Traceability Comments. (11/30/2007)

Julio Garrido Campos (jgarri@uvigo.es)
Vigo University

Index

1. Traceability Conclusions and Comments from last teleconference.

a. Need for more Administrative data collection capabilities (for NC_planning processing) ?.

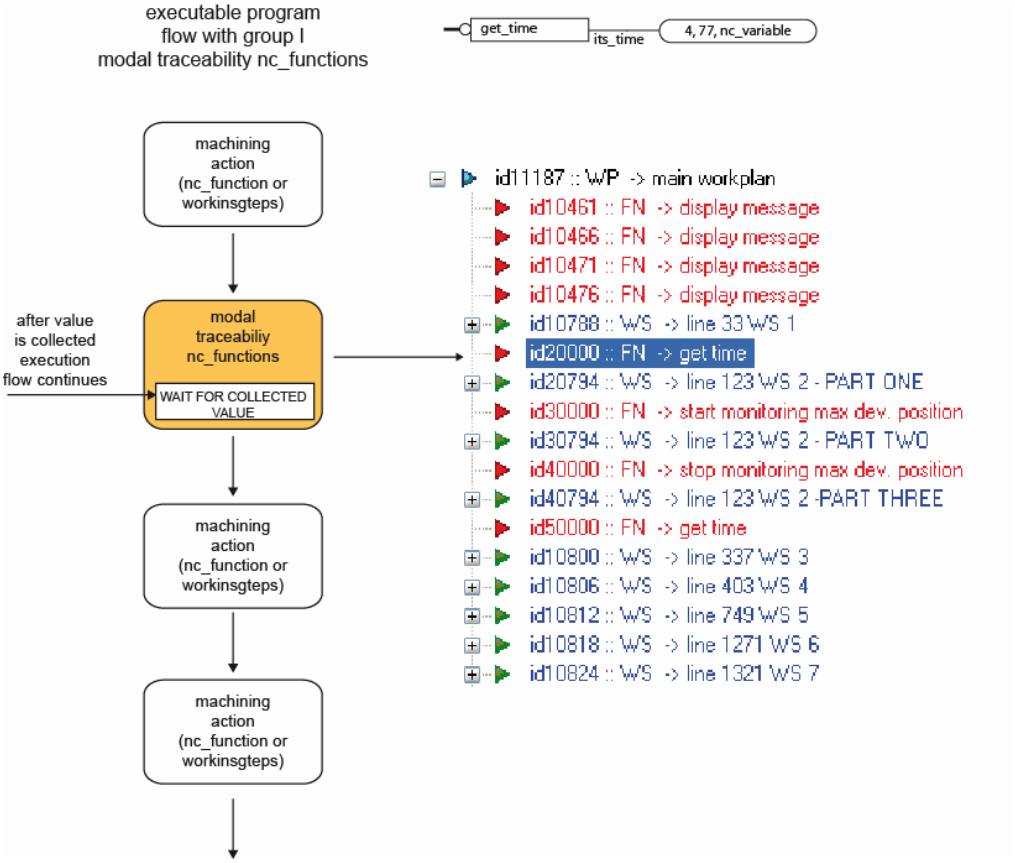
b. Other machinning Data for GROUP II nc_functions (torque, current,...)

c. Need to be aware of manual operator actions like “overrides” ?.

d. Need for a WS time “maximum” or “expected” execution placeholder* in a similar way as there is maximum toolpath deviation value placeholder in AP-238?

