Tcl Training Class

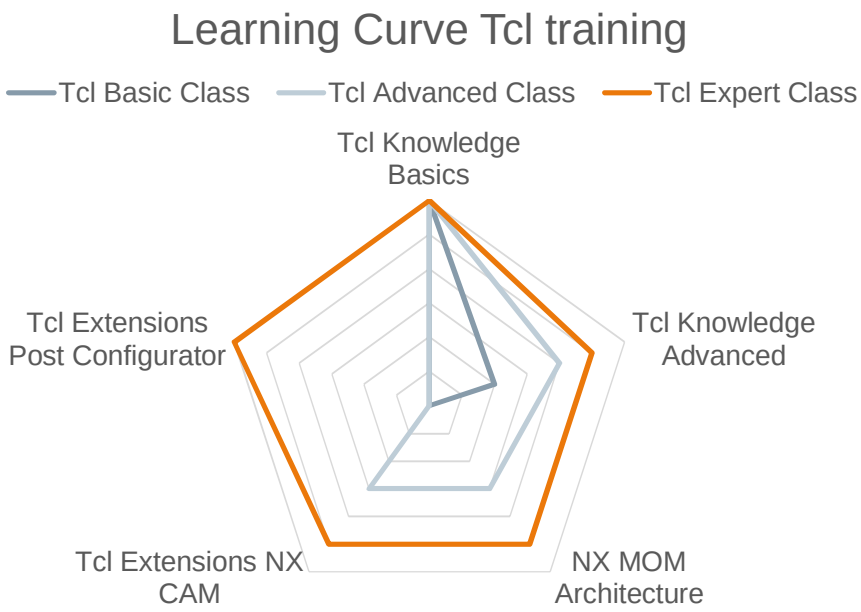
Advanced Class

Realize innovation.

Learning goals class

SIEMENS
Ingenuity for life

- ✓Advanced Tcl commands
- ✓Understanding the NX MOM
- ✓Using Tcl Extensions within NX Post Processor
- ✓Working with MOM commands
- ✓Overwriting existing commands
- ✓Understanding the Definitions files
- ✓Create post processor from scratch
- ✓How to help myself and useful tools



Tcl Advanced

SIEMENS
Ingenuity for life

Agenda:

Helpful documentation and tools for the class
NX MOM architecture
Advanced Tcl commands
Excercises

SIEMENS
Ingenuity for life

Agenda:

Helpful documentation and tools for the class
NX MOM architecture
Advanced Tcl commands
Excercises

Helpful Documentation and Tools for the Class

SIEMENS
Ingenuity for life

PDF Book for Tcl Basics as reference
http://www.freebookcentre.net/programming-books-download/Tcl-Basics-(PDF-118P).html

Wikibooks:
https://en.wikibooks.org/wiki/Tcl_Programming/Introduction

Online help Tcl Basics:
https://stackoverflow.com/questions/tagged/tcl

Siemens NX Manufacturing Forum -> NX CAM Postprocessor Group (Post Processor specific topics)
https://community.sw.siemens.com/s/group/0F94O0000005TFzSAM/nx-cam-postprocessor-group

Empty Tcl-file and Def-file will be used for this class, recommended to use Notepad++

General NX Help MOM commands and variables
https://docs.plm.automation.siemens.com/tdoc/nx/1899/nx_help#uid:xid1128418:index_xid917284

SIEMENS
Ingenuity for life

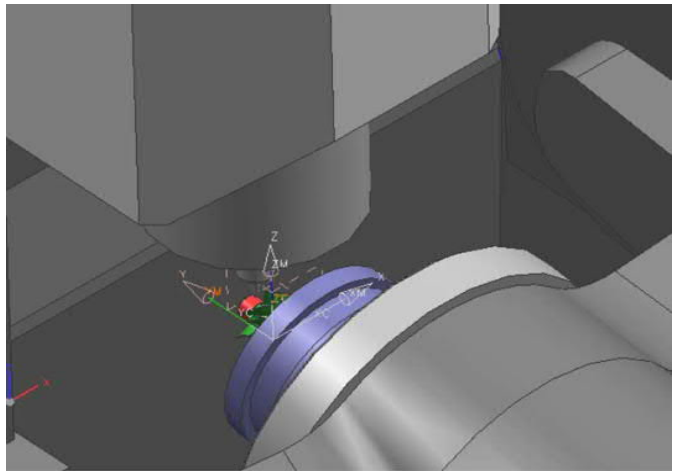
Agenda:

Helpful documentation and tools for the class
NX MOM architecture
Advanced Tcl commands
Excercises

Purpose of a CAM system

SIEMENS
Ingenuity for life

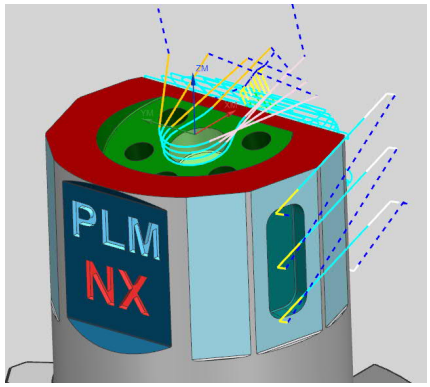
Use of software to control machine tools and related ones in the manufacturing of workpieces. The system define and generate sequences of tool positions (**Toolpath**) that can be used to guide a NC machine in order to produce the desired shape of a part.



