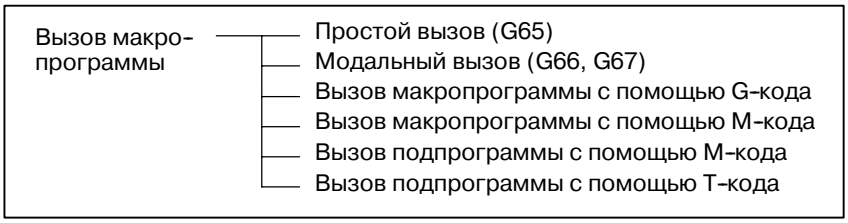
15.6 ВЫЗОВ МАКРО-ПРОГРАММЫ

Можно вызвать макропрограммы с помощью следующих методов:



Ограничения

- **Различия между вызовами макропрограммы и вызовами подпрограммы**

Вызов макропрограммы (G65) отличается от вызова подпрограммы (M98), как описано ниже.

- С помощью G65 можно задать аргумент (данные передаются в макропрограмму). M98 не имеет такой возможности.
- Если в блоке M98 содержится другая команда ЧУ (например, G01 X100.0 M98Pp), то вызов подпрограммы осуществляется после выполнения этой команды. С другой стороны, G65 вызывает макропрограмму без условий.
- Если в блоке M98 содержится другая команда ЧУ (например, G01 X100.0 M98Pp), то станок останавливается в режиме единичного блока. С другой стороны, G65 не приводит к остановке станка.
- При G65 уровень локальных переменных меняется. При M98 уровень локальных переменных не меняется.

15.6.1 Простой вызов (G65)

Если задан G65, то вызывается макропрограмма пользователя, заданная в адресе P. Данные (аргумент) могут передаваться в макропрограмму пользователя.

```
G65 P_ L_ <аргумент-определение> ;

   P_  : Номер вызываемой программы
   L_  : Количество повторов (по умолчанию 1)
   Аргумент :  Данные, переданные в макропрограмму
```

```
O0001 ;                          O9010 ;
   :                             #3=#1+#2;
G65 P9010 L2 A1.0 B2.0 ;         ЕСЛИ [#3 GT 360] GOTO 9 ;
   :                             G00 X#3 ;
M30 ;                            N9 M99 ;
```

Пояснения

- **Вызов**

  - Задайте в адресе P после G65 номер макропрограммы пользователя для вызова.
  - Если требуется ввести количество повторов, после адреса L задайте число от 1 до 9999. Если L пропущено, подразумевается 1.
  - При определении аргумента значения присваиваются соответствующим локальным переменным.

- **Указание аргумента**

  Имеются два типа указания аргумента. В типе I указания аргумента используются буквы, кроме G, L, O, N и P, каждая один раз. В типе II указания аргумента используются буквы A, B и C, каждая один раз, а также используются I, J и K до десяти раз. Тип указания аргумента определяется автоматически согласно используемым буквам.

Указание аргумента

Адрес	Номер переменной
A	#1
B	#2
C	#3
D	#7
E	#8
F	#9
H	#11

Адрес	Номер переменной
I	#4
J	#5
K	#6
M	#13
Q	#17
R	#18
S	#19

Адрес	Номер переменной
T	#20
U	#21
V	#22
W	#23
X	#24
Y	#25
Z	#26

  - Нельзя использовать в аргументах адреса G, L, N, O и P.
  - Можно пропустить адреса, указание которых необязательно. Локальные переменные, соответствующие пропущенным адресам, устанавливаются на нуль.
  - Нет необходимости указывать адреса буквами. Они соответствуют формату адреса слова.
    Однако, I, J и K необходимо задавать буквами.

Пример

B_A_D_ … J_K_ Верно
B_A_D_ … J_I_ Неверно

Указание аргумента II

В типе II указания аргумента используются буквы A, B и C, каждая один раз, а I, J и K используются до десяти раз. Тип II указания аргумента используется для передачи в качестве аргументов таких значений, как трехмерные координаты.

Адрес	Номер переменной
A	#1
B	#2
C	#3
I_1	#4
J_1	#5
K_1	#6
I_2	#7
J_2	#8
K_2	#9
I_3	#10
J_3	#11

Адрес	Номер переменной
K_3	#12
I_4	#13
J_4	#14
K_4	#15
I_5	#16
J_5	#17
K_5	#18
I_6	#19
J_6	#20
K_6	#21
I_7	#22

Адрес	Номер переменной
J_7	#23
K_7	#24
I_8	#25
J_8	#26
K_8	#27
I_9	#28
J_9	#29
K_9	#30
I_{10}	#31
J_{10}	#32
K_{10}	#33

- Нижние индексы I, J и K для обозначения порядка указания аргумента не записываются в фактической программе.