2. Detailed Explanation of Group I/II/III traceability nc_functions.

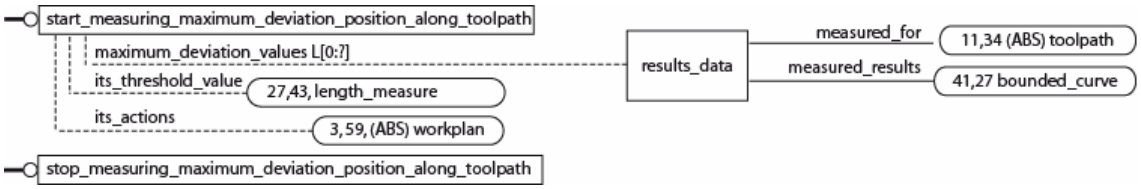
a. **Group I** nc_functions: Just insert them in the executable structure, get a SINGLE value, and "update" an AP-238 nc_variable (or log the value and the workingstep to a file).



b. **Group II/III** nc_functions: ARM model includes OPTIONAL Attributes, and depending on the presence or not of this attributes, the function is a group II or group III function.

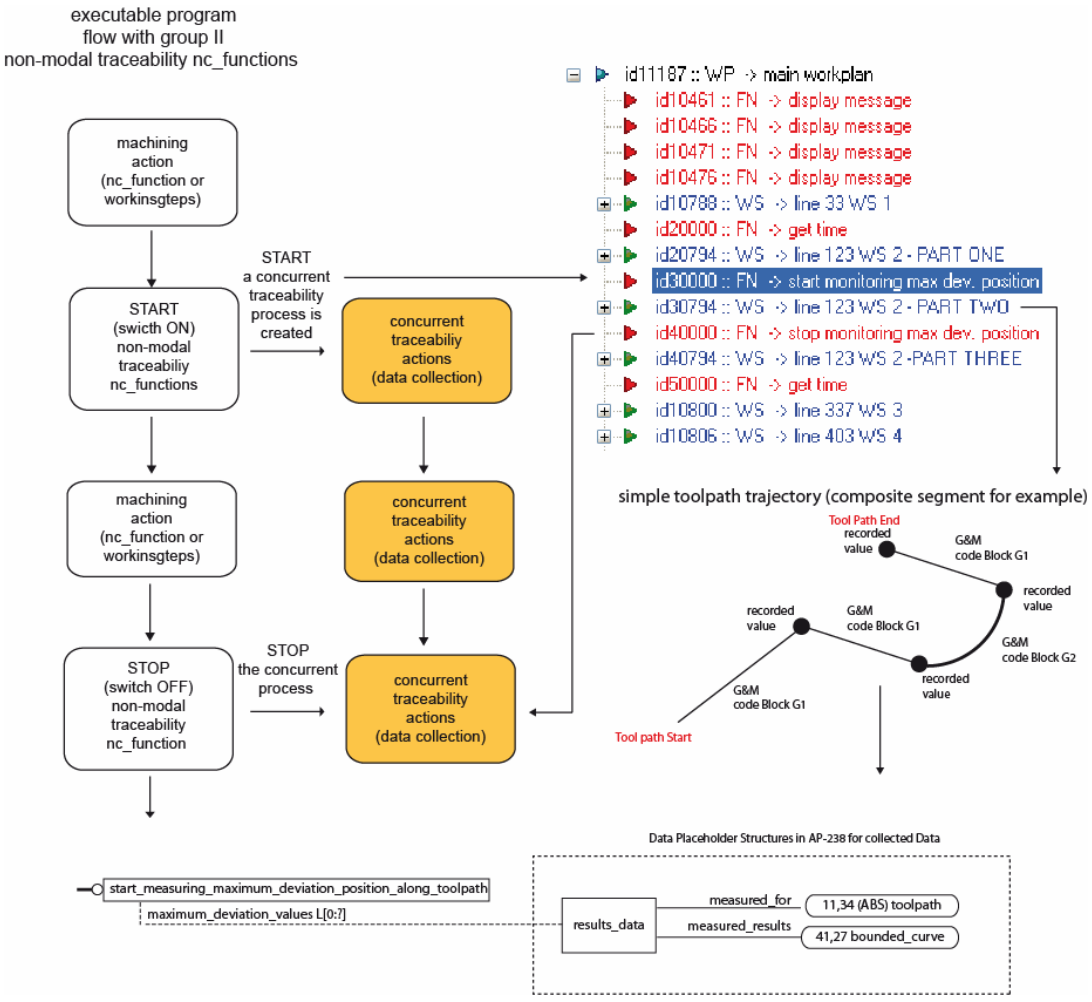
The main difference between GROUP II and GROUP III functions, is that GROUP II functions are thought to COLLECT/LOG data, while GROUP III extend GROUP II functionality to test a specified "condition" and perform some actions depending on the tested condition (and data storage/loging is also optional).