NX CAM Tool Path

SIEMENS
Ingenuity for life

- Tool path contains data that represents the sequence of tool movements and other machine control instructions
- Tool path can be output in different formats in NX CAM




```
TOOL PATH/FACE_TOP,TOOL,UGT0202_001
TLDATA/MILL,40.0000,0.8000,34.6900,0.0000,0.0000
MSYS/0.0000,0.0000,100.0000,1.0000000,0.0000000,0.
$$ centerline data
PAINT/PATH
PAINT/SPEED,10
LOAD/TOOL,1,ADJUST,1
PAINT/COLOR,186
RAPID
GOTO/-79.5990,42.0000,50.0000,0.0000000,0.0000000,
PAINT/COLOR,211
RAPID
GOTO/-79.5990,42.0000,3.0000
PAINT/COLOR,6
FEDRAT/MMPM,1203.1000
GOTO/-79.5990,42.0000,0.0000
GOTO/-56.0000,42.0000,0.0000
PAINT/COLOR,31
GOTO/56.0000,42.0000,0.0000
```

Cutter Location Source Files (CLSF)

```
N110 SUPA X=_X_HOME Y=_Y_HOME A=_A_HOME C=_C_HOME D1
N120 T="UGT0202_001" M6
N130 MSG("MILL_FINISH")
N140 TRAFOOF
N150 SUPA Z=_Z_HOME D0
N160 SUPA X=_X_HOME Y=_Y_HOME A=_A_HOME C=_C_HOME D1
N170 CYCLE832(_camtolerance,1,1)
N180 COMPOF
N190 G54
N200 TRANS X0. Y0. Z0.
N210 G0 A0. C0.
N220 AROT Z0.0
N230 AROT Y0.0
N240 AROT X0.0
N250 TRAORI
N260 G17 X-79.599 Y42. Z50. S2228 D1 M3
N270 Z3.
N280 G1 G90 Z0. F1203.
N290 X-56.
N300 X56.
N310 X79.599
```

Post Processor (controller/ machine specific NC code)

Tcl Advanced

```
switch (camPathToolpathEventType)
{
    case CamPathToolpathEventType.Motion:
        path.IsToolpathEventAMotion(j, out camPathMotionType,
        //theSession.ListingWindow.WriteLine(j.ToString() + "

        switch (camPathMotionType)
        {
            case CamPathMotionType.From:
            case CamPathMotionType.Rapid:
            case CamPathMotionType.Approach:
            case CamPathMotionType.Engage:
            case CamPathMotionType.FirstCut:
            case CamPathMotionType.Cut:
            case CamPathMotionType.SideCut:
            case CamPathMotionType.Stepover:
            case CamPathMotionType.InternalLift:
            case CamPathMotionType.Retract:
            case CamPathMotionType.Traversal:
            case CamPathMotionType.Gohome:
```

Processed with NXOpen API

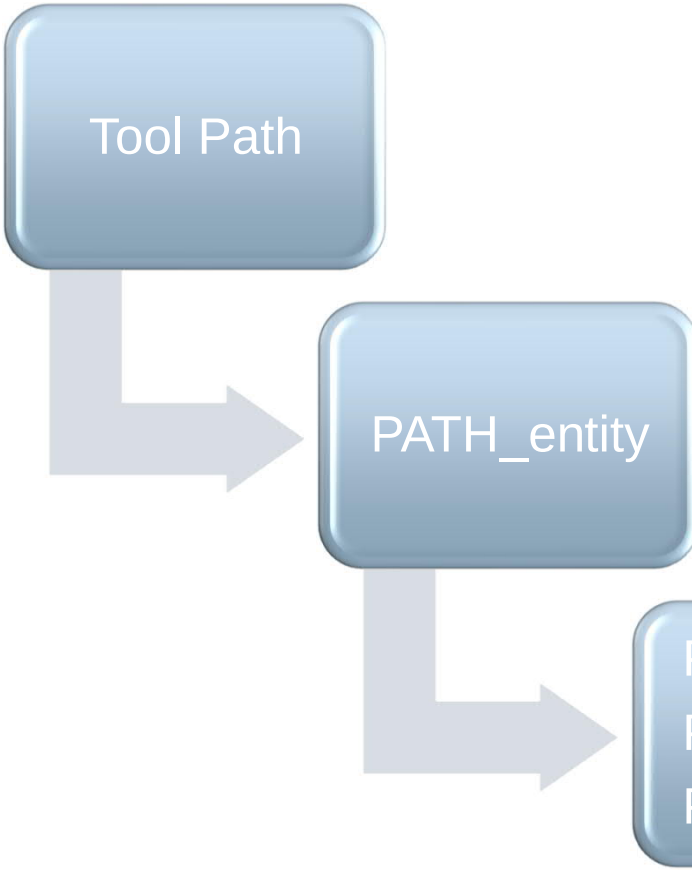
Data Model -OM

SIEMENS
Ingenuity for life

- The data model of CAM utilizes the Object Manager (OM) architecture of NX
- OM is an object oriented data structure
- It is written in C but has many concepts of Object Oriented Languages like C++
- OM has classes, creator, methods, method overriding
- The specialty of the OM is that it provides a way to establish relationships between different objects

Tool Path Data Structure

SIEMENS
Ingenuity for life



- Main holder of a list of Path Events (operation)
- Super class of all other types of tool path events (i.e. motion, machine control)

MOM (Manufacturing Output Manager)

