

Глава 5

Буферизованный вывод



Буферизованный вывод

Для повышения гибкости при формировании файла управляющей программы в системе введен режим буферизованного вывода.

Режим буферизованного вывода

В данном режиме изменён подход к формированию машинной программы. Если в стандартном режиме вывод информации происходит в виде готовых кадров управляющей программы непосредственно в файл, то в расширенном режиме информация о кадре накапливается во внутреннем буфере постпроцессора.

При этом имеется возможность перемещаться по буферу, изменять, добавлять и очищать поля, а при необходимости разбивать или добавлять кадры.

При этом происходит не только автоматический пропуск пустых кадров, нередко появлявшихся при использовании стандартного метода, но и автоматическая нумерация кадров. Для реализации данного режима добавлено несколько специализированных функций, а значения программируемых полей-адресов задаются с помощью предопределённых переменных.

Специальные функции расширенного режима

Функции работы с генерируемой машинной программой

MASH_CREATE()

Пример вызова

CALL MASH_CREATE()

Параметры

Нет

Использование

Основная функция расширенного постпроцессора. Должна быть вызвана до запуска основного цикла постпроцессирования. При её вызове создаётся и инициализируется внутренний буфер кадров. Используется совместно с MASH_FINISH().

MASH_FINISH()

Пример вызова

CALL MASH_FINISH()

Параметры

Нет

Использование

Происходит сброс содержимого внутреннего буфера кадров постпроцессора в файл машинной программы, после этого буфер очищается.

Общая структура главного раздела табличного постпроцессора выглядит следующим образом:

```
' . . . инициализация переменных состояния.
call MASH_CREATE()
DO ' Начало цикла.
  CALL NEXT_STEP () ' Извлечение очередной команды.
  '                 . . .
  ' . . . Тело цикла - обработка команд. . .
  '                 . . .
LOOP while command#<>END#
call MASH_FINISH()
```

Функции работы с кадрами

CDR_MOVE(n#)

Пример вызова

CALL CDR_MOVE(-2)

Параметры

n# - целое число определяющее смещение нового текущего кадра относительно текущего кадра.

Использование

Происходит, если это возможно, смещение текущего кадра. Смещение текущего кадра происходит таким образом, что относительно текущего кадра новый текущий кадр будет отстоять на n# позиций. Смещение текущего кадра невозможно, если перемещение происходит за границу буфера или на несуществующий кадр.

Используется если необходимо выполнить анализ предыдущих/последующих кадров программы, не производится инициализация переменных состояния.

CDR_CREATE()

Пример вызова

CALL CDR_CREATE()

Параметры

нет.

Использование

создаётся кадр следующий за текущим, текущим становится созданный кадр, инициализируются переменные состояния.

CDR_DELETE()

Пример вызова

CALL CDR_DELETE()

Параметры

нет.

Использование

очищает текущий кадр, смещение текущего кадра не производится, так как пустые кадры не выводятся, а также производится инициализация переменных состояния.

CDR_ERASE()

Пример вызова

CALL CDR_ERASE()

Параметры

нет.

Использование

очищает текущий кадр, смещение текущего кадра не производится, так как пустые кадры не выводятся, а также не производится инициализация переменных состояния.

CDR_INIT()

Пример вызова

CALL CDR_INIT()

Параметры

нет.

Использование

Производится инициализация переменных состояния, никаких действий по отношению к текущему или иным кадрам не предпринимается.

CDR_FORWARD()

Пример вызова

CALL CDR_FORWARD()

Параметры

нет.

Использование

Происходит, если это возможно, смещение текущего кадра на один кадр вперёд. Смещение текущего кадра невозможно, если перемещение происходит за границу буфера или на несуществующий кадр.

Не производится инициализация переменных состояния.

CDR_REWARD()

Пример вызова

CALL CDR_REWARD()

Параметры

нет.

Использование

Происходит, если это возможно, смещение текущего кадра на один кадр назад. Смещение текущего кадра невозможно, если перемещение происходит за границу буфера или на несуществующий кадр.

Не производится инициализация переменных состояния.

