Universidad de Vigo
Julio Garrido Campos (jgarri@uvigo.es)

GROUP I FUNCTIONS.

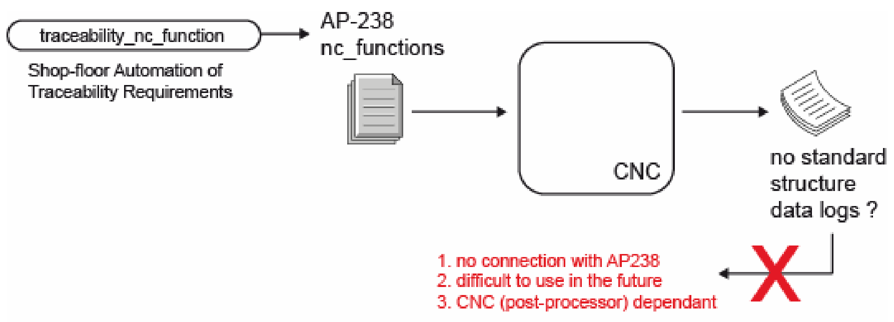
DATA STRUCTURES & GLOBAL/DEFAULT BEHAVIOUR

Main scope should be standard automation of CNC traceability processes and data, independently of CNC architecture…

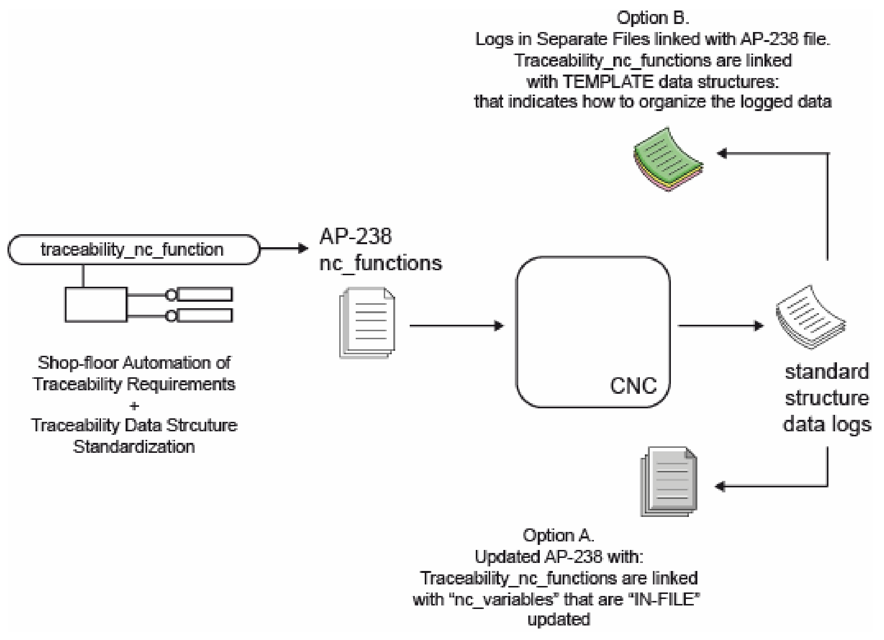
1. Options for logging data:

Three major options for logging data:

1. NC-Functions simply "standardize" the traceability automation → indicate traceability requirements, so log files are non standard. **These seems not to be adequate...**


traceability_nc_function
Shop-floor Automation of Traceability Requirements
AP-238
nc_functions
CNC
no standard structure data logs ?
1. no connection with AP238
2. difficult to use in the future
3. CNC (post-processor) dependant

2. NC-Functions standardize the automation process (requirements) and also the data format (structure) of traceability logs. With this approach there are two alternatives.


Option B.
Logs in Separate Files linked with AP-238 file.
Traceability_nc_functions are linked
with TEMPLATE data structures:
that indicates how to organize the logged data
traceability_nc_function
Shop-floor Automation of Traceability Requirements
+
Traceability Data Strcuture Standardization
AP-238
nc_functions
CNC
standard structure data logs
Option A.
Updated AP-238 with:
Traceability_nc_functions are linked
with "nc_variables" that are "IN-FILE"
updated

a. Traceability data is logged in the AP-238 files.

