radius , colour)*

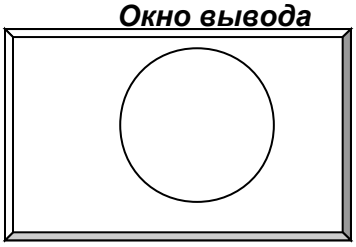
Примечание : Рисуется окружность. Координаты центра задаются точкой *x* и *y*, радиус окружности задается параметром *radius*. Цвет линии задается параметром *colour*.

Ограничения: Предварительно должна быть выполнена функция *window*. Если процедура *window* не будет выполнена, то процедура *circle* игнорируется и программа продолжит свою работу.

Пример:

```
window(20,20,300,300)
x=100
y=100
circle(x, y,75,9)
end
```

Окно вывода



В приведенном примере рисуется окружность с центром в точке (100,100) и радиусом 75 светло-голубым цветом.

Процедура **ellipse**

Функция: Рисует эллипс.

Описание: *ellipse(x1, y1, radiusX, radiusY,colour)*

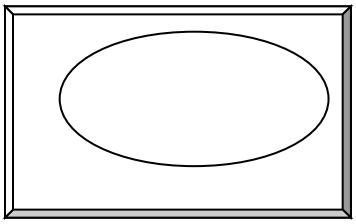
Примечания: Рисуется эллипс. Координаты центра задаются точкой *x* и *y*, оси эллипса задаются параметрами *radiusX* и *radiusY*. Цвет эллипса задается параметром *colour*.

Ограничения: Предварительно должна быть выполнена функция *window*. Если процедура *window* не будет выполнена, то процедура *ellipse* игнорируется и программа продолжит свою работу.

Пример:

```
window(20,20,300,300)
x=100
y=100
ellipse(x, y,100,50,11)
end
```

Окно вывода



В приведенном примере рисуется эллипс с центром в точке (100,100) и осью X 100 пикселей и осью Y 50 пикселей светло-бирюзовым цветом.

Процедура **pset**

Функция: Рисует точку.
Описание: *pset(x, y, colour)*
Ограничения: Предварительно должна быть выполнена функция window. Если процедура *window* не будет выполнена, то процедура *pset* игнорируется и программа продолжит свою работу.

Пример:
```
window(20,20,400,400)
pset(100,100,3)
```

Процедура **cls**

Функция: Очищает окно вывода программы (закрашивает его цветом, установленным процедурой *bcolor*)
Описание: *cls*
Ограничения: Предварительно должна быть выполнена функция *window*. Если процедура *window* не будет выполнена, то процедура *cls* игнорируется и программа продолжит свою работу.

Пример:
```
window(20,20,400,400)
bcolor(0)
cls
```

Процедура **putimage**

Функция: Выводит изображение из файла.
Описание: *putimage(x, y, имя_файла)*
Ограничения: Предварительно должна быть выполнена функция window. Если процедура *window* не будет выполнена, то процедура *putimage* игнорируется и программа продолжит свою работу.

Пример:
```
window(20,20,400,400)
putimage(30,30,"\hdd\lib\image001.bmp")
pic$="\hdd\lib\image002.bmp"
putimage(130,30,pic$)
```

Процедура **setfont**

Функция: Устанавливает шрифт, его размер и параметры.

Описание: *setfont (size,имя_шрифта, opt)*

Примечание: Параметр *size* - размер шрифта, *opt* – может быть равным 0 – обычный шрифт, 1 – курсив, 2 - подчеркнутый.

Ограничения: *имя_шрифта* должно быть из допустимых названий шрифтов в операционной системе.

Процедура **print**

Функция: Выводит на экран значение переменных, функций и выражений.

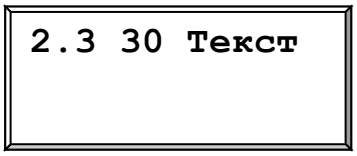
Описание: *print a,b,...*

Ограничения: Предварительно должна быть выполнена функция *window.* Если процедура *window* не будет выполнена, то процедура *print* игнорируется и программа продолжит свою работу.

Пример:

```
a=2.3
b=10
c=3
window(20,20,400,400)
print a,b*c,"Текст"
end
```

Окно вывода



Процедура **locate**

Функция: Устанавливает новое значение точки вывода для оператора *print*.

Описание: *locate (x , y)*

Примечание: Параметры *x* и *y* являются координатами относительно левого верхнего угла окна вывода.

Ограничения: Предварительно должна быть выполнена функция *window.* Если процедура *window* не будет выполнена, то процедура *locate* игнорируется и программа продолжит свою работу.

Процедура **msg**

Функция: Выводит в панель индикации сообщений значение переменных, функций, выражений и текстовых строк.