Групповые функции

К "групповым" относятся такие функции для которых может быть активно только одно состояние, например функции круговой интерполяции по часовой стрелке и против часовой стрелке принадлежат одной группе, поскольку эти две интерполяции являются взаимоисключающими, поэтому из функций принадлежащих одной группе может быть вызвана только одна.

ЕСЛИ БУДЕТ ВЫЗВАНА ФУНКЦИЯ ИЗ ОДНОЙ ГРУППЫ ТО ОНА ЗАМЕСТИТ ФУНКЦИЮ ВЫЗВАННУЮ РАНЕЕ.

G_FUNC(f#)

Пример вызова

CALL G_FUNC(1)

Параметры

f# - целое число определяющее номер вызываемой функции.

Использование

В текущем кадре происходит изменение статуса адреса ассоциированного с

функцией G[f#].

Изменение происходит таким образом, что в данном кадре будет вызвана функция с номером f#, при этом соблюдается правило групп.

SET_G_FUNC_GROUP(f#, g#)

Пример вызова

CALL SET_G_FUNC_GROUP(4, 2)

Параметры

f# - целое число определяющее номер функции.

g# - целое число определяющее номер группы для указанной функции

Использование

функции f# перемещается в группу g#, допускается не более 255 групп функций, в одном кадре не может быть более одной функции принадлежащей одной группе ЕСЛИ БУДЕТ ВЫЗВАНА ФУНКЦИЯ ИЗ ОДНОЙ ГРУППЫ, ТО ОНА ЗАМЕСТИТ ФУНКЦИЮ ВЫЗВАННУЮ РАНЕЕ.

GET_G_FUNC_GROUP(f#, g#)

Пример вызова

CALL SET_G_FUNC_GROUP(4, 2)

Параметры

f# - целое число определяющее номер функции.

g# - целое число определяющее номер группы для указанной функции

Использование

можно узнать к какой группе принадлежит функция F#.

M_FUNC(f#)

добавляется вызов M-функции с номером f#.

SET_M_FUNC_GROUP(f#, g#)

Пример вызова

CALL SET_G_FUNC_GROUP(4, 2)

Параметры

f# - целое число определяющее номер функции.

g# - целое число определяющее номер группы для указанной функции

Использование

функции f# перемещается в группу g#, допускается не более 255 групп функций, в одном кадре не может быть более одной функции принадлежащей одной группе ЕСЛИ БУДЕТ ВЫЗВАНА ФУНКЦИЯ ИЗ ОДНОЙ ГРУППЫ, ТО ОНА ЗАМЕСТИТ ФУНКЦИЮ ВЫЗВАННУЮ РАНЕЕ.

функции F# перемещается в группу G#, допускается не более 255 групп функций, в одном блоке не может быть более одной функции принадлежащей одной группе ЕСЛИ БУДЕТ ВЫЗВАНА ФУНКЦИЯ ИЗ ОДНОЙ ГРУППЫ ТО ОНА ЗАМЕСТИТ ФУНКЦИЮ ВЫЗВАННУЮ РАНЕЕ.

GET_M_FUNC_GROUP(f#, g#)

Пример вызова

CALL SET_G_FUNC_GROUP(4, 2)

Параметры

f# - целое число определяющее номер функции.

g# - целое число определяющее номер группы для указанной функции

Использование

функции f# перемещается в группу g#, допускается не более 255 групп функций, в одном кадре не может быть более одной функции принадлежащей одной группе ЕСЛИ БУДЕТ ВЫЗВАНА ФУНКЦИЯ ИЗ ОДНОЙ ГРУППЫ, ТО ОНА ЗАМЕСТИТ ФУНКЦИЮ ВЫЗВАННУЮ РАНЕЕ.

можно узнать к какой группе принадлежит функция F#.

Переменные состояния

Все встроенные переменные расширенного постпроцессора начинаются с префикса **gpf_** (GeMMa Post Field)

поле - [code][value]

code - символ или символы определяющие значение тип поля.

value - значение поля, например номер инструмента или координата,

значение выводится в соответствии с форматом, который устанавливается

для каждого типа поля отдельно.

формат - [$][+][0]N[.[F[0]]]

здесь в квадратных скобках указаны необязательные значения.

N - имеет смысл только с лидирующим нулём и указывает на минимальное число выводимых знаков.

0 - перед N указывает на необходимость вывода лидирующих нулей.