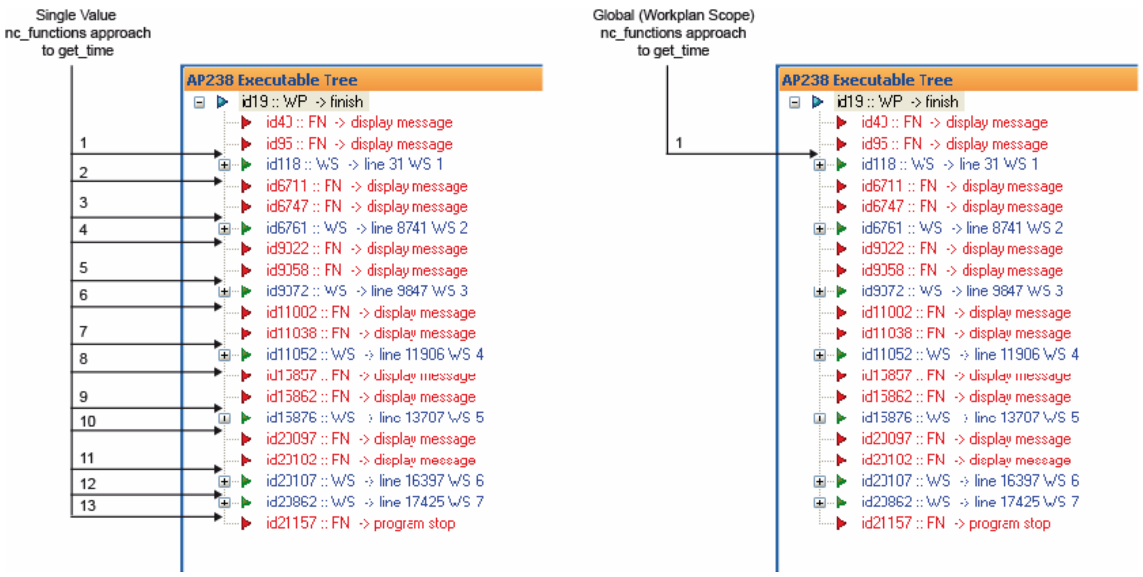
b. Traceability data is logged in separate files but following the STEP data structures indicated by the nc_functions data models.

2. On-Line data Recording vs Global or Default Behaviour.

There are also two major options for **group I** traceability nc_functions functionality. Both have benefits depending on what is wanted to do.

1. If defined as single value functions: CNC executes the function, logs a value and continues execution
2. If defined as global/default process: CNC executes the function and logs values for the specified requirements during all the machining process.

See next figure: alternatives can do the same and implementation details for both can be similar, it is just a “matter” of which approach will be preferred and more useful.