SIEMENS
Ingenuity for life

Mechanism to access UG data and generate formatted output. UG data includes

- Part information
- CAM data
- **Tool Path**

CAM applications based on MOM

- **UG/Post**
- UG/Shop Docs
- UG Library Mechanism

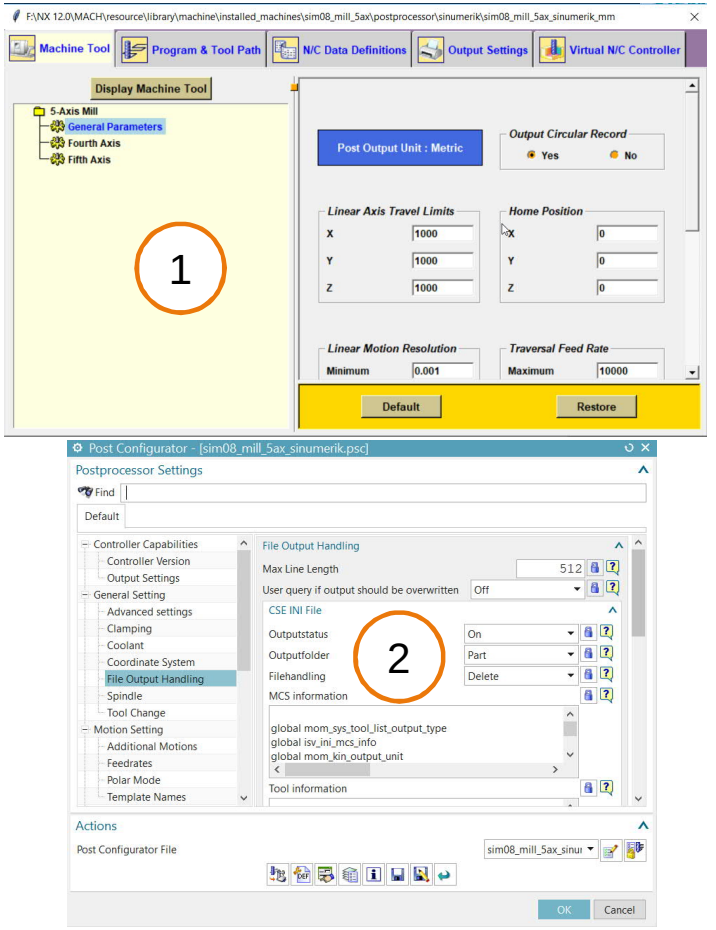
MOM contain a Tcl interface to extend/ access the data

- **< NX12.0.1 MP1 Tcl8.4**
- **> NX12.0.1 MP1 Tcl8.6**

What is NX CAM/Post ?

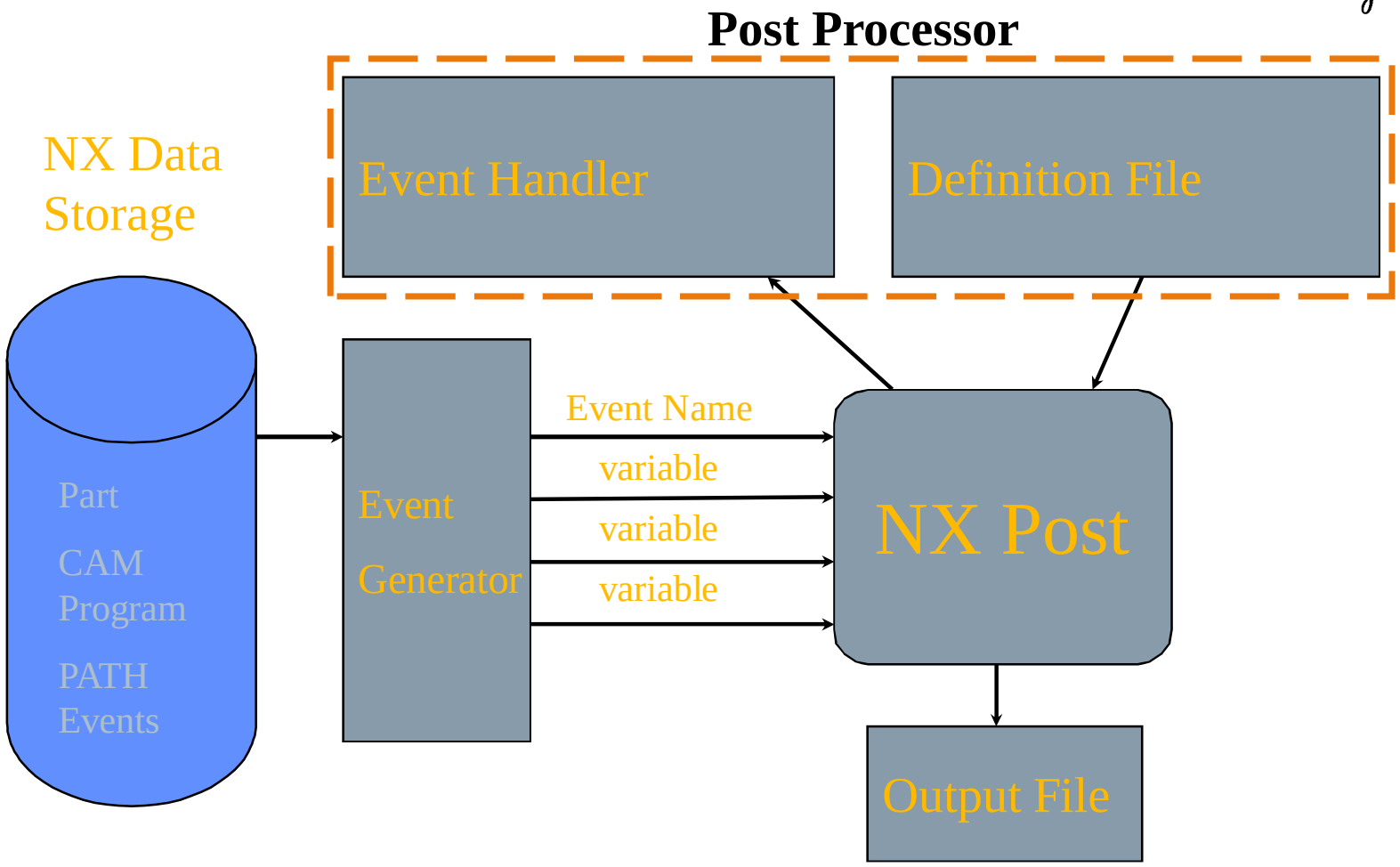
SIEMENS
Ingenuity for life

- Application based on MOM architecture
- Innovative approach (unique in CAM industry)
- Completely customizable & highly extensible
- Full capability postprocessor
- Complementary Modules
  - UG/Post Execute
  - UG/Post Builder 1
  - **NX/Post Configurator** 2

