```
    catch {unset mom_wire_cutcom_adjust_register}

    set is_from ""

    OPEN_files ; #open warning and listing files
    LIST_FILE_HEADER ; #list header in commentary listing
}


#=============================================================
proc PB_DELAY_TIME_SET { } {
#=============================================================
  global mom_sys_delay_param mom_delay_value
  global mom_delay_revs mom_delay_mode delay_time

   # post builder provided format for the current mode:
    if {[info exists mom_sys_delay_param(${mom_delay_mode},format)]
    != 0} {
      MOM_set_address_format dwell
$mom_sys_delay_param(${mom_delay_mode},format)
    }

    switch $mom_delay_mode {
      SECONDS {set delay_time $mom_delay_value}
      default {set delay_time $mom_delay_revs}
    }
}


#=============================================================
proc MOM_before_motion { } {
#=============================================================
  global mom_motion_event mom_motion_type

    FEEDRATE_SET


    switch $mom_motion_type {
      ENGAGE   {PB_engage_move}
      APPROACH {PB_approach_move}
      FIRSTCUT {PB_first_cut}
    }
```

```
    if { [llength [info commands PB_CMD_kin_before_motion]] }
{ PB_CMD_kin_before_motion }
    if { [llength [info commands PB_CMD_before_motion]] }
{ PB_CMD_before_motion }
}


#=============================================================
proc MOM_start_of_group {} {
#=============================================================
  global mom_sys_group_output mom_group_name group_level ptp_file_name
  global mom_sequence_number mom_sequence_increment
mom_sequence_frequency
  global mom_sys_ptp_output pb_start_of_program_flag

    if {[regexp NC_PROGRAM $mom_group_name] == 1} {set group_level 0 ;
return}

    if {[hiset mom_sys_group_output]} { if {$mom_sys_group_output ==
"OFF"} {set group_level 0 ; return}}
    if {[hiset group_level]} {incr group_level} else {set group_level
1}
    if {$group_level > 1} {return}

    SEQNO_RESET ; #<4133654>
    MOM_reset_sequence $mom_sequence_number $mom_sequence_increment
$mom_sequence_frequency

    if {[info exists ptp_file_name]} {
      MOM_close_output_file $ptp_file_name ; MOM_start_of_program
      if {$mom_sys_ptp_output == "ON"} {MOM_open_output_file
$ptp_file_name }
    } else {
      MOM_start_of_program
    }

    PB_start_of_program ; set pb_start_of_program_flag 1
}
```

```
#=============================================================
proc MOM_machine_mode {} {
#=============================================================
  global pb_start_of_program_flag

    if {$pb_start_of_program_flag == 0} {PB_start_of_program ; set
pb_start_of_program_flag 1}
    catch {PB_CMD_machine_mode}
}
############### EVENT HANDLING SECTION ################


#=============================================================
proc PB_start_of_program { } {
#=============================================================

   if { [llength [info commands PB_CMD_kin_start_of_program]] } {
      PB_CMD_kin_start_of_program
   }

   MOM_set_seq_off
   MOM_do_template rewind_stop_code
   MOM_set_seq_on
   MOM_force Once G_cutcom G_plane G_feed G_mode
   MOM_do_template absolute_mode
}


#=============================================================
proc MOM_start_of_path { } {
#=============================================================
  global first_linear_move ; set first_linear_move 0
   TOOL_SET MOM_start_of_path

   if { [llength [info commands PB_CMD_kin_start_of_path]] } {
      PB_CMD_kin_start_of_path
   }

   PB_CMD_start_of_operation_force_addresses
}
```

```
#=============================================================
proc MOM_from_move { } {
#=============================================================
  global mom_feed_rate mom_feed_rate_per_rev  mom_motion_type
mom_kin_max_fpm
   COOLANT_SET ; CUTCOM_SET ; SPINDLE_SET ; RAPID_SET
}


#=============================================================
proc MOM_first_tool { } {
#=============================================================
  global mom_tool_change_type mom_manual_tool_change
   if {[info exists mom_tool_change_type]} {
      switch $mom_tool_change_type {
         MANUAL { PB_manual_tool_change }
         AUTO   { PB_auto_tool_change }
      }
   } elseif {[info exists mom_manual_tool_change]} {
      if {$mom_manual_tool_change == "TRUE"} {
         PB_manual_tool_change
      }
   }
}


#=============================================================
proc PB_auto_tool_change { } {
#=============================================================
   PB_CMD_tool_change_force_addresses
   MOM_force Once G_mode G Z
   MOM_do_template tool_change