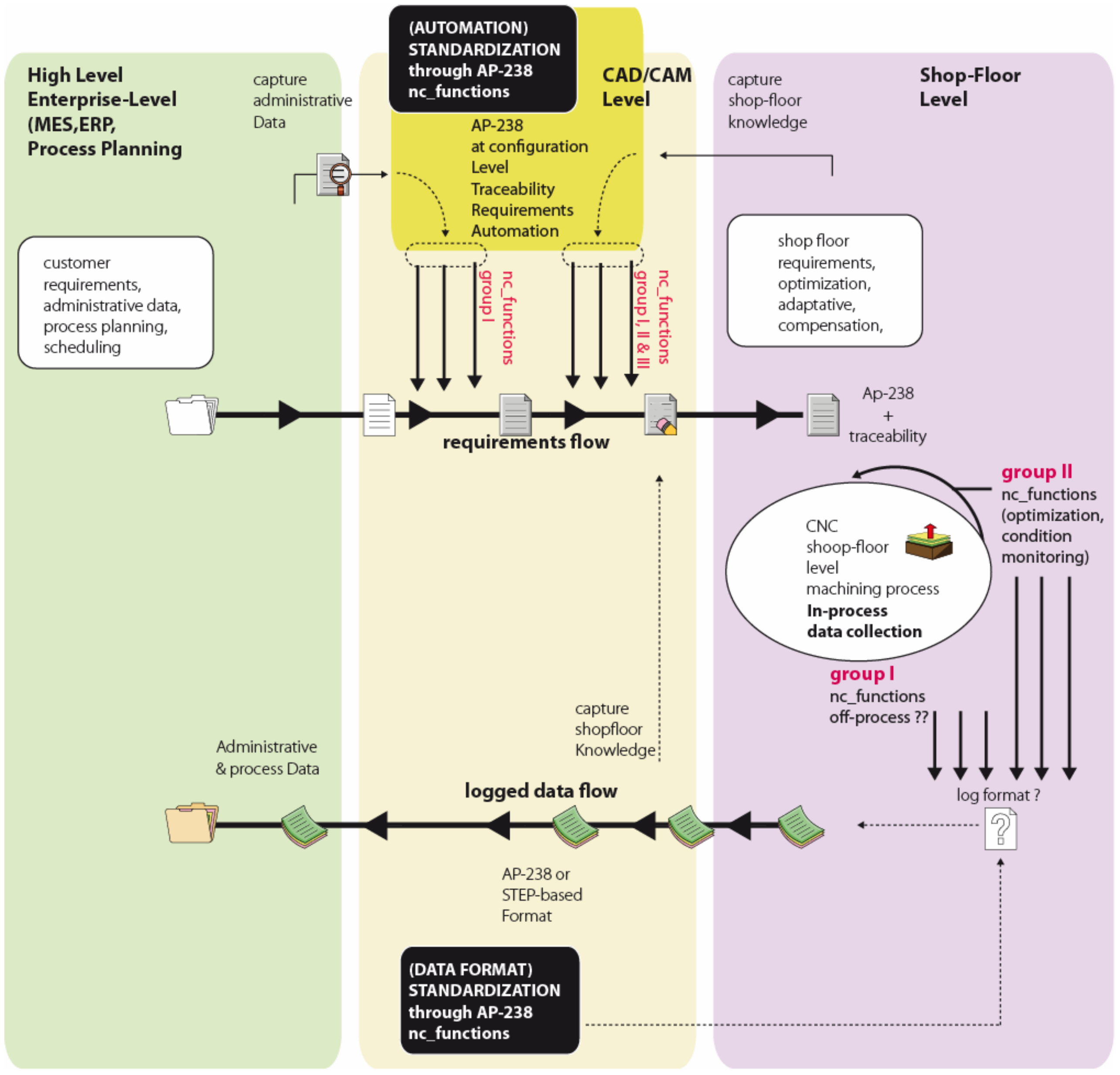

Single Value
nc_functions approach
to get_time
AP238 Executable Tree
id19 :: WP -> finish
id40 :: FN -> display message
id95 :: FN -> display message
id118 :: WS -> line 31 WS 1
id6711 :: FN -> display message
id6747 :: FN -> display message
id6761 :: WS -> line 8741 WS 2
id9022 :: FN -> display message
id9058 :: FN -> display message
id9072 :: WS -> line 9847 WS 3
id11002 :: FN -> display message
id11038 :: FN -> display message
id11052 :: WS -> line 11906 WS 4
id13857 :: FN -> display message
id13862 :: FN -> display message
id13876 :: WS -> line 13707 WS 5
id20097 :: FN -> display message
id20102 :: FN -> display message
id20107 :: WS -> line 16397 WS 6
id20862 :: WS -> line 17425 WS 7
id21157 :: FN -> program stop
1
2
3
4
5
6
7
8
9
10
11
12
13
Global (Workplan Scope)
nc_functions approach
to get_time
AP238 Executable Tree
id19 :: WP -> finish
id40 :: FN -> display message
id95 :: FN -> display message
id118 :: WS -> line 31 WS 1
id6711 :: FN -> display message
id6747 :: FN -> display message
id6761 :: WS -> line 8741 WS 2
id9022 :: FN -> display message
id9058 :: FN -> display message
id9072 :: WS -> line 9847 WS 3
id11002 :: FN -> display message
id11038 :: FN -> display message
id11052 :: WS -> line 11906 WS 4
id13857 :: FN -> display message
id13862 :: FN -> display message
id13876 :: WS -> line 13707 WS 5
id20097 :: FN -> display message
id20102 :: FN -> display message
id20107 :: WS -> line 16397 WS 6
id20862 :: WS -> line 17425 WS 7
id21157 :: FN -> program stop
1