Описание: *msg a,b,...*

Пример:

msg "Значение температуры=",t

Строковые функции

Функция concat

Функция: Выполняет сцепление двух строковых переменных.
Описание: *concat(b$,"строка2")*
Примечания: Каждый параметр является выражением строкового типа. Результат представляет собой объединение двух строковых параметров. Если длина результирующей строки превышает 255 символов, то выдается сообщение об ошибке и программа останавливается.

Пример:

```
b$="FMS-"
c$="3000"
a$ = concat(b$,c$)
```

Функция **copy**

Функция: Возвращает для строки подстроку.
Описание: *copy (s$, index, count)*
Примечания: Параметр *s$* - выражение строкового типа. Параметры *index* и *count* являются выражениями целого типа. Функция *copy* возвращает строку, число символов которой соответствует параметру *count* и которая начинается с символа строки *s$*, номер которого задан параметром *index*. Если значение параметра *index* превышает длину строки, то возвращается пустая строка. Если параметр *count* задает больше символов, чем остается в строке, начиная с символа *index*, то возвращается только остаток строки.

Пример:

```
b$ = 'ABCDEF';
a$ = copy(b$,2,3)
```

Функция delete

Функция: Удаляет из строки подстроку.
Описание: *delete (s$, index, count)*
Примечания: Параметр *s$* представляет собой выражение строкового типа. Параметры *index* и *count* являются выражениями целого типа. Функция *delete* удаляет символы, количество которых соответствует параметру *count*, начиная с символа строки *s$*, номер которого задан параметром *index*. Если значение параметра *index* превышает длину строки, то символы не удаляются. Если параметр *count* задает больше символов, чем остается в строке, начиная с символа *index*, то удаляется остаток строки.

Пример:

```
b$ = 'ABCDEF';
a$ = delete(b$,2,2)
```

Функция insert

Функция: Вставляет в строку подстроку.
Описание: *insert (s1$, s2$, index)*
Примечания: Параметры *s1$, s2$* представляют собой выражения строкового типа. Параметр *index* является выражением целого типа. Данная функция вставляет строку, задаваемую параметром *s2$*, в строку, задаваемую параметром *s1$*, начиная с позиции, определяемой параметром *index*. Если получившаяся в результате строка превышает 255 символов, то возникает сообщение об ошибке и программа останавливается.

Пример:

```
s$ ="FMS3000"
a$=insert('-',s$,3);
```

Функция pos

Функция: Ищет подстроку в строке.
Описание: *pos (sub$, s$)*
Примечания: Параметры *sub$* и *s$* являются выражениями строкового типа. Данная функция ищет подстроку, заданную параметром *sub$*, в строке *s$* и возвращает целое значение, являющееся позицией первого символа подстроки в строке *s$*. Если подстрока не найдена, то
функция возвращает значение 0.

Пример:

```
s$ = "   123.5"
i = pos("23",s$)
```

Функция length

Функция: Возвращает динамическую длину строки.
Описание: *length (s$)*
Примечания: Параметр *s$* представляет собой выражение строкового типа. Результатом будет длина *s$*.

Пример:

```
s$ = 'abc';
i = length(s$)
```

Функция **val**

Функция: Преобразует строковое значение в его численное представление.

Описание: *val (s$)*

Примечания: Параметр *s$* представляет собой выражение строкового типа. Функция *val* преобразует строку *s$* в ее численное представление. Если где-либо в строке встречается недопустимый символ, то возникает сообщение об ошибке и программа останавливается.

Ограничения: Предшествующие пробелы должны быть удалены.

Пример:

```
s$="23.456"
i=val(s$)
```

Функция **compare**

Функция: Сравнивает два строковых значения.

Описание: *compare (a$, b$)*

Примечания: Параметры *a$* и *b$* представляют собой выражения строкового типа. Функция *compare* сравнивает строки *a*$ и *b$* и возвращает 0, в случае равенства и 1 в случае неравенства.

Пример:

```
if compare (a$, "text" ) then goto 100
```

Функция **copynum**

Функция: Возвращает для строки подстроку, содержащую строковое представление числа.

Описание: *copynum (s$, index)*

Примечания: Параметр *s$* - выражение строкового типа. Параметр *index* является выражением целого типа. Функция *copynum* возвращает строку, которая начинается с символа строки *s$*, номер которого задан параметром *index*, причем из строки s$ копируются только символы “+”, “-“, “.” и цифры. Если значение параметра *index* превышает длину строки, то возвращается пустая строка.

Пример:

```
s$=”X-123.765”
n$=copynum(s$, 2 )
window( 20, 20, 200, 200)
print n$
end
```

Окно вывода

```
-123.765
```

Функция **str**

Функция: Преобразует число в его символьное представление.
Описание: str *(a:n)*
Примечания: Параметр *a* представляет собой выражение числового типа. Параметр *n* определяет количество знаков после запятой в символьном представлении значения выражения, а также знак, до которого округляется это значение перед преобразованием. Отсутствие параметра *n* соответствует экспоненциальной форме представления значения выражения.