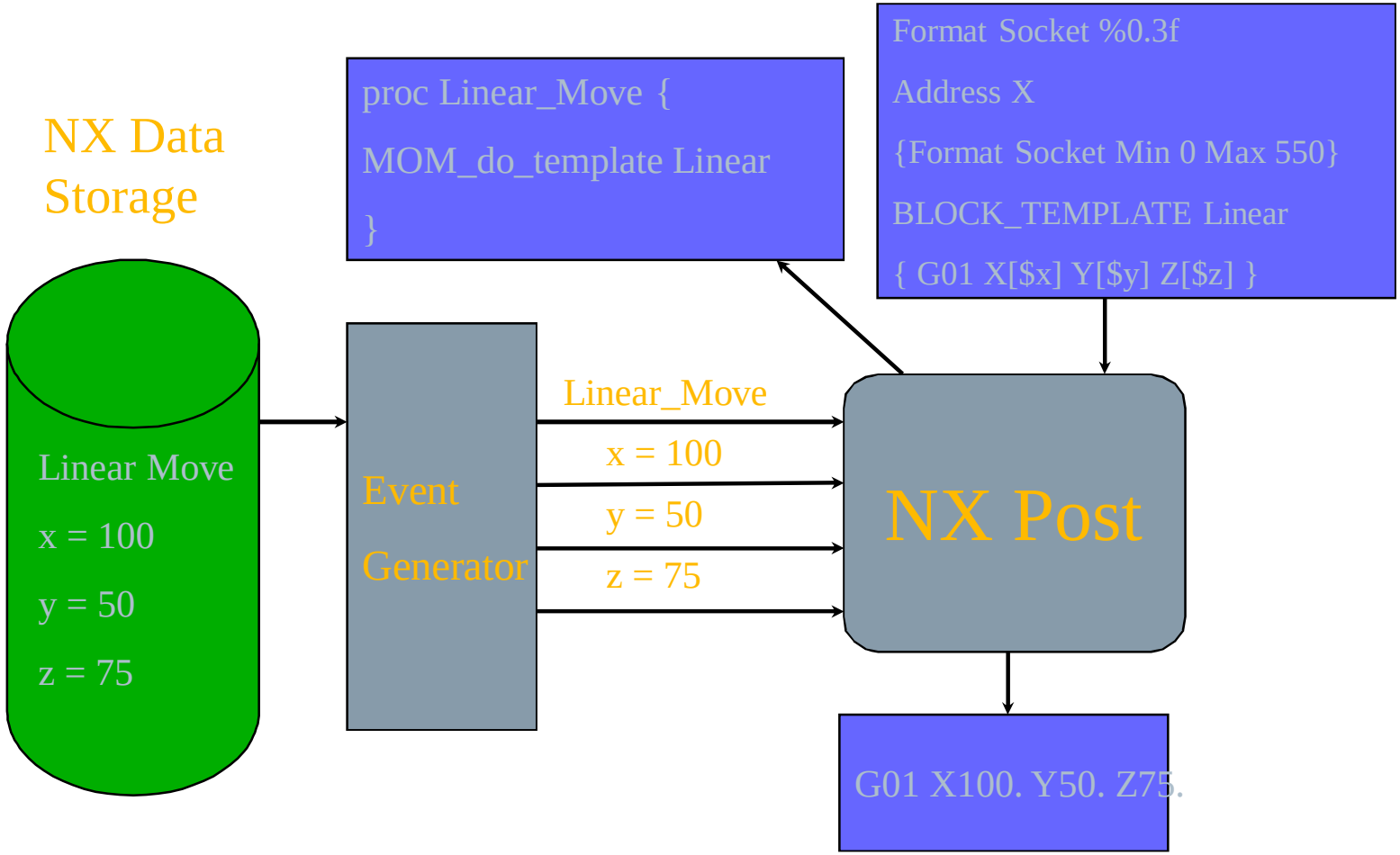


Tcl Advanced

Post Data Flow



Post Data Flow



The MOM object

SIEMENS
Ingenuity for life

```
mom_obj = NXPost( definition_file, event_handler_file, output_file )
```


- Creates a new Tcl interpreter in background and execute the Tcl code
- Creates a Definition parser for definition elements (Words, Formats, Block Templates)
- Creates a new output file with content from the Tcl coding

The created MOM object contains predefined Tcl commands, e.g.:

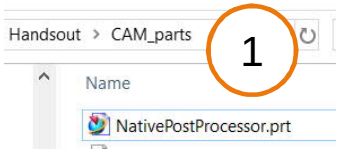
- MOM_output_literal ….
- MOM_output_to_listing_device ….

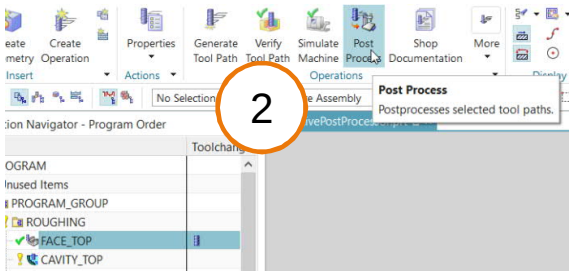
And variables, e.g.:

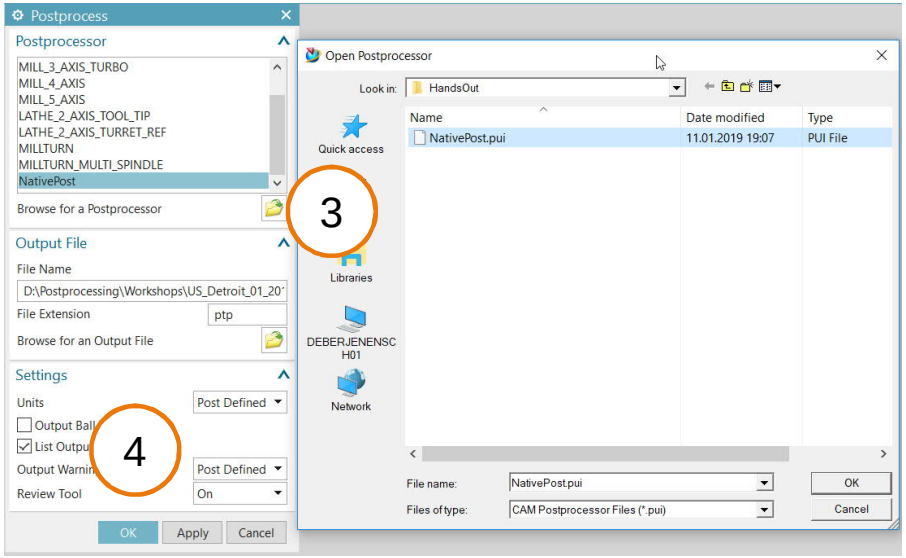
- mom_tool_name
- mom_pos(0)

Based on the used Tcl version in NX this interpreter is extended with multiple extensions, called MOM commands. These commands are not available in a native Tcl interpreter. It‘s also possible to extend the Tcl interpreter in NX with additional commands with C++.