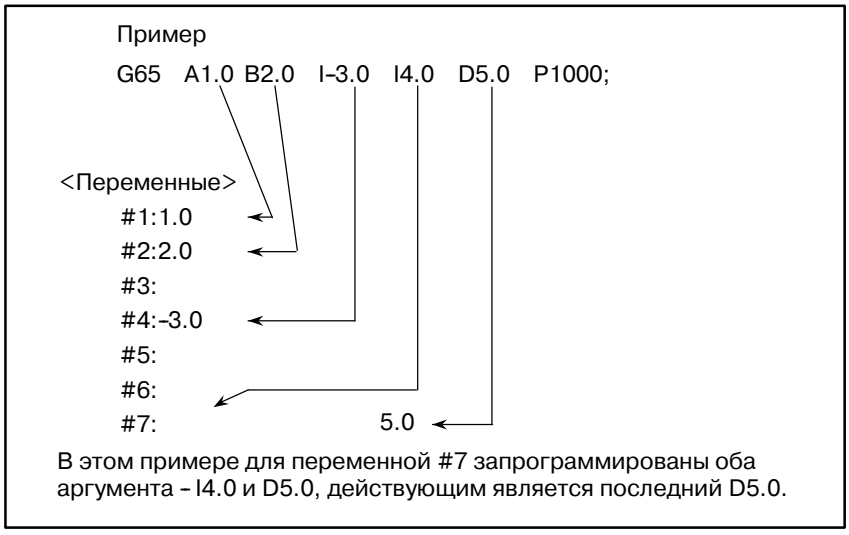
Ограничения

- **Формат**

Перед аргументом необходимо задать G65.

- **Комбинация типа I и II указания аргумента**

ЧПУ внутренне идентифицирует тип I или тип II указания аргумента. Если задана комбинация типа I и типа II указания аргумента, то применяется тип указания аргумента, заданный последним.



В этом примере для переменной #7 запрограммированы оба аргумента – I4.0 и D5.0, действующим является последний D5.0.

- **Положение десятичной точки**

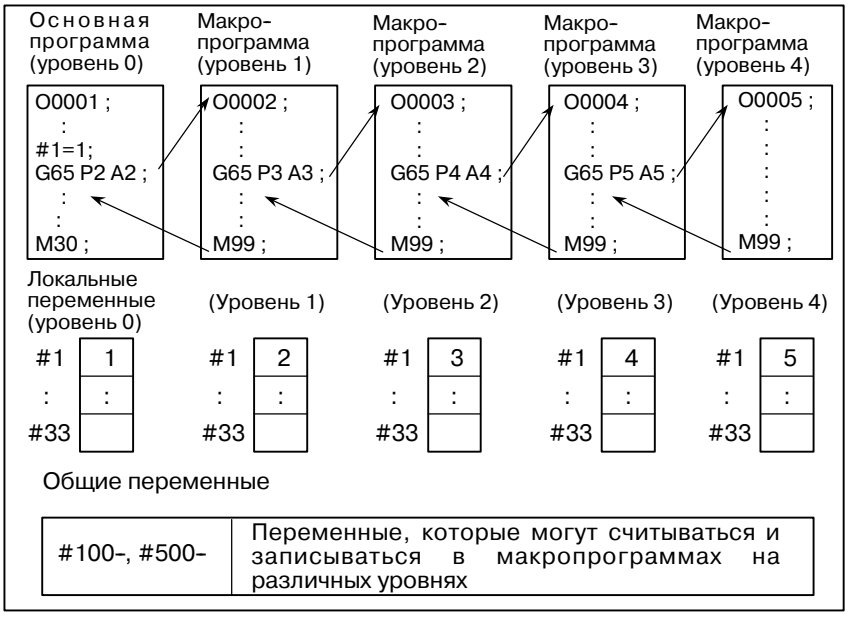
Единицы, используемые для данных аргумента, передаваемых без десятичной точки, соответствуют наименьшему вводимому приращению в каждом адресе. Значение аргумента, передаваемого без десятичной точки, может варьироваться в зависимости от системной конфигурации станка. Рекомендуется использовать десятичные точки в аргументах вызовов макропрограмм в целях поддержания программной совместимости.