Пример:

```
a=23.56
b$=str(a:3)      Значение переменной b$=23.560
c$=str(a:0)      Значение переменной c$=24
d$=str(a)        Значение переменной d$=2.35599999999977E+0001
```

Функции работы с временем и датой

Процедура **getdate**

Функция: Возвращает текущую дату, установленную в операционной системе.
Описание: *getdate (year, month, day)*
Примечания: Возвращаемые значения имеют следующие диапазоны: *year* (год) - 1980..2099, *month* (месяц) - 1..12, *day* (число) 1..31.

Пример:
getdate (y, m, d)

Процедура **gettime**

Функция: Возвращает установленное в операционной системе текущее время.
Описание: *gettime (hour, min, sec)*
Примечания: Возвращаемые параметры принимают следующие значения: *hour* (час) - от 0 до 23, *min* (минута) - от 0 до 59, *sec* (секунда) - от 0 до 59.

Пример:
gettime(h,m,s)

Функции работы с окнами и цветом

Процедура **color**

Функция: Устанавливает цвет, которым выводится текстовая информация.
Описание: *color (c)*
Примечания: Параметр *c* представляет собой выражение целого типа и определяет код цвета.

Пример:
```
window(1,1,600,400)
color(3)
```

Процедура **bcolor**

Функция: Устанавливает цвет фона окна, которым выводится текстовая информация.
Описание: *bcolor (c)*
Примечания: Параметр *c* представляет собой выражение целого типа и определяет код цвета.

Пример:
```
window(1,1,600,400)
bcolor(6)
```

Процедура **window**

Функция: Определяет на экране окно.
Описание: *window (x1, y1, x2, y2)*
Примечания: Параметры *x1, y1* представляют собой координаты верхнего левого угла окна, параметры *x2, y2* - это координаты правого нижнего угла. Правый левый угол экрана соответствует координате (1,1). Обычно максимальный размер окна определяется как (1,1,800,540). Все координаты, которые указываются в процедурах (кроме самих координат окна) являются относительными координатами данного окна. Например, *locate(1,1)* всегда позиционирует курсор на верхний левый угол текущего окна. Все процедуры и функции вывода зависят от текущего окна. Если в программе процедура window уже была выполнена, то следующий вызов приводит к изменению размеров окна и появлении его на переднем плане экрана.

Пример:
```
window(1,1,600,400)
```

Процедура **closewindow**

Функция: Закрывает окно программы.

Описание: *closewindow*

Примечания: Выполнение процедуры не приводит к остановке программы. Если окно не было открыто, то процедура игнорируется.

Пример:

```
window(1,1,600,400)
closewindow
```

Процедура **dialog**

Функция: Определяет на экране дилоговое окно.

Описание: *dialog (x1, y1, x2, y2)*

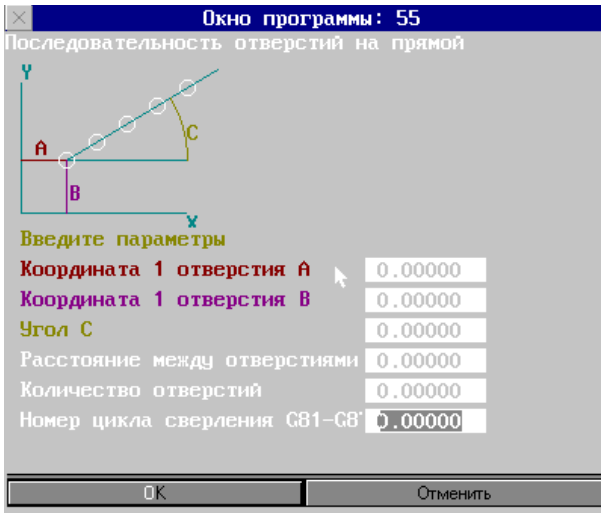
Примечания: Параметры *x1, y1* представляют собой координаты верхнего левого угла диалогового окна, параметры *x2, y2* - это координаты правого нижнего угла. Правый левый угол экрана соответствует координате (1,1). Обычно максимальный размер окна определяется как (1,1,800,540). Все координаты, которые указываются в процедурах (кроме самих координат окна) являются относительными координатами данного окна. Например, *locate(1,1)* всегда позиционирует курсор на верхний левый угол текущего окна. Все процедуры и функции вывода зависят от текущего окна.

Диалоговое окно снабжено двумя кнопками – «OK» и «Отменить». При появлении диалогового окна на зкране создается переменная *inputdialog,* которая доступна в программе, вызвавшей процедуру *dialog*. Начальное значение переменной *inputdialog* равно нулю.

Если диалоговое окно было закрыто кнопкой «Отменить» или клавишей ESC, то значение переменой *inputdialog* становится равным –1.