Exercise 1 – MOM Tcl



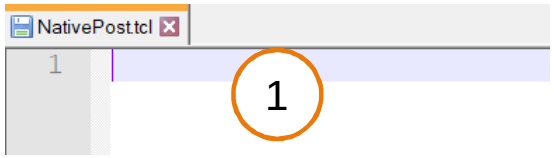


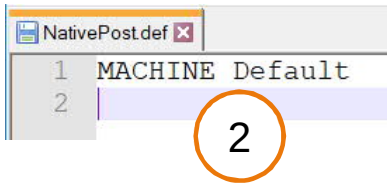


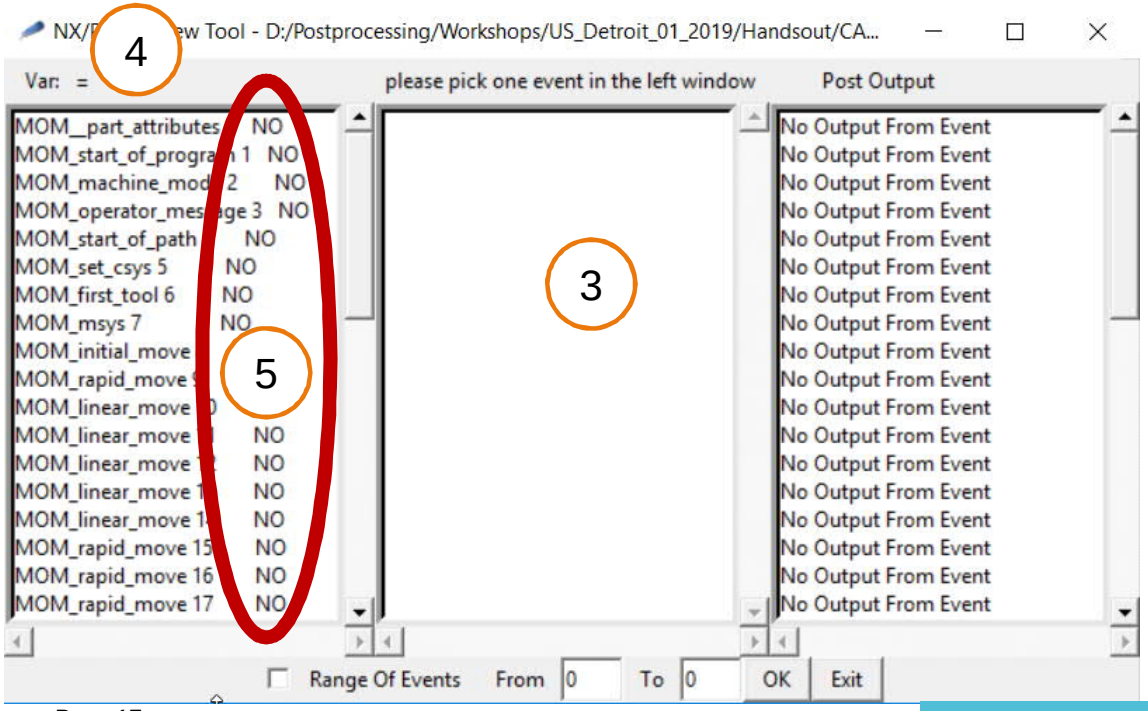
1. Open **NativePostprocessor.prt** from \CAM_parts
2. Select FACE_TOP operation and click Post Process
3. Browse for post processor and select the **NativePost.pui** from \001_Snippet\HandsOut\
4. Switch on the Review Tool

The .pui-file was initally created for Post Builder and contains the Post Builder information. The dialog **not expect any information** in that file, in background it search for the filename with .tcl and .def extension and load the files.