Seeing both alternatives, the following consequences can be “extracted”. With single value functions, complexity goes into the AP-238 executable structure, (does this have implications on the machining speed also or this will be CNC architecture dependant?). With Global nc_functions (workplan scope) complexity goes on the “process concurrency” that has to track the workplan progress and log execution times … (it has been already some previous discussion whether for Ap-238 Ed2, concurrency z …

3. Traceability Data Requirements and nc_functions

Some are covered by group II functions.

Ap-238 Edition II Objectives: Implementable Traceability ?


(AUTOMATION) STANDARDIZATION through AP-238 nc_functions
High Level Enterprise-Level (MES,ERP, Process Planning
capture administrative Data
CAD/CAM Level
capture shop-floor knowledge
Shop-Floor Level
AP-238 at configuration Level Traceability Requirements Automation
customer requirements, administrative data, process planning, scheduling
group I nc_functions
group I, II & III nc_functions
shop floor requirements, optimization, adaptative, compensation,
requirements flow
Ap-238 + traceability
group II nc_functions (optimization, condition monitoring)
CNC shoop-floor level machining process In-process data collection
group I nc_functions off-process ??
capture shopfloor Knowledge
Administrative & process Data
logged data flow
log format ?
AP-238 or STEP-based Format
(DATA FORMAT) STANDARDIZATION through AP-238 nc_functions

Possible GROUP I data requisites:

Administrative data such us:

- **Execution times (part and workingstep …)**
- **Machine Efficiency (Total run Time vs. Cutting Time …)**
- **Consumed Power, Current … (per Workingstep, per part)***
- **Machine System Data**
- **Machine Event Monitoring (Alarms, Warnings, Stops, Program Interrupts…)**
- **Operator Data, Tool Data, Material Data …**

Process Knowkedge Data

- **Time at one execution points**
- **NC Data at execution points**
  - **Axis Position**
  - **Sensor Data (NCregister, Feed, Speed, Power, Current …)**

4. STEP by STEP GROUP I functions.

Goup I, nc_function and get a single actual nc_value, this functions interact directly with the nc unit, and get values from it (internal memory, register, coupled sensors. These are not globlally or default defined functions. They present more flexibility to collect punctual data, but they add complexity to the configuration process…

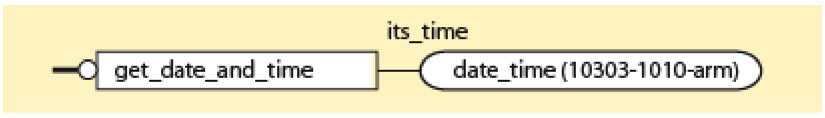
2.1 get_time*

The nc_function name can be changed into **get_time_and_date:** the purpose of this function will be to collect the actual controller date and time value. Also a **get_workingstep_execution_times*** function to collect time interval (workingstep/toolpath execution times) could be added as a traceability group II function or **as a GLOBAL group I function**.