Если в диалоговом окне была нажата кнопка «OK» или клавиша ENTER, то значение переменой *inputdialog* равно +1.

В диалоговые окна можно вставлять строки ввода данных с помощью процедуры *insertcontrol.* Также для диалоговых окон допустимы все операции вывода графической и текстовой информации.



Пример диалогового окна.

Пример:
dialog(100,0,400,180)
color(15)
locate(10,20)
name$=" "
print ",Введите имя файла "
insertcontrol(100,20,180,36,name$)
N909 if inputdialog=0 then goto 909
closedialog
if inputdialog<>1 then end

Процедура **insertbutton**

Функция: Вставляет в панель макропрограммы кнопку.
Описание: *insertbutton (name,n)*
Примечания: Процедура *insertbutton* позволяет вставить в диалоговое окно кнопку. Строковое выражение name создает надпись на кнопке. Выражение n должно находиться в диапазоне от 1 до 8 и указывает номер функциональной клавиши, с которой связана кнопка. Проверить нажатие кнопки можно с помощью переменной inputdialog. Если значение переемменной равно n, то это означает, что пользователь нажал соответсвующую функциональную клавишу или щелкнул на кнопке мышкой.

Пример.
insertbutton(“OK”,2)
insertbutton(“Отменить”,3)

Процедура **insertcontol**

Функция: Вставляет в диалоговое окно строку ввода данных.
Описание: *insertcontrol (x1, y1, x2, y2 ,var)*
Примечания: Процедура *insertcontrol* позволяет вставить в диалоговое окно строку ввода данных. Координаты *x1,y1,x2,y2* определить положение строки в диалоговом окне.

Переменная *var* может быть числовой или строковой. Переменные другого типа использовать не допускается. Значение переменной изменяется только в том случае, когда нажата клавиша «ОК» в диалоговом окне.

Пример.

```
insertcontrol(20,40,120,60,alfa)
insertcontrol(20,80,120,100,name$)
```

Процедура **closedialog**

Функция: Закрывает диалоговое окно.

Описание: *closedialog*

Примечания: Выполнение процедуры не приводит к остановке программы. Если диалоговое окно не было открыто, то процедура игнорируется.

Пример.

```
dialog(100,0,400,180)
color(15)
locate(10,20)
name$=" "
print ",Введите имя файла "
insertcontrol(100,20,180,36,name$)
N909 if inputdialog=0 then goto 909
closedialog
if inputdialog<>1 then end
```

Функция **getdatacadr**

Функция: Возвращает данные управляющей программы, включая кадр, предшествующий функции.

Описание: *getdatacadr (n)*

Примечания: Параметр *n* представляет собой выражение целого типа и определяет вид возвращаемых данных.

Соответствие между номерами параметров и видом данных:

1..12 - абсолютные координаты осей в текущей системе, заданные в управляющей программе. Порядок осей определяется в соответствии с базовыми станочными параметрами;

13..112 - действующие на текущий момент G-функции:

13 - G0-G3;
17 - G4;
22 - G9;
23 - G10;
27 - G14,G15;
30 - G17-G20;
32 - G21,G22;
34 - G23,G24;
40 - G27;
41 - G28;
43 - G30,G31;
45 - G32;
46 - G33;
48 - G35,G36;
50 - G37;
51 - G38;
53 - G40-G42;
56 - G43,G44,G49;
58 - G45,G46;
66 - G53-G59,G92;
73 - G60;
74 - G50,G61,G62,G63;
77 - G64;
78 - G65,G66;
80 - G67,G68;
82 - G69;
93 - G80-G89;
103 - G90,G91;
107 - G94, G95
109 - G96, G97
113 - номер 1-й оси для плоскости, заданной G20;
114 - номер 2-й оси для плоскости, заданной G20;

115..126 - текущие смещения нулей координат. Порядок осей определяется в соответствии с базовыми станочными параметрами;

127 - номер предыдущего кадра;
128 - номер корректора на радиус D;
129 - значение корректора на радиус;
130 - номер корректора на длину H;

131 - значение корректора на длину;

132 - подача F;

133	-	частота вращения шпинделя S;
134	-	номер инструмента T;
135	-	время задержки E(G4);
136	-	текущее состояние шпинделя (3,4,5);
137-148	-	координаты точки касания в станочной системе координат
149-160	-	коэффициенты масштабирования по осям

Функция **getsystemdata**

Функция: Возвращает данные системных переменных.
Описание: *getsystemdata (n)*
Примечания: Параметр *n* представляет собой выражение целого типа и определяет вид возвращаемых данных.