   PB_CMD_start_of_alignment_character
   MOM_force Once T M
   MOM_do_template tool_change_1
   PB_CMD_end_of_alignment_character
   MOM_do_template tool_change_2
}


#=============================================================
proc PB_manual_tool_change { } {
```

```
#=============================================================
   PB_CMD_tool_change_force_addresses
   MOM_do_template stop
}


#=============================================================
proc MOM_initial_move { } {
#=============================================================
  global mom_feed_rate mom_feed_rate_per_rev mom_motion_type
  global mom_kin_max_fpm mom_motion_event
   COOLANT_SET ; CUTCOM_SET ; SPINDLE_SET ; RAPID_SET

  global mom_programmed_feed_rate
   if { [EQ_is_equal $mom_programmed_feed_rate 0] } {
      MOM_rapid_move
   } else {
      MOM_linear_move
   }
}

#=============================================================
proc MOM_first_move { } {
#=============================================================
  global mom_feed_rate mom_feed_rate_per_rev mom_motion_type
  global mom_kin_max_fpm mom_motion_event
   COOLANT_SET ; CUTCOM_SET ; SPINDLE_SET ; RAPID_SET
   catch {MOM_$mom_motion_event}
}


#=============================================================
proc PB_approach_move { } {
#=============================================================
}


#=============================================================
proc PB_engage_move { } {
#=============================================================
}
```

```
#=============================================================
proc PB_first_cut { } {
#=============================================================
}


#-------------------------------------------------------------
proc PB_first_linear_move { } {
#=============================================================
}


#=============================================================
proc PB_return_move { } {
#=============================================================
}


#=============================================================
proc MOM_gohome_move { } {
#=============================================================
   MOM_rapid_move
}


#=============================================================
proc MOM_end_of_path { } {
#=============================================================
}


#=============================================================
proc MOM_end_of_program { } {
#=============================================================
   MOM_do_template end_of_program
   MOM_set_seq_off
   MOM_do_template rewind_stop_code

#**** The following procedure lists the tool list with time in
commentary data
```

```
   LIST_FILE_TRAILER

#***** The following procedure closes the warning and listing files
   CLOSE_files
}


#=============================================================
proc MOM_tool_change { } {
#=============================================================
  global mom_tool_change_type mom_manual_tool_change
   if {[info exists mom_tool_change_type]} {
      switch $mom_tool_change_type {
         MANUAL { PB_manual_tool_change }
         AUTO   { PB_auto_tool_change }
      }
   } elseif {[info exists mom_manual_tool_change]} {
      if {$mom_manual_tool_change == "TRUE"} {
         PB_manual_tool_change
      }
   }
}


#=============================================================
proc MOM_length_compensation { } {
#=============================================================
   TOOL_SET MOM_length_compensation
   MOM_do_template tool_length_adjust
}


#=============================================================
proc MOM_set_modes { } {
#=============================================================
   MODES_SET
}


#=============================================================
proc MOM_spindle_rpm { } {
#=============================================================
```

```
   SPINDLE_SET
   MOM_force Once S M_spindle
   MOM_do_template spindle_rpm
}


#=============================================================
proc MOM_spindle_off { } {
#=============================================================
   MOM_do_template spindle_off
}


#=============================================================
proc MOM_coolant_on { } {
#=============================================================
   COOLANT_SET
}


#=============================================================
proc MOM_coolant_off { } {
#=============================================================
   COOLANT_SET
   MOM_do_template coolant_off
}


#=============================================================
proc PB_feedrates { } {
#=============================================================