- **Вложение вызова**

- **Уровни локальных переменных**

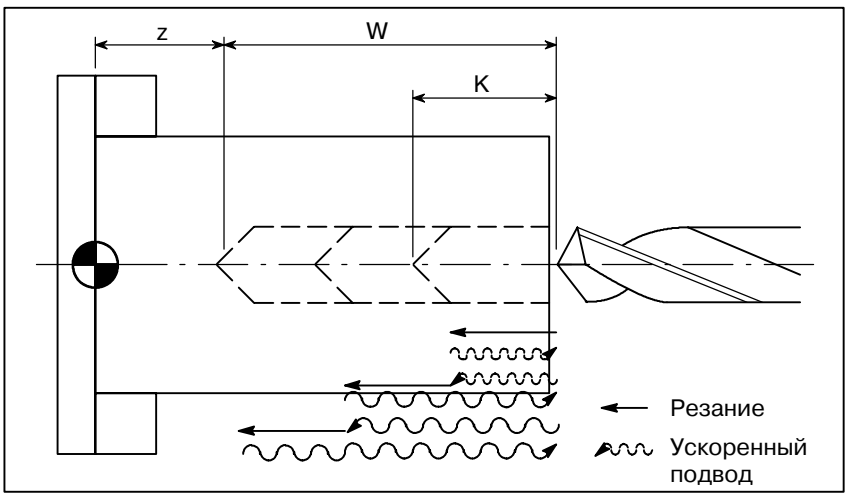
Можно представить вызовы в виде вложений до четырех уровней, включая простые вызовы (G65) и модальные вызовы (G66). Вызовы подпрограмм (M98) не включаются.

- Предусмотрено вложение переменных 0 - 4 уровня.
- Уровень основной программы - 0.
- Каждый раз при вызове макропрограммы (с помощью G65 или G66) уровень локальной переменной увеличивается на единицу. В ЧПУ хранятся значения локальных переменных предыдущих уровней.
- Если M99 выполняется в макропрограмме, то управление возвращается в вызывающую программу. В этот момент уровень локальной переменной уменьшается на единицу; а значения локальных переменных, сохраненные при вызове макропрограммы, восстанавливаются.



Образец программы (Цикл сверления)

Заблаговременное перемещение инструмента по оси X и оси Z в положение, в котором начинается цикл сверления. Задайте Z или W для указания глубины отверстия, K - для глубины резания, и F - для рабочей подачи при сверлении отверстия.



- **Формат вызова**

G65 P9100 { Zz / Ww } Kk Ff ;

Z: Глубина отверстия (абсолютное значение)
U: Глубина отверстия (значение в приращениях)
K: Величина резания за цикл
F: Рабочая подача

- **Программа, вызывающая макропрограмму**

```
O0002 ;
G50 X100.0 Z200.0 ;
G00 X0 Z102.0 S1000 M03 ;
G65 P9100 Z50.0 K20.0 F0.3 ;
G00 X100.0 Z200.0 M05 ;
M30 ;
```

- **Макропрограмма (вызванная программа)**

```
  O9100 ;
  #1=0; ..... Стирает данные глубины текущего отверстия.
  #2=0; ..... Стирает данные глубины предыдущего отверстия.
  IF [#23 NE #0] GOTO 1 ;
 . . При инкрементном программировании задает переход к N1.
  IF [#26 EQ #0] GOTO 8 ;
. . . . . Если ни Z, ни W не задано, то возникает ошибка.
  #23=#5002-#26 ; Вычисляет глубину отверстия.
N1 #1=#1+#6 ; Вычисляет глубину текущего отверстия.
  IF [#1 LE #23] GOTO 2 ;
. . . . Определяет, не слишком ли глубоко прорезано отверстие.
  #1=#23 ;
. . . . Осуществляет фиксацию на глубине текущего отверстия.
N2 G00 W-#2 ; . . . . . Перемещает инструмент на глубину
                предыдущего отверстия на скорости подачи резания.
  G01 W- [#1-#2] F#9 ; . . . . . Выполняет сверление отверстия.
  G00 W#1 ;
. . . . . Перемещает инструмент в начальную точку сверления.
  IF [#1 GE #23] GOTO 9 ;
. . . . . Осуществляет проверку завершения сверления.
  #2=#1 ; . . . Записывает в память глубину текущего отверстия.
  GOTO 1;
N9 M99 ;
N8 #3000=1 (NOT Z OR U COMMAND)
```