Exercise 1 – MOM Tcl

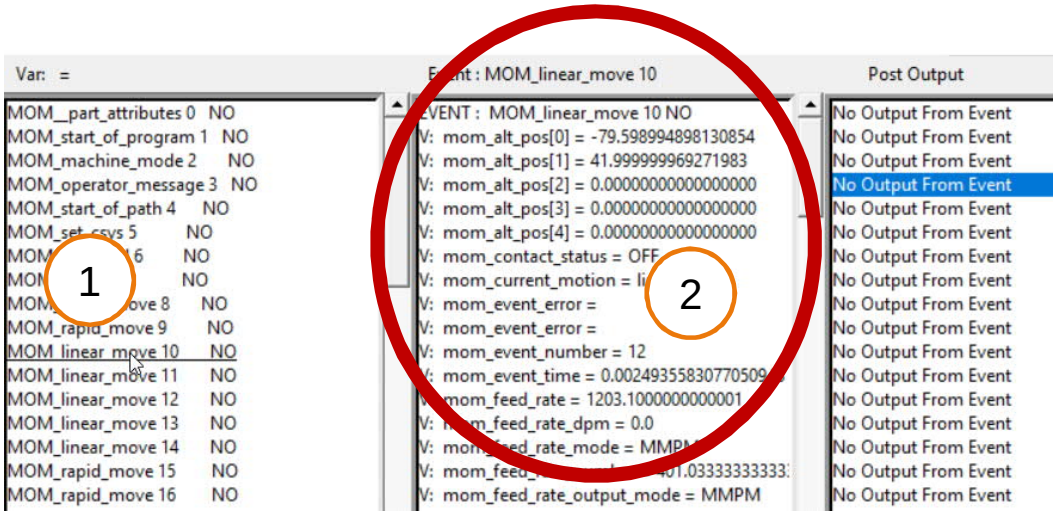






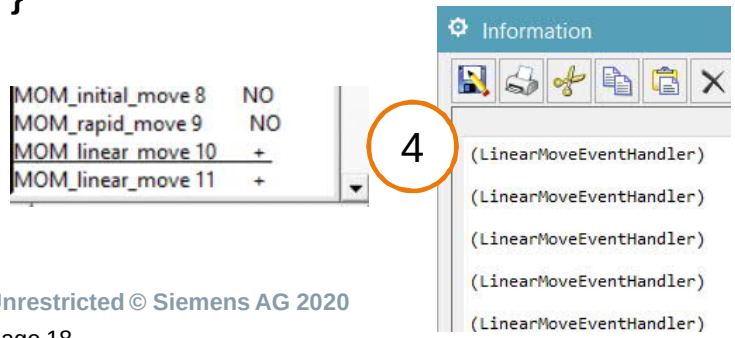
1. Tcl-file is empty
2. Def-file contains mininum the MACHINE default line
3. Run the post processor and Review Tool will open
4. Generated MOM Events from Tool Path
5. **NO** indicates that **no Tcl eventhandler** exists for this MOM Event

Exercise 1 – MOM Tcl




```
proc MOM_linear_move {} {
MOM_output_literal "(LinearMoveEventHandler)"
}
```

3



1. Select a MOM event, e.g. MOM_linear_move
2. The available mom variables will be listed which are created from the Tool Path
3. Extend the NativePost.tcl with an Eventhandler for the MOM_linear_move
4. Run post process again with Review Tool, notice that the Event indicates the Tcl eventhandler and in listing window appears the output

Not all MOM Events are shown in the Review Tool. Most of MOM events can be found in the help collection: https://docs.plm.automation.siemens.com/tdoc/nx/12.0.2/nx_help#uid:xid1128418:index_xid917284:id1319759:xid1398609

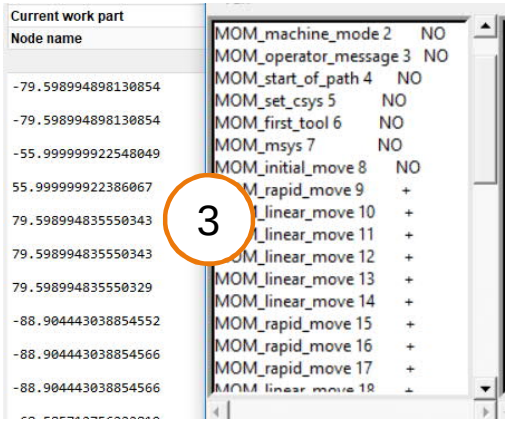
Exercise 1 – MOM Tcl

SIEMENS
Ingenuity for life

1

```
proc MOM_linear_move {} {
MOM_output_literal "$::mom_pos(0)"
}

proc MOM_rapid_move {} {
MOM_output_literal "$::mom_pos(0)"
}
```

2



3

4

```
proc MOM_before_motion {} {
MOM_output_text "----->Before Motion"
}
```

1. Add MOM_rapid_move eventhandler in NativePost.tcl
2. Output the mom_pos(0) value for MOM_linear_move and MOM_rapid_move
3. Run post process to validate the commands
4. Add MOM_before_motion eventhandler

Output directly the values of variables happens unformatted. With native Tcl command „format“ this can be changed or using integrated functionality of MOM core.

Some MOM Events are not visible in the Review Tool, e.g. the MOM_before_motion.

Exercise 1 – MOM Def

SIEMENS
Ingenuity for life

```
MACHINE Default
FORMATTING
{
 FORMAT AbsCoord "&__5.3_"
 ADDRESS X
 {
    FORMAT      AbsCoord
    FORCE       off
    MAX         99999.999 Abort
    MIN        -99999.999 Abort
    LEADER      "X"
    TRAILER     ""
 }
 BLOCK_TEMPLATE linear_move
 {
   X[$mom_pos(0)]
 }
}
```

1

2

3

1. Open NativePost.def and add Format Abscoord
2. Add Address X
3. Add Blocktemplate linear_move

The .def file contains information for:

- Formats
- Addresses
- Block Templates
- Sequence

It‘s based on a LexYacc parser and can be extended easily. A .def-file is sourced during the selection of the post processor. A .def-file must contain „**MACHINE xxx**“ and all definitions must be between the **FORMATTING{xxx}**

Exercise 1 – MOM Def/ Tcl

SIEMENS
Ingenuity for life

1

```
FORMAT String "%s"

ADDRESS Text
{
    FORMAT      String
    FORCE       always
    LEADER      ""
}
```

3

```
BLOCK_TEMPLATE linear_move
{
   Text[G1]
   X[$mom_pos(0)]
}
```

2

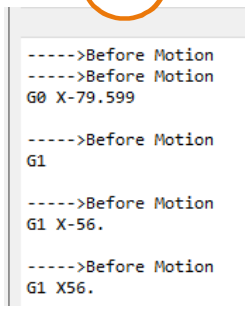
3

```
BLOCK_TEMPLATE rapid_move
{
   Text[G0]
   X[$mom_pos(0)]
}
```

```
proc MOM_linear_move {} {
#MOM_output_literal "$::mom_pos(0)"
MOM_do_template linear_move
}
proc MOM_rapid_move {} {
#MOM_output_literal "$::mom_pos(0)"
MOM_do_template rapid_move
}
```

4

5



1. Add Address **Text** and Format **String**
2. Add a Block Template for rapid_move
3. Extend linear and rapid move with a Text element, e.g. G0 and G1
4. Call the Block Template in NativePost.tcl for linear and rapid moves
5. Run post process to validate result, values are formatted due the definitions in the .def file