}


#=============================================================
proc MOM_cutcom_on { } {
#=============================================================
   CUTCOM_SET
}
```

```
#=============================================================
proc MOM_cutcom_off { } {
#=============================================================
   CUTCOM_SET
   MOM_do_template cutcom_off
}


#=============================================================
proc MOM_delay { } {
#=============================================================
   PB_DELAY_TIME_SET
   MOM_do_template delay
}


#=============================================================
proc MOM_opstop { } {
#=============================================================
   MOM_do_template opstop
}


#=============================================================
proc MOM_auxfun { } {
#=============================================================
   MOM_do_template auxfun
}


#=============================================================
proc MOM_prefun { } {
#=============================================================
   MOM_do_template prefun
}


#=============================================================
proc MOM_load_tool { } {
#=============================================================
  global mom_tool_change_type mom_manual_tool_change
}
```

```
#=============================================================
proc MOM_stop { } {
#=============================================================
   MOM_do_template stop
}


#=============================================================
proc MOM_tool_preselect { } {
#=============================================================
  global mom_tool_preselect_number mom_tool_number
mom_next_tool_number
   if {[info exists mom_tool_preselect_number]} {
      set mom_next_tool_number $mom_tool_preselect_number
   }
   MOM_do_template tool_preselect
}


#=============================================================
proc MOM_linear_move { } {
#=============================================================
  global feed_mode mom_feed_rate mom_kin_rapid_feed_rate

   if { $feed_mode == "IPM" || $feed_mode == "MMPM" } {
      if { [EQ_is_ge $mom_feed_rate $mom_kin_rapid_feed_rate] } {
         MOM_rapid_move ; return
      }
   }

  global first_linear_move

   if {!$first_linear_move} { PB_first_linear_move ; incr
first_linear_move }

   MOM_do_template linear_move
}


#=============================================================
```

```
proc MOM_circular_move { } {
#=============================================================
   CIRCLE_SET
   MOM_do_template circular_move
}


#=============================================================
proc MOM_rapid_move { } {
#=============================================================
  global rapid_spindle_inhibit rapid_traverse_inhibit
  global spindle_first is_from
  global mom_cycle_spindle_axis traverse_axis1 traverse_axis2
  global mom_motion_event

   set aa(0) X ; set aa(1) Y ; set aa(2) Z
   RAPID_SET
   set spindle_block rapid_spindle
   set traverse_block rapid_traverse
   if {$spindle_first == "TRUE"} {
      if {$rapid_spindle_inhibit == "FALSE"} {
         if { $mom_motion_event == "initial_move" || $mom_motion_event
== "first_move" } {
            MOM_force once $aa($mom_cycle_spindle_axis)
         }

         MOM_suppress once $aa($traverse_axis1) $aa($traverse_axis2)
         MOM_do_template $spindle_block $is_from
         MOM_suppress off $aa($traverse_axis1) $aa($traverse_axis2)
      }
      if {$rapid_traverse_inhibit == "FALSE"} {

         if { $mom_motion_event == "initial_move" || $mom_motion_event
== "first_move" } {
            MOM_force once $aa($traverse_axis1) $aa($traverse_axis2)
         }

         MOM_suppress once $aa($mom_cycle_spindle_axis)
         MOM_do_template $traverse_block $is_from
         MOM_suppress off $aa($mom_cycle_spindle_axis)
      }
```

```
   } elseif {$spindle_first == "FALSE"} {
      if {$rapid_traverse_inhibit == "FALSE"} {

         if { $mom_motion_event == "initial_move" || $mom_motion_event
== "first_move" } {
            MOM_force once $aa($traverse_axis1) $aa($traverse_axis2)
         }

         MOM_suppress once $aa($mom_cycle_spindle_axis)
         MOM_do_template $traverse_block $is_from
         MOM_suppress off $aa($mom_cycle_spindle_axis)
      }
      if {$rapid_spindle_inhibit == "FALSE"} {

         if { $mom_motion_event == "initial_move" || $mom_motion_event
== "first_move" } {
            MOM_force once $aa($mom_cycle_spindle_axis)
         }

         MOM_suppress once $aa($traverse_axis1) $aa($traverse_axis2)