Соответствие между номером параметра и видом данных:

1..12	-	заданное положение для оси координат с номером "n" – в микронах;
13..24	-	абсолютное положение оси координат с номером "n-12" - в микронах;
25..36	-	заданное приращение(скорость) для оси координат с номером "n-24" - в микронах за такт таймера;
37..48	-	код на ЦАП для оси координат с номером "n-36" без учета дрейфа нуля;
49..60	-	значение положительного программного конечного выключателя для оси координат с номером "n-48" - в микронах;
61..72	-	значение отрицательного программного конечного выключателя для оси координат с номером "n-60" - в микронах;
73..84	-	значение параметров N7006, N7106 и т.д. (радиусное/диаметральное задание и индикация перемещений)
85	-	значение параметра N3030 (действие диаметрального задания на IJK)
86	-	значение параметра N2021 (способ задания IJK)
90	-	номер отрабатываемого кадра
101..112	-	значения добротностей по координатам
121..132	-	значения коэффициентов скоростной компенсации по координатам
133..144	-	заданное положение для осей координат в микронах с учетом перемещений в ручном режиме при установленной нечувствительности к ручным движениям
301..312	-	значение аддитивного смещения нуля для функций G54-G59 для оси координат с номером "n-300" - в миллиметрах;
401..412	-	значение смещения нуля для функции G54 для оси координат с номером "n-400" - в миллиметрах;
501..512	-	значение смещения нуля для функции G55 для оси координат с номером "n-500" - в миллиметрах;
601..612	-	значение смещения нуля для функции G56 для оси координат с номером "n-600" - в миллиметрах;
701..712	-	значение смещения нуля для функции G57 для оси координат с номером "n-700" - в миллиметрах;
801..812	-	значение смещения нуля для функции G58 для оси координат с номером "n-800" - в миллиметрах;
901..912	-	значение смещения нуля для функции G59 для оси координат с номером "n-900" - в миллиметрах;
1001..1255	-	значение корректора с номером "n-1000" – в миллиметрах;
1301..1332	-	значение целочисленного параметра пользователя 2 группы с номером “n-1300”;
1401..1416	-	значение вещественного параметра пользователя 1 группы с номером “n-1400”;
1500	-	значение параметра N3033(способ задания тригонометрических фукнций);

1501 - значение параметра N2005(включена/выключена векторная коррекция);
1502 - значение параметра N1000(количество управляемых осей).

Процедура **setsystemdata**

ВНИМАНИЕ !!!!!!!!!!!

Некорректное использование процедуры setsystemdata может привести к сбоям в работе системы, вызвать поломки станка.

Функция: Устанавливает новые значения системных переменных.

Описание: *setsystemdata (n, value)*

Примечания: Параметр *n* представляет собой выражение целого типа и определяет номер системной переменной. Параметр *value* представляет собой выражение вещественного типа, которое содержит новое значение системной переменной.

Соответствие между номером параметра и видом данных:

1..12 - заданное положение для оси координат с номером "n" – в микронах;
13..24 - абсолютное положение оси координат с номером "n-12"- в микронах;
25..36 - заданное приращение(скорость) для оси координат с номером "n-24" - в микронах за такт таймера;
37..48 - код на ЦАП для оси координат с номером "n-36" без учета дрейфа нуля;
49..60 - значение положительного программного конечного выключателя для оси координат с номером "n-48" – в микронах;
61..72 - значение отрицательного программного конечного выключателя для оси координат с номером "n-60" - в микронах;
101..112 - значения добротностей по координатам;
121..132 - значения коэффициентов скоростной компенсации по координатам;
141..152 - значение коэффициентов масштабирования по координатам;
161..172 - значение абсолютного нуля для оси координат с номером "n-160";
301..312 - значение аддитивного смещения нуля для функций G54-G59 для оси координат с номером "n-300" - в миллиметрах;
401..412 - значение смещения нуля для функции G54 для оси координат с номером "n-400" - в миллиметрах;
501..512 - значение смещения нуля для функции G55 для оси координат с номером "n-500" - в миллиметрах;
601..612 - значение смещения нуля для функции G56 для оси координат с номером "n-600" - в миллиметрах;
701..712 - значение смещения нуля для функции G57 для оси координат с номером "n-700" - в миллиметрах;
801..812 - значение смещения нуля для функции G58 для оси координат с номером "n-800" - в миллиметрах;
901..912 - значение смещения нуля для функции G59 для оси координат с номером "n-900" - в миллиметрах;
1001..1255 - значение корректора с номером "n-1000" – в миллиметрах;

Процедура **geteadata**

Функция: Возвращает данные переменных модуля электроавтоматики.

Описание: *geteadata (group, index, var)*

Примечания: Параметр *group* представляет собой выражение целого типа и определяет группу переменных, параметр *index* определяет адрес байта(слова) в группе. Информация из байта(слова) записывается в переменную *var*. В том случае, когда указаны неверная группа или адрес байта в группе в переменную записывается ноль, программа продолжает свою работу и никакого сообщения не выдается.