NX Tcl commands 1/2

SIEMENS
Ingenuity for life

MOM_output_literal „text“

- Output text automtically with sequence number, can be used for information to the user or for debugging purpose

MOM_output_text „text“

- Output text without sequence number

MOM_do_template [Block template name]

- Output a defined Block Template from the definition file

MOM_set_seq_on

- Switch on the sequence numbers

MOM_set_seq_off

- Switch off the sequence numbers

Online Documentation:

https://docs.plm.automation.siemens.com/tdoc/nx/12.0.2/nx_help#uid:xid1128418:index_xid917284:id1319759:xid916798

NX Tcl commands 2/2

SIEMENS
Ingenuity for life

MOM_enable_address [Address1] [Address2]

- Enable an address from the definition file

MOM_disable_address [Address1] [Address2]

- Disable an address from the definition file

MOM_suppress <Always|Once|Off> [Address1] [Address2]

- Suppress the output of an adress once or always in next Block Template

MOM_force <Always|Once|Off> [Address1] [Address2]

- Force the output of an adress once or always in next Block Template

MOM_output_to_listing_device [Text]

- Very useful for debugging purposes, will generate an output in the beginning of the listing window, no output in the output file

Online Documentation:

https://docs.plm.automation.siemens.com/tdoc/nx/12.0.2/nx_help#uid:xid1128418:index_xid917284:id1319759:xid916798

Useful Tcl snippet

```
proc name {optional arg} {
#code
}
```

•Procedure body, arguments are optional. To call a procedure from use the name in calling proc, e.g. name

```
if {$var == $value} {
#do something
}
```

•If commands can be used for easy matching of conditions (==,!=,<,>), take care of comparing values with expr command

```
switch $var {
„Value1“ {#do something}
„Value2“ {#do something}
default {#do default}
}
```

•Switch commands allows multiple conditions matching, e.g. ask the mom_motion_type

```
foreach value {#list} {
   # do something
}
```

•Cycle through a list of elements, very useful e.g. to collect data during a post process run and output it at the end

Online Documentation:

https://wiki.tcl-lang.org/

For native Tcl programming and commands the Tcl Wiki is a good starting point. The site includes tutorials and descriptions of all available Tcl commands.

Tcl basics

```
set myvar „text“
set myvar2 $myvar
```

- A local or global variable can be set with a value or the value of a different variable
- To read the value of a variable use the $

```
global myvar
MOM_output_literal „$myvar“
Is equal to
MOM_output_literal „$::myvar“
```

- The started Tcl interpreter has a global namespace
- It‘s possible to have multiple namespace (additional extensions)
- in NX Post normally one global namespace is used
- To get a variable from a different procedure this variable must be declared as global
- The :: is a shorten form of writing

```
lappend ::mytoollist $::mom_tool_name
…
foreach toolname $::mytoollist {
MOM_output_literal "--->$toolname"
}
```

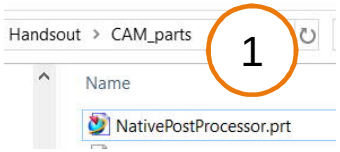
- Append the tool name to a list, e.g. in MOM_end_of_program output all elements of the list

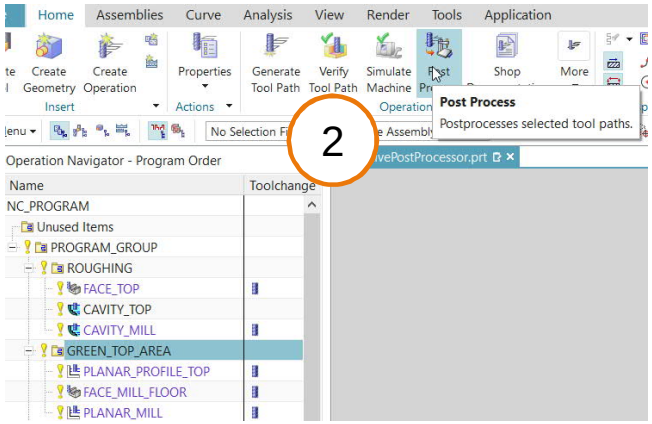
Error handling:

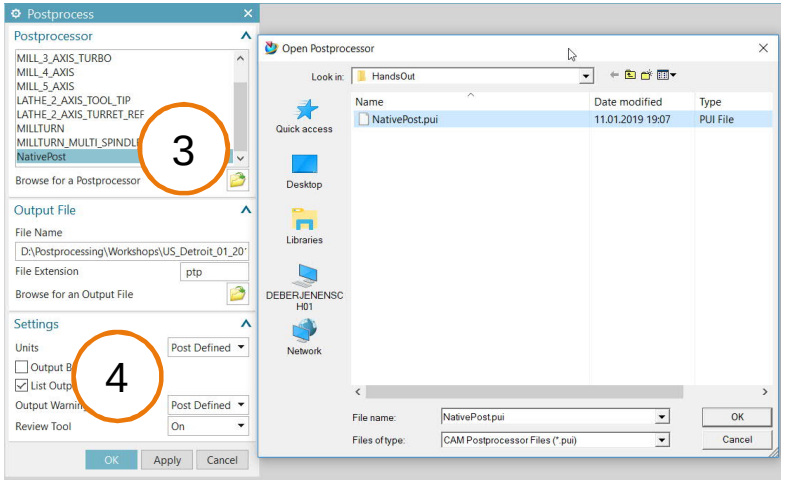
For debugging and avoid crashes of the Tcl interpreter the **info exists** can be used to avoid such situations e.g.:

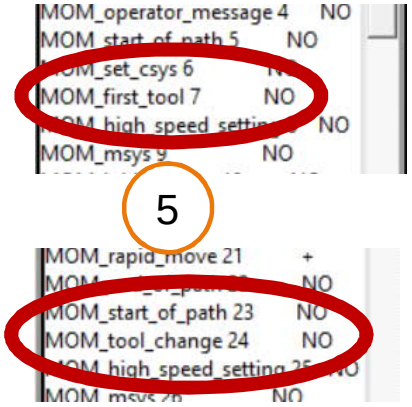
```
If {[info exists ::mytoollist]} {
#do something
}
```

Exercise 2 – MOM Tcl Toolname list









1. Open **NativePostprocessor.prt** from \CAM_parts
2. Select GREEN_TOP_AREA program group and click Post Process
3. Browse for post processor and select the **NativePost.pui** from \001_Snippet\HandsOut\
4. Switch on the Review Tool
5. Run the post process and notice the MOM tool change events

In MOM two events are generated for the Toolchange. The first toolchange in a postprocessor run is named **MOM_first_tool**, for additional tool changes the **MOM_tool_change** event is generated.