         MOM_do_template $spindle_block $is_from
         MOM_suppress off $aa($traverse_axis1) $aa($traverse_axis2)
      }
   } else {

      if { $mom_motion_event == "initial_move" || $mom_motion_event ==
"first_move" } {
         MOM_force once $aa($traverse_axis1) $aa($traverse_axis2)
$aa($mom_cycle_spindle_axis)
      }

      MOM_do_template $traverse_block $is_from
   }
}


#=============================================================
proc MOM_cycle_off { } {
#=============================================================
   MOM_do_template cycle_off
}
```

```
#=========================================================
proc MOM_cycle_plane_change { } {
#=========================================================
  global cycle_init_flag

   set cycle_init_flag TRUE
}


#=========================================================
proc MOM_drill { } {
#=========================================================
  global cycle_name
  global cycle_init_flag

   set cycle_init_flag TRUE
   set cycle_name DRILL
   CYCLE_SET
}

#=========================================================
proc MOM_drill_move { } {
#=========================================================
  global cycle_init_flag

   MOM_do_template cycle_drill
   set cycle_init_flag FALSE
}


#=========================================================
proc MOM_drill_dwell { } {
#=========================================================
  global cycle_name
  global cycle_init_flag

   set cycle_init_flag TRUE
   set cycle_name DRILL_DWELL
   CYCLE_SET
}
```

```
#=============================================================
proc MOM_drill_dwell_move { } {
#=============================================================
  global cycle_init_flag

   MOM_do_template cycle_drill_dwell
   set cycle_init_flag FALSE
}


#=============================================================
proc MOM_drill_deep { } {
#=============================================================
  global cycle_name
  global cycle_init_flag

   set cycle_init_flag TRUE
   set cycle_name DRILL_DEEP
   CYCLE_SET
}


#=============================================================
proc MOM_drill_deep_move { } {
#=============================================================
  global cycle_init_flag

   MOM_do_template cycle_drill_deep
   set cycle_init_flag FALSE
}


#=============================================================
proc MOM_drill_break_chip { } {
#=============================================================
  global cycle_name
  global cycle_init_flag

   set cycle_init_flag TRUE
   set cycle_name DRILL_BREAK_CHIP
```

```
   CYCLE_SET
}


#=============================================================
proc MOM_drill_break_chip_move { } {
#=============================================================
  global cycle_init_flag

   MOM_do_template cycle_drill_break_chip
   set cycle_init_flag FALSE
}


#=============================================================
proc MOM_tap { } {
#=============================================================
  global cycle_name
  global cycle_init_flag

   set cycle_init_flag TRUE
   set cycle_name TAP
   CYCLE_SET
}


#=============================================================
proc MOM_tap_move { } {
#=============================================================
  global cycle_init_flag

   MOM_do_template cycle_tap
   set cycle_init_flag FALSE
}


#=============================================================
proc MOM_bore { } {
#=============================================================
  global cycle_name
  global cycle_init_flag
```

```
   set cycle_init_flag TRUE
   set cycle_name BORE
   CYCLE_SET
}


#=========================================================
proc MOM_bore_move { } {
#=========================================================
  global cycle_init_flag

   MOM_do_template cycle_bore
   set cycle_init_flag FALSE
}


#=========================================================
proc MOM_bore_drag { } {
#=========================================================
  global cycle_name
  global cycle_init_flag

   set cycle_init_flag TRUE
   set cycle_name BORE_DRAG
   CYCLE_SET
}


#=========================================================
proc MOM_bore_drag_move { } {
#=========================================================
  global cycle_init_flag

   MOM_do_template cycle_bore_drag
   set cycle_init_flag FALSE
}


#=========================================================
proc MOM_bore_no_drag { } {
#=========================================================
  global cycle_name
```

```
  global cycle_init_flag

   set cycle_init_flag TRUE
   set cycle_name BORE_NO_DRAG
   CYCLE_SET
}