\+ - указывает на обязательный вывод знака перед числом.

$ - указывает на необходимость вывода пробела перед полем.

. - указывает на необходимость вывода точки в числе.

F - определяет число знаков после запятой которое будет выведено.

0 - указывает на необходимость вывода завершающих нулей дополняющих значение до числа знаков указанных в F.

gpf_incorrect# - константа недостоверного целого числа.

gpf_incorrect - константа недостоверного числа с плавающей точкой.

gpf_incorrect$ - константа недостоверной строки.

gpf_x - координата x

gpf_y - координата y

gpf_z - координата z

gpf_a - поворот вокруг оси A

gpf_b - поворот вокруг оси B

gpf_c - поворот вокруг оси C

gpf_i - центр дуги

gpf_j - центр дуги

gpf_k - центр дуги

gpf_u - параллельная ось X

gpf_v - параллельная ось Y

gpf_w - параллельная ось Z

gpf_p - параметр

gpf_q - параметр

gpf_r - параметр

gpf_d - корректор

gpf_e - корректор

gpf_feed - подача

gpf_h# - корректор на длину

gpf_l# - метка подпрограммы

gpf_o# - заголовок программы

gpf_tool# - номер инструмента

gpf_spn# - скорость вращения шпинделя

gpf_gcode# - подготовительная функция

gpf_mcode# - вспомогательная функция

gpf_narrow$, gpf_narrow1$, gpf_narrow2$, gpf_narrow3$, gpf_narrow4$, gpf_narrow5$ - строки прямого кода, выводится без изменений и к ним не применяются форматы.

Любому полю соответствует символ и строка форматирования поля.

Для задания этих значений используются переменные соответственно с суффиксами

_fmt$ и _code$.

gpf_x_fmt$ - формат

gpf_x_code$ - символ

gpf_y_fmt$ - формат

gpf_y_code$ - символ

gpf_z_fmt$ - формат

gpf_z_code$ - символ

gpf_a_fmt$ - формат

gpf_a_code$ - символ

gpf_b_fmt$ - формат

gpf_b_code$ - символ

gpf_c_fmt$ - формат

gpf_c_code$ - символ

gpf_i_fmt$ - формат

gpf_i_code$ - символ

gpf_j_fmt$ - формат

gpf_j_code$ - символ

gpf_k_fmt$ - формат

gpf_k_code$ - символ

gpf_u_fmt$ - формат

gpf_u_code$ - символ

gpf_v_fmt$ - формат

gpf_v_code$ - символ

gpf_w_fmt$ - формат

gpf_w_code$ - символ

gpf_p_fmt$ - формат

gpf_p_code$ - символ

gpf_q_fmt$ - формат

gpf_q_code$ - символ

gpf_r_fmt$ - формат

gpf_r_code$ - символ

gpf_d_fmt$ - формат

gpf_d_code$ - символ

gpf_e_fmt$ - формат

gpf_e_code$ - символ

gpf_h_fmt$ - формат

gpf_h_code$ - символ

gpf_l_fmt$ - формат

gpf_l_code$ - символ

gpf_o_fmt$ - формат

gpf_o_code$ - символ

gpf_feed_fmt$ - формат подачи

gpf_feed_code$ - символ подачи

gpf_tool_fmt$ - формат поля инструмента

gpf_tool_code$ - символ инструмента

gpf_spn_fmt$ - формат поля вращения шпинделя

gpf_spn_code$ - символ поля вращиния шпинделя

gpf_gcode_fmt$ - формат

gpf_gcode_code$ - символ

gpf_mcode_fmt$ - формат

gpf_mcode_code$ - символ

gpf_NL_code$ - набор символов и кодов обозначающих переход на новую строку

Данные константы должны быть определены до вызова MASH_CREATE()

gpf_start_block$ - стартовая последовательность в программе (по умолчанию "%")

gpf_fini_block$ - завершающая последовательность в программе (по умолчанию "%%").

gpf_msh_file$ - имя выходного файла с машинной программой.

gpf_sequence$ - символ последовательности если пустой то не выводятся и номера кадров.

gpf_comment$ - поле комментария для текущего кадра

gpf_gcode_grp# - число групп G-функций

gpf_mcode_grp# - число групп M-функций

gpf_step# - шаг нумерации кадров.

gpf_cdr_frm$ - формат последовательности полей в кадре.