ARM model example for **"get_maximum_deviation_position_along_toolpath"**:

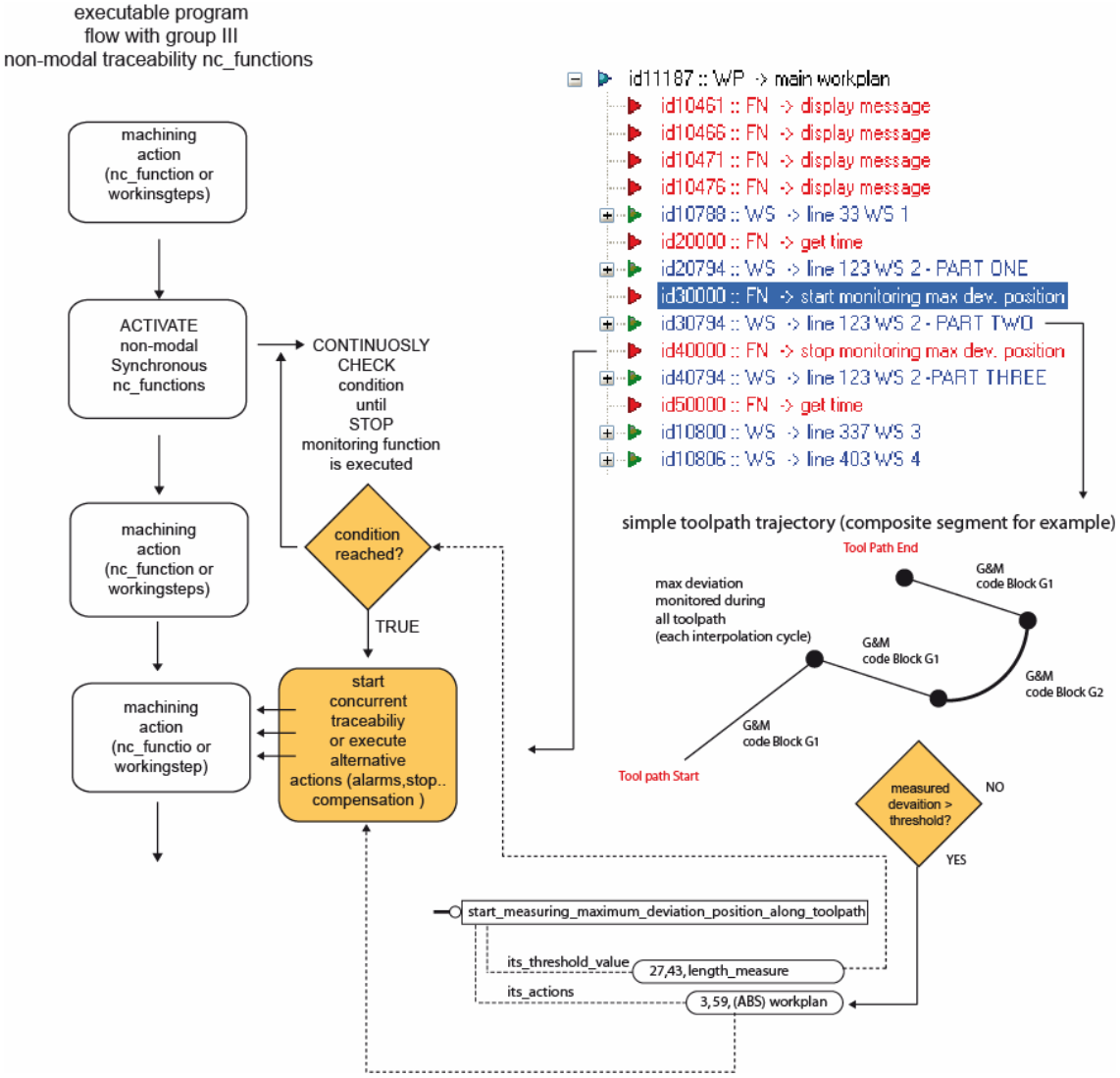


This is the generic model, where:
maximum_deviation_values L[0:?] is used to store in AP-238 a bounded curve, series of collected values (per toolpath and following the same parametrization as the corresponding workingstep toolpath).
its_threshold_value is used **only if** nc_function acts as a group III function to specify a threshold value for the comparing/triggering condition.
its_actions is an alternative workplan (a series of actions) to be done in case the specified condition is fulfilled (just for group III).

<u>Example for GROUP II function use…</u>



Example for GROUP III function use…



3. Proposed Actions for telecom comments …

Comment a: Administrative Data (through GROUP I functions):

With Group I function for accessing punctual values, the **"get_sensor" data, if allowed to access internal CNC registers**, could be used to gather a information, (at least from 840d CNC variables) like:

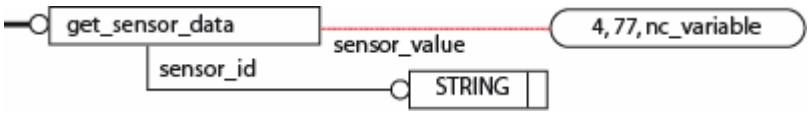
Machine NCK version	($AN_NCK_VERSION)
Program Name	($P_PROG[0])
Machine Runtime	($AC_OPERATING_TIME)
Program Runtime	($AC_CYCLE_TIME)
Operation Time	($AC_CUTTING_TIME)
Tool Data, Tool/Axis Offsets	……

There can be done using for example the **"get_sensor_data"** nc_function, but to log these values the following alternatives could be proposed:

1. Making the sensor_value optional, so it could be specified that if not present, the value is written into a log file, but will not be into the AP-238 data.

2. Why ?. Because for some of this values, are STRINGS, TIME VALUES and it will be necessary to enhance the AP-238 concept of nc_variable.

3. Also there is work to do about the specification for the "sensor_id" attribute.



Comment b:
Other machining Data for GROUP II nc_functions:

There are other interesting data that even it can be accessed using the Group II **"star_monitoring_ _sensor_data", if used also to access internal CNC registers**, like:

% Drive LOAD	($AA_LOAD[X]).
TORQUE	($AA_TORQUE[X]),
POWER	($AA_POWER[X]),
CURRENT	($AA_CURR[X]) …

However, one question for these, and maybe other similar data, as they are measured by AXIS, is if logging should include all axis by defect, or it should be explored the possibility to specify which axis to monitor ?

Comment c:
Need to be aware of operator actions like "overrides"?. Synchronous traceability data collection" vs. "traceability of CNC asynchronous events