#=========================================================
proc MOM_bore_no_drag_move { } {
#=========================================================
  global cycle_init_flag

   MOM_do_template cycle_bore_no_drag
   set cycle_init_flag FALSE
}


#=========================================================
proc MOM_bore_manual { } {
#=========================================================
  global cycle_name
  global cycle_init_flag

   set cycle_init_flag TRUE
   set cycle_name BORE_MANUAL
   CYCLE_SET
}


#=========================================================
proc MOM_bore_manual_move { } {
#=========================================================
  global cycle_init_flag

   MOM_do_template cycle_bore_manual
   set cycle_init_flag FALSE
}


#=========================================================
proc MOM_bore_dwell { } {
```

```
#=============================================================
  global cycle_name
  global cycle_init_flag

   set cycle_init_flag TRUE
   set cycle_name BORE_DWELL
   CYCLE_SET
}


#=============================================================
proc MOM_bore_dwell_move { } {
#=============================================================
  global cycle_init_flag

   MOM_do_template cycle_bore_dwell
   set cycle_init_flag FALSE
}


#=============================================================
proc MOM_bore_back { } {
#=============================================================
  global cycle_name
  global cycle_init_flag

   set cycle_init_flag TRUE
   set cycle_name BORE_BACK
   CYCLE_SET
}


#=============================================================
proc MOM_bore_back_move { } {
#=============================================================
  global cycle_init_flag

   MOM_do_template cycle_bore_back
   set cycle_init_flag FALSE
}
```

```
#=============================================================
proc MOM_bore_manual_dwell { } {
#=============================================================
  global cycle_name
  global cycle_init_flag

   set cycle_init_flag TRUE
   set cycle_name BORE_MANUAL_DWELL
   CYCLE_SET
}


#=============================================================
proc MOM_bore manual_dwell_move { } {
#=============================================================
  global cycle_init_flag

   MOM_do_template cycle_bore_manual_dwell
   set cycle_init_flag FALSE
}

#=============================================================
proc MOM_sequence_number { } {
#=============================================================
   SEQNO_SET
}


#=============================================================
proc PB_CMD_before_motion { } {
#=============================================================
}


#=============================================================
proc PB_CMD_end_of_alignment_character { } {
#=============================================================
 #  Return sequnece number back to orignal
 #  This command may be used with the command
"PM_CMD_start_of_alignment_character"
```

```
  global mom_sys_leader saved_seq_num
  set mom_sys_leader(N) $saved_seq_num
}


#=============================================================
proc PB_CMD_start_of_alignment_character { } {
#=============================================================
 # This command can be used to output a special sequence number
character.
 # Replace the ":" with any character that you require.
 # You must use the command "PB_CMD_end_of_alignment_character" to
reset
 # the sequence number back to the original setting.

  global mom_sys_leader saved_seq_num
  set saved_seq_num $mom_sys_leader(N)
  set mom_sys_leader(N) ":"
}


#=============================================================
proc PB_CMD_start_of_operation_force_addresses { } {
#=============================================================
  MOM_force once S M_spindle X Y Z F
}


#=============================================================
proc PB_CMD_tool_change_force_addresses { } {
#=============================================================
  MOM_force once G_adjust H
}


#=============================================================
proc DELAY_TIME_SET {  } {
#=============================================================
  global mom_sys_delay_param mom_delay_value
  global mom_delay_revs mom_delay_mode delay_time

   # post builder provided format for the current mode:
```

```
    if {[info exists mom_sys_delay_param(${mom_delay_mode},format)] !=
0} {
      MOM_set_address_format dwell
$mom_sys_delay_param(${mom_delay_mode},format)
    }

    switch $mom_delay_mode {
      SECONDS {set delay_time $mom_delay_value}
      default {set delay_time $mom_delay_revs}
    }
}
```