Соответствие между номером группы и видом данных:

1 - Входы (байт)
2 - Выходы (байт)
3 - Динамическая память (байт)
4 - Таймеры (текущее значение) (слово)
5 - Счетчики (текущее значение) (слово)
6 - Статическая память (байт)
7 - Обменные ячейки (байт)

Пример.

```
window (20, 20, 200, 200)
rem Чтение входного сигнала I2
geteadata( 1, 2, inp )
rem Проверка наличия 3 бита  во входном сигнале
if ( inp and 4)<>0 then goto 10
print “Бит 3 не установлен”
goto 20
N10 print “Бит 3 установлен”
N20 end
```

Процедура **seteadata**

ВНИМАНИЕ !!!!!!!!!!!
Некорректное использование процедуры seteadata может привести к сбоям в работе системы, вызвать поломки станка.

Функция: Устанавливает новые значения системных переменных.
Описание: *seteadata (group, index, value)*
Примечания: Параметр *group* представляет собой выражение целого типа и определяет группу переменных, параметр *index* определяет адрес байта(слова) в группе. Значение определяемое выражением *value* в указанный байт(слово). В том случае, когда указаны неверная группа или адрес байта в группе процедура *seteadata* игнорируется, программа продолжает свою работу и никакого сообщения не выдается.
Соответствие между номером группы и видом данных:

1 - Входы (байт)
2 - Выходы (байт)
3 - Динамическая память (байт)
4 - Таймеры – не обрабатываются
5 - Счетчики (текущее значение) (слово)
6 - Статическая память (байт)
7 - Обменные ячейки (байт)

Пример.

```
rem Установка динамической памяти M10
mem = 48
seteadata( 3, 10, mem )
end
```

Процедура **gettooldata**

Функция: Возвращает данные из таблицы инструментов для заданного инструмента.

Описание: *gettooldata (num, index, var)*

Примечания: Параметр *num* представляет собой выражение целого типа и определяет номер инструмента, параметр *index* определяет вид запрашиваемых данных. Информация записывается в переменную *var*. В том случае, когда указаны недопустимый номер инструмента, недопустимый вид данных или адрес переменной, выдается соответствующее сообщение.

Соответствие между номером *index* и видом данных:

1	-	длина инструмента
2	-	радиус инструмента
3	-	номер гнезда инструмента
4	-	тип инструмента
5	-	комментарий (строковый тип)
6	-	ориентация
7	-	скругление
40	-	идентификатор
41	-	полное время жизни
42	-	оставшееся время жизни
43	-	аналог
10..19	-	смещения по осям
20..29	-	позиция смены инструмента по осям
30..39	-	аддитивные смещения по осям

Пример.

```
rem Запрос смещения по оси 2 для инструмента 10 с размещением в переменной sm
gettooldata(10,11,sm)
```

Процедура **settooldata**

Функция: устанавливает данные в таблицу инструментов для заданного инструмента.

Описание: *settooldata (num, index, var)*

Примечания: Параметр *num* представляет собой выражение целого типа и определяет номер инструмента, параметр *index* определяет вид устанавливаемых данных. Информация устанавливается из переменной *var*. В том случае, когда указаны недопустимый номер инструмента, недопустимый вид данных или адрес переменной, выдается соответствующее сообщение.

Соответствие между номером *index* и видом данных:

1	-	длина инструмента
2	-	радиус инструмента
3	-	номер гнезда инструмента
4	-	тип инструмента
5	-	комментарий (строковый тип)
6	-	ориентация
7	-	скругление
40	-	идентификатор
41	-	полное время жизни
42	-	оставшееся время жизни
43	-	аналог
10..19	-	смещения по осям
20..29	-	позиция смены инструмента по осям
30..39	-	аддитивные смещения по осям

Пример.
rem Установка длины для инструмента 10 из переменной abc
settooldata(10,1,abc)

Функция **paramactive**

Функция: Проверяет наличие параметра при вызове подпрограммы.
Описание: *paramactive (param)*
Примечания: Параметр *param* представляет собой выражение целого типа в диапазоне от 65 до 90 и определяет, был ли указан параметр при вызове подпрограммы *P*. Числовое значение параметра соответствует коду ASCII символов от “A” до “Z” латинского алфавита. Значение функции равно 0, если параметр не был указан и равно 1, если параметр был указан. Если значение параметра *param* выходит за пределы диапазона 65..90 выдается сообщение об ошибке и программа останавливается. Использовать функцию paramactive можно только при вызове подпрограммы с помощью символа P. При вызове подпрограмм с помощью процедуры gosub данная функция не имеет смысла.