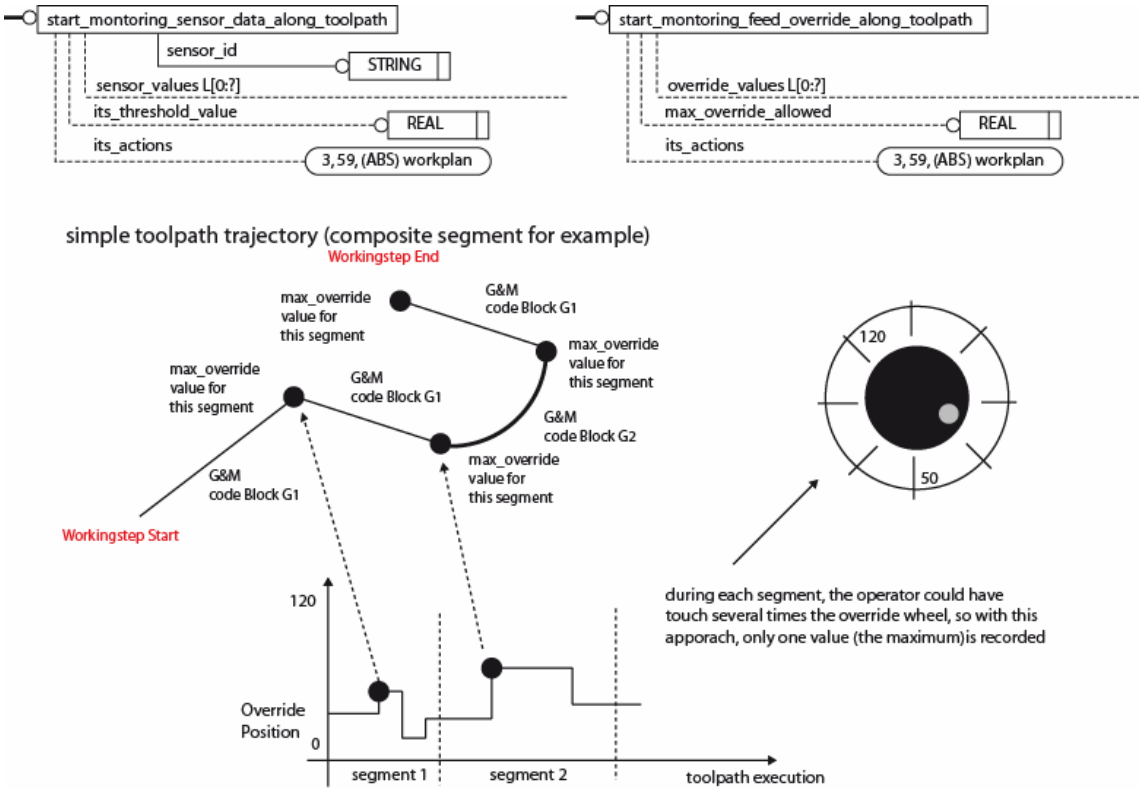
From comments, it seems interesting to detect and record **Operator actions** like feed/spindle overrides. These are completely asynchronous with program execution, so during a workingstep or tool path execution, the operator could have "played" with the override hand wheels. To record this, there is an internal CNC register (in 840D) that control and calculates the applied override before each interpolation cycle.

So changing the selected "sensor", the "**start_monitoring_sensor_along_toolpath"** concept could be applied to have a ONE value per toolpath segment/ G&M code Block monitoring of the operator actions on the overrides handwheels.

```
AVALIABLE 840d Varaibles:
   $AC_OVR (active path/federate override)
   $AC_DRF  (axial  override  value  caused  by  the
handwheell)
   (other values for total override, $AA_TOTAL_OVR[ax] =
   PLC_OVR*NC_OVR, could also be accessed if considered
   as better alternatives …)
```

The idea could be as follows:

Monitor any of these variables with a group-II/III like nc_functions (or using the already proposed "**start_monitoring_sensor_along_toolpath**"), during G-code block execution, log the maximum values (SINGLE VALUE LOG APPROACH)