# 附录C　输出格式定义

## 格式代码：&abcdef

- a = + 或 _，“+”表示输出正号“+”；“_”表示省略正号“+”。
- b = 0 或 _，“0”表示输出前 0；“_”表示省略前 0。
- c = {0, 1,……,9}，小数点左边的位数。
- d = . 或_，“.”表示输出小数点；“_”表示省略小数点。
- e = {0,1,……,9}，小数点右边的位数。
- f = 0 或 _，“0”表示输出后 0；“_”表示省略后 0。

# 附录D 术 语

**Address——字地址。**在 NC 程序中，比如 X1.234、G01、M05、S1445，其中 X、G、M 和 S 就是字地址。告诉控制系统这些数据保存在什么地方。在 UG 中 Address、Word Address 和 Word 都指字地址。

**Block——程序行。**NC 程序中的一行。行是由一系列字地址组成的。如：N0100 G01 X1.234 Y1.234 Z1.234 F100. S1500 M03。

**Custom Command——用户命令。**在 Post Builder 中。用户可以建立一个 Tcl 子程序，然后加在 Post Builder 的事件或序列里，生成只用 Post Builder 不能输出的特殊指令。

**Definition file——定义文件。**是一个文件，后缀是“.def”。定义机床 / 控制系统的输出格式，也叫 def 文件。与 tcl 和 pui 文件同时使用。

**Event——事件。**由事件生成器产生并传给后处理器。后处理器处理的就是一系列事件，每一事件又包含一系列变量。UG/Post 处理一个事件就产生一个机床 / 控制系统执行的动作。

**Event Generator——事件生成器。**它提取刀轨信息整理成事件和变量传给 UG/Post。当你在 CAM 里选了操作，在 UG/Post 里选了后处理器，单击 OK 或 Apply 时，事件生成器就开始工作了。

**Event Handler——事件处理文件。**是一个文件，包含 Tcl 代码和子程序，决定事件的处理方式。每一个子程序决定是否输出 NC 程序，输出什么样的程序指令。UG 扩展了 Tcl 的功能，使得用户很容易在定义文件的基础上控制输出内容。事件处理文件还可以处理事件生成器产生的变量。

**Extensions——Tcl 功能扩展。**UG 扩展了 Tcl 的功能，加了一些指令方便于后处理输出和减少程序量。这些指令都以 MOM_开头，比如 MOM_do_template。

**Global variable——全局变量。**在事件处理文件中，在所有子程序都有效的标量。这些变量同样在定义文件中有效。如果想在几个子程序中都用这个变量，必须定义成全局变量。

**Leader——字头。**在定义文件中。也就是字地址，如 X、G 等。

**Marker——标记。**Post Builder 把整个后处理分成 5 个序列（sequence），每个序列又由一些标记组成。标记是可能出现的事件。每一个标记 / 事件又定义几个程序行的输出信息。Start_of_Path 就是一个标记。

**Mom_variable——CAM 输出变量。**事件生成器对每个事件定义的变量。通常是全局变量，如 mom_feed_rate。

**MOM——加工输出管理器（Manufacturing Output Manager）。**这是 UG CAM 的最基本功能，也用于输出车间工艺文档、CLS，访问 UG library，与 UG/Post 的操作。小写

的 mom 是变量名，如 mom_feed_rate；大写的 MOM 是时间处理名，如 MOM_linear_move；或 Tcl 扩展名，如 MOM_do_template。

**Operation End Sequence——操作结束序列。**Post Builder 中的一个序列。处理从最后退刀到操作结束之间的事件。

**Operation Start Sequence——操作开始序列。**Post Builder 中的一个序列（共有 5 个序列：程序头、操作头、刀轨、操作尾、程序尾）。处理从操作开始到第一个切削运动之间的事件。

**Operation Message——操作信息。**在 NC 程序中包含的给机床操作者看的信息。一般需要特殊代码，比如“（”或“（msg”），以区别于机床执行指令。例如：N0010 (***Tool Number 12***)。

**Postprocessor——后处理器或后处理文件。**是把刀轨信息转化成机床可接受代码的工具。在 UG/Post 中，它包括 3 个文件：事件处理文件（.tcl）、定义文件（.def）、Post Builder 界面文件（.pui）。在使用时这些文件被定义在模板文件 template_post.dat 中。

**Postprocessor template file——后处理模板文件。**这个文件里的内容是在 UG CAM 中可用后处理器的列表。每一行包括名称、事件处理文件和定义文件。模板文件在 CAM 的配置文件的 TEMPLATE_POST 部分指定。默认名是：template_post.dat。