Пример.

```
N10 P200 X12.34 Y45 H44 K73
……
……

:200
rem Проверка параметра X
if paramactive ( 88 ) then goto 10
msg “Не задан параметр X”
end
N10 x=getparameter ( 88 )
rem Проверка параметра Y
if paramactive ( 89 ) then goto 20
msg “Не задан параметр Y”
end
N20 y=getparameter ( 89 )
rem Проверка параметра H
if paramactive ( 72 ) then goto 30
msg “Не задан параметр H”
end
N30 h=getparameter ( 72 )
rem Проверка параметра K
if paramactive ( 75 ) then goto 40
msg “Не задан параметр K”
end
N40 k=getparameter ( 75 )
….
M99
```

Функция **getparameter**

Функция: Возвращает значение соответствующего параметра, заданного в подпрограмме.

Описание: *getparameter (param)*

Примечания: Параметр *param* представляет собой выражение целого типа в диапазоне от 65 до 90 и определяет значение параметра, заданного при вызове подпрограммы *P*. Числовое значение параметра соответствует коду ASCII символов от “A” до “Z” латинского алфавита. Значение функции не определено, если параметр не был задан. Если значение параметра *param* выходит за пределы диапазона 65..90 выдается сообщение об ошибке и программа останавливается. Для проверки того, был ли задан параметр, необходимо использовать функцию paramactive.

Символ	***Код***
A	**65**
B	**66**
C	**67**
D	**68**
E	**69**
F	**70**
G	**71**
H	**72**
I	**73**
J	**74**
K	**75**
L	**76**
M	**77**
N	**78**
O	**79**
P	**80**
Q	**81**
R	**82**
S	**83**
T	**84**
U	**85**
V	**86**
W	**87**
X	**88**
Y	**89**
Z	**90**

Пример.

```
N10 P200 X12.34 Y45 H44 K73
……
……

:200
rem Проверка параметра X
if paramactive ( 88 ) then goto 10
msg “Не задан параметр X”
end
N10 x=getparameter ( 88 )
rem Проверка параметра Y
if paramactive ( 89 ) then goto 20
msg “Не задан параметр Y”
end
N20 y=getparameter ( 89 )
rem Проверка параметра H
if paramactive ( 72 ) then goto 30
msg “Не задан параметр H”
end
N30 h=getparameter ( 72 )
rem Проверка параметра K
if paramactive ( 75 ) then goto 40
msg “Не задан параметр K”
end
N40 k=getparameter ( 75 )
….
M99
```

Процедура **dump**

Функция: Выводит на экран значения всех переменных, используемых в программе.

Описание: *dump*

Примечания: Процедура *dump* открывает на экране окно и выводит в него информацию о значениях всех переменных, используемых в программе. Данная процедура используется для отладки.

Процедура **rem**

Функция: Весь дальнейший текст (до конца строки) считается комментарием.

Описание: *rem*

Примечания: Процедура *rem* позволяет вводить в текст комментарии, поясняющие работу программы. Также в процессе отладки можно временно “закомментировать” некоторые операторы.

Пример

rem Это комментарий

Процедура **input**

Функция: Производит ввод с клавиатуры числовых или строковых данных в переменную.

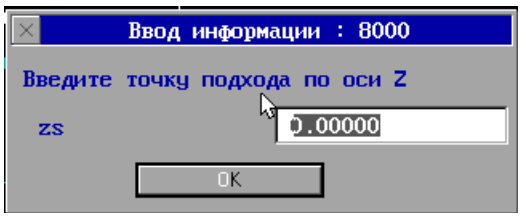
Описание: *input “comment” , inp*

Примечания: Процедура *input* позволяет вводить значения числовых или строковых переменных в процессе работы программы. При вызове процедуры input на экране появляется диалоговое окно ввода информации. Параметр *“comment”* является комментарием, который выводится в окне. Этот параметр допускается опускать. Параметр inp является переменной, в которую заносится информация из диалогового окна.

Пример

input “Введите точку подхода по оси Z”, zs

Диалоговое окно ввода информации



Процедура **open**

Функция: Открывает файл на диске для чтения (записи, добавления).

Описание: *open filename for mode as #number*

Примечания: Процедура *open* открывает файл для ввода, вывода или добавления.

Параметр *filename* является выражением(переменной) строкового типа и представляет собой имя файла. В именах не допускается использовать следующие символы - ; , = + < > | " [ ] \ , .

Параметр *mode* может принимать 4 значения

- input (открыть существующий файл для ввода из него информации);
- output (создать, а если существует, то заменить файл для записи в него информации);
- append (открыть существующий файл для записи в него информации, она записывается в конец файла, то есть добавляется к уже существующей);
- work (открыть файл для отработки – загрузить на отработку), при этом параметр *as #number* не указывается.

Параметр *number* является идентификатором файла для операций read и write и может принимать значения от 1 до 9, то есть можно одновременно открыть не более 9 файлов.