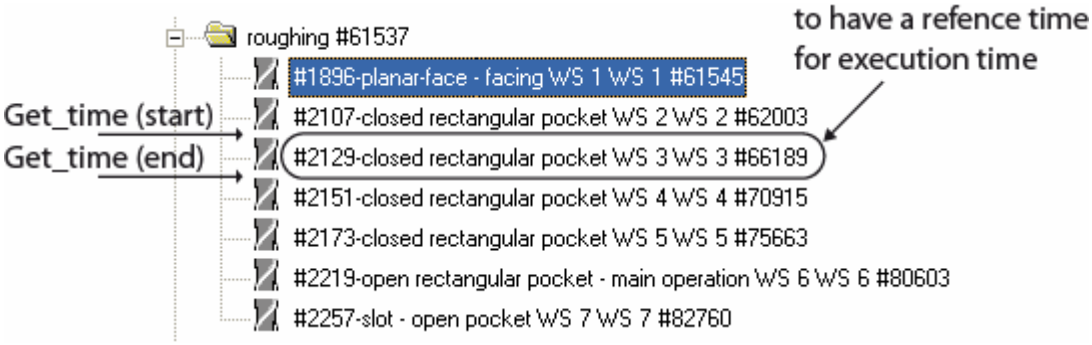


Another option, depending on the amount of data that could be allocated in the NC-Controller, could be to use a data structure (for example a FIFO ($AC_FIFO on 840d) to log several override values per segment, allowing to record an N_NUMBER (the bigger ones) of operator override values.

Also from this asynchronous event perspective, it seems interesting, logging other kind of events, as CNC mode changes, so monitoring the $AC_PROG variable could be important to detect if the program has been reset(0), stopped(1), is active (2), waiting (3) or interrupted (4). More to explore on this are operations performed on MANUAL mode, and how to log them

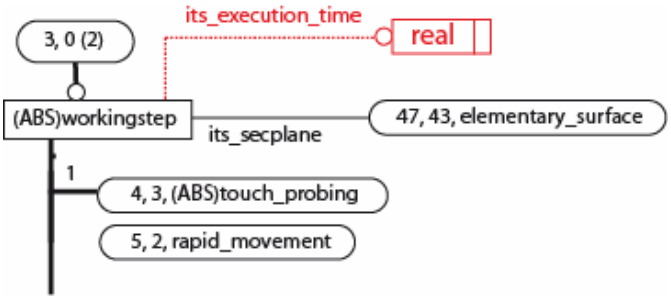
Comment d:
The get_time functions …

The objective could be to have a reference value to compare it with the time as collected by two get_time nc_functions and take the necessary actions …



Options for setting this maximum execution time for the workingstep, similar as the maximum_path_deviation present in AP-238 tool path, to compare collected values. This could be solved in two ways:

1. By adding a execution_time value (expected maximum time) as and optional attribute to the AP-238 workingstep entity …



2. But also with a group III nc_function,

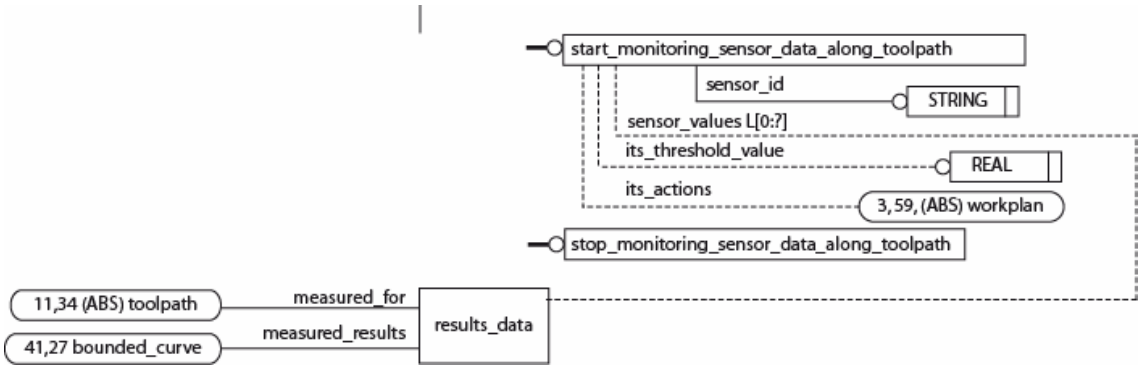
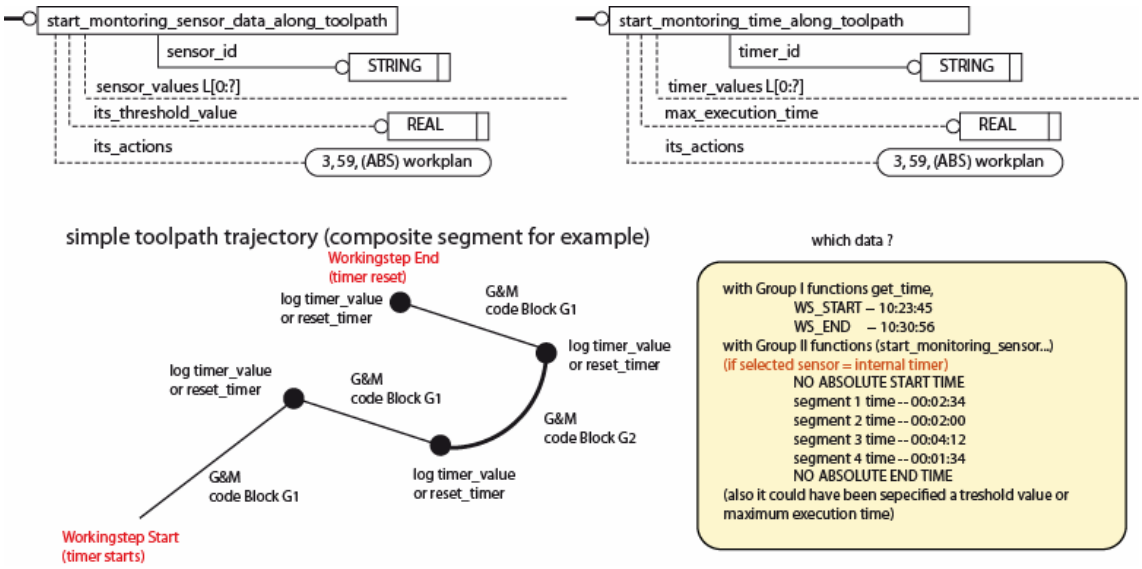