задаётся формат последовательности полей в кадре.

поля должны быть разделены символом "|", а заканчивать последовательность необходимо двойным символом "|" т.е. "‖" гарантируется что поля будут расставлены в порядке их появления в строке формата, поля не обозначенные символами форматирования например U,V,W будут выведены после отформатированных полей, в том порядке в котором они были созданы при постпроцессировании.

Строка форматирования должна быть определена до вызова MASH_CREATE()

gpf_buffer# - размер буфера кадров.

gpf_autoseq# - использовать ли автоматическую нумерацию кадров.

gpf_comment$ - строка комментария, выводится в скобках , т.е. "(COMMENT)".

Имена полей в строке форматирования кадра и их соответствие переменым.

seq - gpf_seq#

axisX - gpf_x

axisY - gpf_y

axisZ - gpf_z

axisI - gpf_i

axisJ - gpf_j

axisK - gpf_k

axisA - gpf_a

axisB - gpf_b

axisC - gpf_c

axisU - gpf_u

axisV - gpf_v

axisW - gpf_w

feed - gpf_feed

spindle - gpf_spn

tool - gpf_tool#

len_cor - gpf_h#

rad_cor - gpf_d

sub - gpf_l#

header - gpf_o#

parP - gpf_p

parQ - gpf_q

parR - gpf_r

DWELL - gpf_e

g-group<N> - Здесь <N> номер группы, например - g-group11

m-group<N> - Здесь <N> номер группы, например - m-group11

narrow_string - gpf_narrow$

narrow_string1 - gpf_narrow1$

narrow_string2 - gpf_narrow2$

narrow_string3 - gpf_narrow3$

narrow_string4 - gpf_narrow4$

narrow_string5 - gpf_narrow5$

Пример текста постпроцессора

```
REM       BERENC 17 Вырубной пресс          ; инкрементальный
'*************************************************************
'BRNC17NEW.PSS - файл спецификаций постпроцессора'
'Вид обработки               :  вырубной пресс
'Станок                      :  BERENC
'Система управления          :
'СТАТУС постпроцессора       :  внедрён на МРТЗ'
'Дискретность                :0.001 мм - декартовы перемещения.
'Основной режим              : в приращениях (инкрементальный).
'Круговые перемещения        :  да
'Нумерация кадров            :  да
'
'Функции :
'
'1. Коррекция на длину                   : нет
'2. Коррекция на радиус инструмента      : нет
'3. Смена инструмента                    : автоматическая
'4. Поворот стола                        : нет
'5. Смена системы координат              : нет
'6. Программируемый стоп                 : нет
'7. Циклы сверления                      : нет
'8. Комментарии на смену инструмента     : нет
'9. Комментарии на проход                : нет
'10. Круговая интерполяция               : XY
'11. Поддержка подпрограмм               : нет
'12. Торможение в конце кадра            : нет
'13. Смена системы координат             : нет
'
'Разработчики                : НТЦ "ГеММа"
'Дата последней модификации: 01.11.2003
'Версия ГеММы                : 8.0 и выше
'*************************************************************
```

```
'Особенности:
' в начале программы выводится список всех используемых
инструментов
'*****************************************************************
'Модификация:
'
'*****************************************************************
'
'
'*****************************************************************
'Раздел инициализации содержит
'присвоение начальных значений основным параметрам
'и определение имен констант, вводимых пользователем

const RDISCR  =  0.001
'присваивание параметру RDISCR значения 0.001
'дискретность по перемещениям  0.001 мм

'инициализация констант
'стандартная инициализация форматов и символов кадра

const gpf_buffer# = 10000    ' задаём объём буфера равным 10000
кадров.
const gpf_x_fmt$  = "$8"
const gpf_x_code$ = "X"