Exercise 2 – MOM Tcl Toolname list

SIEMENS
Ingenuity for life

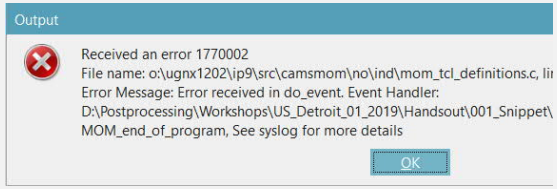
```
proc MOM_first_tool {} {
lappend ::mytoollist $::mom_tool_name
}

proc MOM_tool_change {} {
lappend ::mytoollist $::mom_tool_name
}

proc MOM_end_of_program {} {
if {[info exists ::mytoollist]} {
        foreach toolname $::mytoollist {
                MOM_output_literal "--->$toolname"
        }
}
}
```

2
3
2
3
4
5
6

```
--->UGT0201_088
--->UGT0201_015
--->UGT0201_092
```

1. Open **NativePost.tcl** in editor
2. Add MOM_first_tool and MOM_tool_change events
3. Append the provided NX Post value for the tool name to a list
4. Add MOM_end_of_program event
5. Cycle thru the list and output all the tool names
6. Run post process and validate result

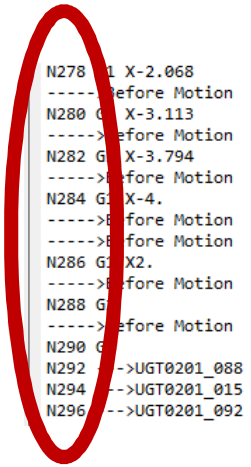
Using of „info exists“ avoid crashes if the tool list variable not exists.



Final exercise

SIEMENS
Ingenuity for life

- Add the sequence number from MOM
- Create the sequence command
- Create a sequence address
- Create a Block template for sequence numbering



Hint:

- Take a look into existing Post Builder or Post Configurator post processor to find the correct implementation
- The sequence is a standard implementation in MOM, it‘s not needed to add any Tcl

SIEMENS
Ingenuity for life

Agenda:

Helpful documentation and tools for the class
NX MOM architecture
Advanced Tcl commands
Excercises

Array

SIEMENS
Ingenuity for life

Arrays:

- set tool_axis(0) x
- set tool_axis(1) y
- set tool_axis(2) z

Associative arrays:

- set capital(USA) “Washington D.C.”
- set capital(GERMANY) “Berlin”
- array names capital -> USA, GERMANY

list

SIEMENS
Ingenuity for life

- The list is the basic Tcl data structure
- A list is simply an ordered collection of stuff
- numbers, words, strings, or other lists
- Even commands in Tcl are just lists in which the first list entry is the name of a proc, and subsequent members of the list are the arguments to the proc

```
set x "a b c"
puts "Item at index 2 of the list {$x} is: [lindex $x 2]\n"
set y [split 7/4/1776 "/"]
puts "We celebrate on the [lindex $y 1]'th day of the [lindex $y 0]'th month\n"
set z [list puts "arg 2 is $y" ]
puts "A command resembles: $z\n"
set i 0
foreach j $x {
    puts "$j is item number $i in list x"
    incr i
}
```

Tcl internals

SIEMENS
Ingenuity for life

info commands	**(shows Tcl commands)**
info commands {a*}	**(shows Tcl commands with a*)**
info tclversion	**(Tcl version)**
info exists *varname*	**(check exists varname)**

Unset variable

SIEMENS
Ingenuity for life

unset *?-nocomplain? ?--? ?name name name ...?*

- This command removes one or more variables
- Helpful if using a UDE in operations which affect may following operations
- Each name is a variable name, specified in any of the ways acceptable to the set command
- If a name refers to an element of an array then that element is removed without affecting the rest of the array
- If a name consists of an array name with no parenthesized index, then the entire array is deleted
- If -nocomplain is specified as the first argument, any possible errors are suppressed
- The unset command returns an empty string as result.

```
unset myvar_ude
}
```

Namespace

SIEMENS
Ingenuity for life

- The namespace command lets you create, access, and destroy separate contexts for commands and variables
- Can be useful to separate functionality
- A namespace must be created before use of namespace variables or commands
- When create a new procedure for a namespace the namespace name and :: is necessary (SPLM::xxxxxx)

```
namespace eval SPLM {

}
```

```
proc SPLM::OutputMotion {} {
    MOM_output_literal "XYZ"
}
```

SIEMENS
Ingenuity for life

Agenda:

Helpful documentation and tools for the class
NX MOM architecture
Advanced Tcl commands
Excercises

Excercise I

SIEMENS
Ingenuity for life

Output subprogram with zero offsets at the end of program with label calls.

Conditions:

- Identify the MOM event for coordinate system
- Collect information in a list
- Control the behavior with a variable to switch on/ off
- If off standard output of the zero offset, if on output the Label call
- Output the elements in the end of program event

SIEMENS
Ingenuity for life

Thomas Jenensch

Product Portfolio Lead NX CAM Infrastructure
Siemens Industry Software

thomas.jenensch@siemens.com

Thank you.