figure 3

1. **SENSOR** → if sensor, could also refer to an internal CNC register/value, a TIMER could be used. Then a programmed "**start_monitoring_sensor_along_toolpath"** with **sensor_id** → "time", and **its_threshold value**→ "the maximum time", could do the work and optional, log/or not detailed time execution for the workingstep (richer than start-end time, as show in next figure).
2. However, a more appropriate function could be **"start_monitoring_time_execution_along_toolpath" (right EXPRESS-G model)**



G&M implementation approaches for the proposed case could depend on: if the threshold (maximum execution time) refers to the TOTAL tool path execution time or for each segments or for the total workingstep

Pseudocode Threshold (valid as maximum time per segment, logging incremental times for segment values)

```
N100  $AC_TIMER[1] = 0 …
;RESET R[] variables
;R[1] WILL Hold the threshold value
..........
;START SINCHRONYZED ACTION threshold set for each segment
ID=1 WHEN $R[1] < $AC_TIMER[1] DO (ACTION: stop, alararm .. log data)
G1….
WRITE(“ERROR”,”LOGFILE”,”SEGMENT 1 TIME: “ << $AC_TIMER[1]);
$AC_TIMER[1]= 0 ;
G1…
WRITE(“ERROR”,”LOGFILE”,”SEGMENT 1 TIME: “ << $AC_TIMER[1]);
$AC_TIMER[1]= 0 ;
G1…
……..
CANCEL(1)
```

Pseudocode Threshold (valid as maximum time per workingstep/toolpath, and logging absolute times for

segment values: note in this case timer is not reset after each G block)

```
N100  $AC_TIMER[1] = 0 …
;RESET R[] variables
;R[1] WILL Hold the threshold value
..........
;START SINCHRONYZED ACTION threshold set for each segment
ID=1 WHEN $R[1] < $AC_TIMER[1] DO (ACTION: stop, alararm .. log data)
G1…..
WRITE(“ERROR”,”LOGFILE”,”SEGMENT 1 TIME: “ << $AC_TIMER[1]);
G1…
WRITE(“ERROR”,”LOGFILE”,”SEGMENT 1 TIME: “ << $AC_TIMER[1]);
G1…
……..
CANCEL(1)
```