const gpf_y_fmt$  = "$1."
const gpf_y_code$ = "Y"
const gpf_z_fmt$  = "$1."
const gpf_z_code$ = "Z"
const gpf_i_fmt$  = "$1."
const gpf_i_code$ = "I"
const gpf_j_fmt$  = "$1."
const gpf_j_code$ = "J"
const gpf_k_fmt$  = "$1."
const gpf_k_code$ = "K"
const gpf_a_fmt$  = "1."
const gpf_a_code$ = "A"
const gpf_b_fmt$  = "1."
const gpf_b_code$ = "B"
const gpf_c_fmt$  = "1."
const gpf_c_code$ = "C"
const gpf_u_fmt$  = "1."
const gpf_u_code$ = "U"
const gpf_v_fmt$  = "1."
const gpf_v_code$ = "V"
const gpf_w_fmt$  = "1."
const gpf_w_code$ = "W"
const gpf_p_fmt$  = "1."
const gpf_p_code$ = "P"
const gpf_q_fmt$  = "1."
const gpf_q_code$ = "Q"
const gpf_r_fmt$  = "1."
const gpf_r_code$ = "R"
const gpf_d_fmt$  = "1."
const gpf_d_code$ = "D"
const gpf_e_fmt$  = "1"
const gpf_e_code$ = "E"
const gpf_h_fmt$  = "1."
```



```
const gpf_h_code$ = "H"
const gpf_l_fmt$  = "1."
const gpf_l_code$ = "L"
const gpf_o_fmt$  = "04"
const gpf_o_code$ = "O"
const gpf_feed_fmt$   = "1"
const gpf_feed_code$  = "F"
const gpf_tool_fmt$   = "02"
const gpf_tool_code$  = "T"
const gpf_spn_fmt$    = "1"
const gpf_spn_code$   = "S"
const gpf_gcode_fmt$  = "$02"
const gpf_gcode_code$ = "G"
const gpf_mcode_fmt$  = "$02"
const gpf_mcode_code$ = "M"
const gpf_NL_code$    = NL$
const gpf_msh_file$   = fileout$
const gpf_sequence$   = "N"
const gpf_seq_fmt$    = "4"
const gpf_step#       = 1          'Шаг номеров кадров
const gpf_num_start#  = 1          'Первый номер кадра
const gpf_start_block$= "%"+NL$+":"+STR$(partno#)
const gpf_fini_block$ = "%%"
const gpf_cdr_frm$ = "|narrow_string|seq|g-group0|g-group7|
g-
group8|axisX|axisY|axisZ|spn|feed|axisA|axisB|axisC|axisI|axis
J|axisK|tool|m-group0|comment||"
const gpf_autoseq# = TRUE#   'Автоматическая нумерация кадров.

'Установка групп используемых G-функций
'По умолчанию в системе может установлена 255 групп,
'но все функции принадлежат нулевой группе.
call set_g_func_group(30, 4)
call set_g_func_group(57, 5)
call set_g_func_group(81, 7)
call set_g_func_group(90, 8)
call set_g_func_group(91, 8)

call MASH_CREATE()                              ' создаёт
новую программу, если постпроцессор

' создаёт только одну машинную программу, то

' данную функцию можно и не вызывать, она

' будет вызвана системой неявно при первой

' необходимости, следует учитывать, что константы

' должны быть определены до её вызова.
DO
'цикл вывода комментария в начале программы по
'используемому инструменту
CALL NEXT_STEP()
 if   command# = HEADER#   then
    call Header()
  else if command# = LOAD_TOOL# then
    const gpf_seq# = -1
```

```
'При автоматической нумерации кадров

'если задать номер кадра -1, то данный

'кадр будет выведен без номера и он не будет

'учитываться в нумерации
    const gpf_comment$ = "T"+STR$(cutter_no#) + " " +
STR$(cutter_diameter) + " * "+STR$(cutter_rf)
    call cdr_create()
  endif
LOOP while command# <> END#

CALL FIRST_STEP()             'возвращаемся к началу обработки

' стандартный цикл обработки комманд
DO
CALL NEXT_STEP()

'В зависимости от вида анализируемой команды
'т.е. значения параметра command
'описываем действия постпроцессора по
'формированию управляющей программы :

  if   command# = END#   then
     call ProgramEnd()

  elseif   command# = LOAD_TOOL#  then
    call LoadTool()

  elseif   command# = MOVE#   then
    call MachineMove()
  endif
LOOP while command#<>END#

call MASH_FINISH()


'////////////////////////////////////////////////////////////
///////////
'///////////command# = HEADER# ////////////////////////
'////////////////////////////////////////////////////////////