Пример.

```
open “c:\report.txt” for append as #4
open “d:\data.tap” for input as #2
open “c:\program\out.dat” for output as #5
```

Процедура **read**

Функция: Считывает одно или более значений из файла в одну или более переменных. Описание*: read #number a , b , c ,...*

Примечания: Параметр *number* является идентификатором файла (см. процедуру *open*).

Каждый параметр *a, b , c, …* является переменной строкового или числового типа.

В случае переменной числового типа процедура *read* считывает из файла последовательность символов, которые образуют число со знаком. Любые пробелы, знаки табуляции или метки конца строки, предшествующие числовой строке, пропускаются. Считывание прекращается при обнаружении первого пробела, символа табуляции, не цифры или метки конца строки, которые следуют за числовой строкой, или в том случае, если файл заканчивается. Если числовая строка не соответствует ожидаемому формату, то происходит ошибка и программа останавливается. В противном случае переменной присваивается значение. Если файл закончился до появления числа, то переменной присваивается нулевое значение. Следующая операция *read* начнется с пробела, символа табуляции или метки конца строки, которыми завершилась числовая строка.

В случае переменной строкового типа процедура *read* считывает все символы, вплоть до следующей метки конца строки (но не включая ее), или пока файл не кончится, или пока размер результирующей строки не достигнет 255 символов. Переменной присваивается получившаяся в результате символьная строка. Следующая операция *read* начинается с метки конца строки, которой завершилась предыдущая строка.

Ограничения: Процедура *read* работает только с файлами открытыми для ввода информации.

Процедура **write**

Функция: Записывает значение одной или более переменных в файл.

Описание*: write #number a , b , c ,...*

Примечания: Параметр *number* является идентификатором файла (см. процедуру *open*). При использовании параметра *number* равным нулю, значения переменных записываются в окно текстового редактора(если оно открыто).

Ограничения: Процедура *write* работает только с файлами открытыми для вывода или добавления (output или append) информации.

Процедура **close**

Функция: Закрывает файл.

Описание*: close #number*

Примечания: Параметр *number* является идентификатором файла (см. процедуру *open*).

Функция **eof**

Функция: Возвращает состояние файла.

Описание*: eof(number)*

Примечания: Параметр *number* является идентификатором файла (см. процедуру *open*). Значение функции равно 1, если внутренний указатель позиции находится в конце файла и равно 0 в противном случае. Если файл не открыт, то появляется сообщение об ошибке и программа останавливается. Если значение параметра number больше 9 или меньше 1, то появляется сообщение об ошибке и программа останавливается.

Перечень сообщений об ошибках в макропрограммах

Арифметическое переполнение - значение переменной, выражения или параметра больше допустимого;
Деление на ноль;
Длина строки более 255 символов – длина результирующей строковой переменной после выполнения операций *concat* и *insert* превышает 255 символов;
Должен быть оператор "as" – в процедуре *open* не задан оператор *as*;
Должно быть число, переменная или функция – недопустимое сочетание символов в выражении;
Корень из отрицательного значения;
Недопустимое значение - недопустимое значение переменной, выражения или параметра;
Недопустимое значение аргумента функции – значение аргумента функций *tan, lg, ln* находится вне области определения этих функций;
Недопустимое имя файла – имя файла в процедуре open не соответствует требованиям DOS;
Недопустимый номер – недопустимое значение индекса массива, идентификатора файла в файловых процедурах, номера параметра в процедурах и функциях *getdatacadr, getsystemdata, setsystemdata, geteadata, seteadata***;**
Не найден оператор "else" – в конструкции *iff* отсутствует оператор *else*;
Не найдена метка – при выполнении операторов *goto* или *gosub* не найдена точка перехода;
Нет знака = ;
Нет оператора "endiff" – в конструкции *iff* отсутствует оператор *endiff***;**
Нет оператора "for";
Нет оператора "next";
Нет оператора "then";
Нет оператора "to";
Нет открытого опрератора "iff" – задание оператора *else* без *iff*;
Нет "(";
Нет ")";
Не указан идентификатор файла ;
Не указан режим открытия файла (input,output,append);
Ожидается конец строки или кадра – в строке с прцедурами работы с файлами дополнительно заданы другие выражения, процедуры и функции;
Ожидается переменная – использование константы вместо переменной;
Ожидается число – использование переменной вместо константы**;**
Ожидается ",";
Ожидается ';
Оператор "return" без оператора "gosub"';
Отсутствует оператор – пропущены символы = < >**;**
Ошибка доступа к файлу;
Ошибка чтения числа из файла – при чтении в файле встретилось не число**;**
Синтаксическая ошибка – недопустимый символ или сочетание символов**;**
Требуется строковая переменная – пересылка строковых данных в переменную другого типа;
Файл не открыт ;
Файл не открыт для ввода;
Файл с данным идентификатором уже открыт – задание процедуры *open* с уже используемым идентификатором файла #;
Функции *getstring* или *restore* используются не в подпрограмме;