**Procedure——子程序。**Tcl 中的子程序，有时也写作 proc。后处理中的每一个事件都对应事件处理文件中的一个子程序。如：MOM_start_of_program。

**Program End Sequence——程序结束序列。**Post Builder 中的一个序列。处理从最后一个操作到程序结束之间的事件。

**Program End Sequence——程序开始序列。**Post Builder 中的一个序列。处理从程序开始到第一个操作之间的事件。

**pui file——Post Builder 界面文件。**是一个文件，后缀是“.pui”。与事件处理文件（.tcl）和定义文件（.def）一起组成后处理器。在做后处理时并没有用到 pui 文件，但当用 Post Builder 修改后 tcl 和 def 文件时，是通过 pui 文件打开的。

**Review Tool——检查工具。**在 Post Builder 中把 Reiew Tool 打开后，在后处理时显示 3 个窗口，分别是事件、变量和输出内容。可以选取一个事件，看它有哪些变量以及输出结果。

**Sequence——序列。**Post Builder 把输出的 NC 程序分成 5 个部分：程序头、操作头、刀轨、操作尾、程序尾。每一部分是一个序列。用于组织处理事件的顺序和输出程序行的顺序，如：Operation_start_Sequence。

**Tcl——Tcl（Tool Command Language）计算机语言。**解释性执行语言，由于简便好用而很流行。还有两部分：Tk——用户交互工具；WISH —windows 指令，包括 Tk。

**Tk——Tcl 语言中使用的用户交互工具。**Tk 提供基本的交互工具如按键、选项框、滚动条等。

**tcl file——事件处理文件。**参看 Event Handler。

**UG/Post Execute——执行后处理。**UG/Post 是 UG 提供的后处理工具。用用户定义的事件处理文件和定义文件，把刀轨处理成机床代码。

**Post Builder——后处理构造器。**UG提供的一个图形界面构造后处理器的工具。可以用拖拽的方式创建格式、字地址、程序行、序列，定义输出内容和格式，控制程序头尾、操作头尾、换刀、循环等。

**User Defined Event——用户定义事件，简称UDE。**用于UG CAM的machine control中，可以在start post里，或end post里。可以附加在刀具或程序上。用户可以根据机床的要求增加或修改。

**Wish——Tcl和Tk的解释器和执行器，是Windows Shell的缩写。**如果要用UG的UG/Post Review Tool功能，必须安装ugwish.exe，在mach\auxiliary目录下。

**Word——字地址。**参看Address。

# 读者意见反馈卡

1. 您对本书的总体感觉:

□满意　　□一般　　□不满意

2. 您认为本书的层次结构:

□很好　　□一般　　□不好

3. 您认为本书的语言文字水平:

□很好　　□一般　　□不好

4. 您认为本书的版式编排:

□很好　　□一般　　□不好

5. 您认为本书中所涉及各项操作说明的准确性:

□准确　　□较准确　　□不准确

6. 您最需要哪方面的图书? _____

7. 您是从哪里第一次听说这本书的?

□书店　　□广告　　□从朋友、同事等处听说　　□其他

8. 您一年中购买计算机类图书的数量:

□2～5 本　　□6～10 本　　□多于 10 本

9. 您使用的操作系统是:

□DOS　　□Windows　　□OS/2　　□Macintosh　　□Unix　　□Linux　　□其他

10. 您感兴趣的计算机类新书为:

□操作系统类　　□办公软件类　　□程序设计语言类

□图形、图像设计类　　□排版软件类　　□网络技术类

□多媒体制作类　　□其他

11. 您使用 PC 机的地方:

□家庭　　□单位　　□学校　　□其他

12. 您认为本书还需改进的方面: _____

_____

_____

读者姓名:　　单位名称:　　联系电话:

请填好本卡后寄给:

清华大学校内金地公司(E-mail: thjd@thjd.com.cn)

《UG 后处理技术》编辑部收

邮编: 100084

联系电话:(010)62791976　　传真:(010)62788903

公司网址: www.thjd.com.cn

**如需本书可与本编辑部联系邮购,汇款请按以上地址填写,另加邮费 15%(挂号)**