///////////
'Вывод информации в начале программы.
Sub Header()
    'Вывод информации в начале программы.
    'Вывод заголовка программы : %partno#
    'производится с помощью глобальной переменной
gpf_start_block$
    if(partno_text$ <> "") then
      const gpf_narrow$ = "(" + partno_text$ + ")"
      call cdr_create()
    endif
EndSub

'////////////////////////////////////////////////////////////
///////////
'////////////// command# = END#    ////////////////////////
```



```
'////////////////////////////////////////////////////////////
//////////
'Вывод информации в конце программы.
Sub ProgramEnd()

  'Вывод информации в конце программы.
  'В конце программы перемещаемся в нулевую точку.
   ' "N# G57 X0 Y0"
   ' "N# M30"
   const gpf_gcode# = 57
   const gpf_x = 0.
   const gpf_y = 0.
   call CDR_CREATE()
   const gpf_gcode# = 30
   call cdr_create()
EndSub

'////////////////////////////////////////////////////////////
//////////
'//////// command# = LOAD_TOOL#     ////////////////
'////////////////////////////////////////////////////////////
//////////
'Вывод информации,
'связанной со сменой инструмента.

Sub LoadTool()
  ' "N# G81 G91 X Y T"
  const gpf_gcode# = 81
  const gpf_x = dx*RDISCR
  const gpf_y = dy*RDISCR
  const gpf_tool# = cutter_no#
  call G_FUNC(91)
  call CDR_CREATE()
EndSub
'////////////////////////////////////////////////////////////
//////////
'////////// command# = MOVE#  ////////////////////////
'////////////////////////////////////////////////////////////
//////////
'Вывод информации, связанной с перемещением
'рабочих органов станка.

Sub MachineMove()
   'Обрабатываются только линейные перемещения
   if (primitive# = LOCATE#) then
    'Перемещаемся в точку с ударом.
    if(dx <> 0.) then const gpf_x = dx*RDISCR
    if(dy <> 0.) then const gpf_y = dy*RDISCR
   else if(primitive# = LINE#) then
    'Вырубаем вдоль линии
     if(punch# > 1) then
       if(dx <> 0.) OR (dy <> 0.) then
         if (dx <> 0.) then const gpf_x = len *
RDISCR/(punch#-1)
         if (dy <> 0.) then const gpf_y = len *
RDISCR/(punch#-1)
         const gpf_1# = punch# - 1
       else
```

```
          const gfp_narrow$ =  "(ERROR, oshibka v proxode: dx
= 0 and dy = 0 no of punch > 1)"
        endif
      else
        if(dx <> 0.) then const gpf_x = dx*RDISCR
        if(dy <> 0.) then const gpf_y = dy*RDISCR
      endif
   else if(primitive# = ARC_CLW#) then
' обрабатываются круговые перемещения
      if(primitive# <> primitive_#) then
        gpf_gcode# = 2
      endif
     if(dx <> 0.) then cosnt gpf_x = dx*RDISCR
     if(dy <> 0.) then const gpf_y = dy*RDISCR
     if(di <> 0.) then const gpf_i = di*RDISCR
     if(dj <> 0.) then const gpf_j = dj*RDISCR
   else if(primitive# = ARC_CCLW#) then
 ' обрабатываются круговые перемещения
      if( primitive# <> primitive_#) then
         gpf_gcode# = 3
      endif
      if(dx <> 0.) then const gpf_x = dx*RDISCR
      if(dy <> 0.) then const gpf_y = dy*RDISCR
      if(di <> 0.) then const gpf_i = di*RDISCR
      if(dj <> 0.) then const gpf_j = dj*RDISCR
   endif
   'Если высечной режим
   if(mode# = NIBBLING# )then                    'Высечка
     const gpf_e =  nib_step                     'Выводим шаг
   else if(mode# = PUNCH# AND primitive# <> LINE#) then
'Вырубка
      const gpf_l# =  punch#        'Выводим число ударов
   endif
   call CDR_CREATE()
EndSub

'*************************************************************
*********
'конец описания и конец файла brnc17new.PSS
'*************************************************************
```