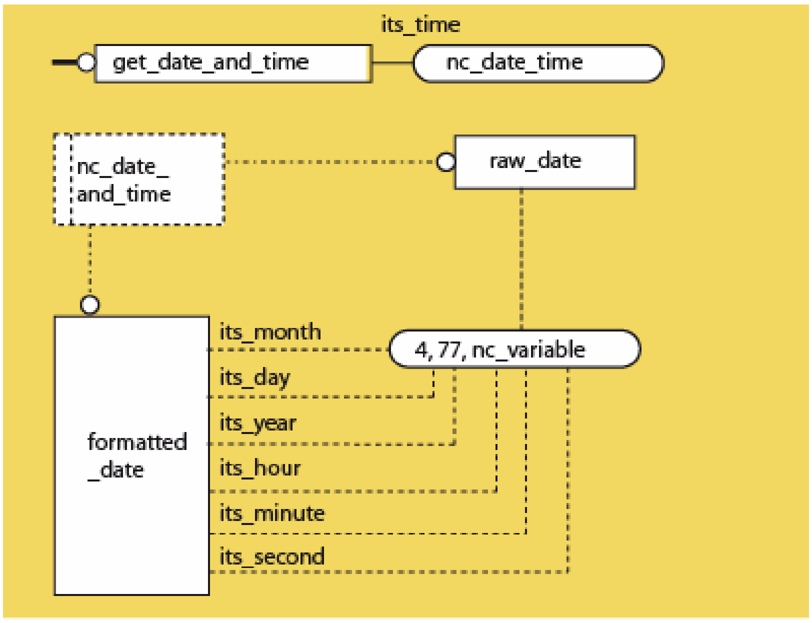
Possible data structures for this function:

a. If data will be recorded in AP-238,

The first option is an approach similar to bounded curve (no nc_variables), and the "get_date_and_time" nc_function is linked to a date_time structure as defined by module 10303-1010.

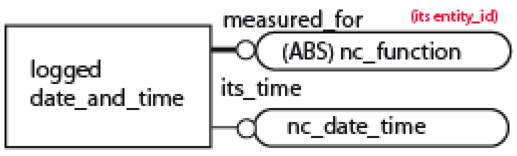


The second option is similar, but redefining the linked data structures as a set/array of nc_variable (real numeric type). A "date_time_format_select" has been defined to cover the case of returning and absolute "non-formatted" time value (like raw values in pc internal date counters), or a "formatted _date" value.



b. If data is not recorded in AP-238 and will be logged into a separate file.

It will be a task of the HMI/POST to take the "actions" to include in the log file the necessary data to link the AP-238 nc_function and the resulting value, for example, including nc_function entity id. The defined data structures (as shown in the first option), can be used to indicate the HMI/POST how to structure the logged data. So the information should be modelled somehow as follows:



In this case, it will be alsonecessary **if not done by default,** to write a **header in the log file** the linking information with the AP-238 file for which the log file will be written …

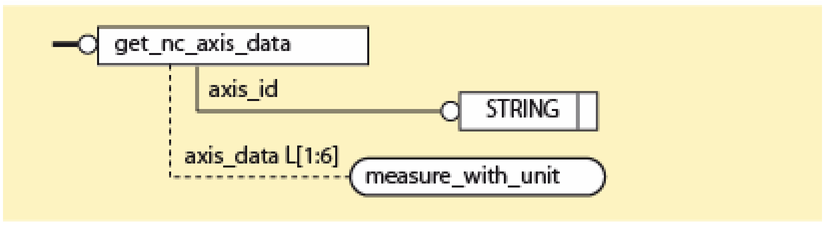
2.2 get_x_location_data,
2.3 get_y_location_data,
2.4 get_z_location_data,
2.5 get_i_axis_data,
2.6 get_j_axis_data,
2.7 get_k_axis_data.

These six functions can be grouped under a more generic **get_nc_axis_data,** the purpose of this function will be to collect the actual controller nc_data. A mandatory string will be used to identify what axis data to collect: "X", "Y", "Z", "I", "J", "K" or any valid combination of them, as "XYZ" … (the G-code conversor will break this string to find which variables/data it include in code to be collected). The alternatives for data definition are similar as those presented for the get_date_and_time_function.

Possible data structures for this function:

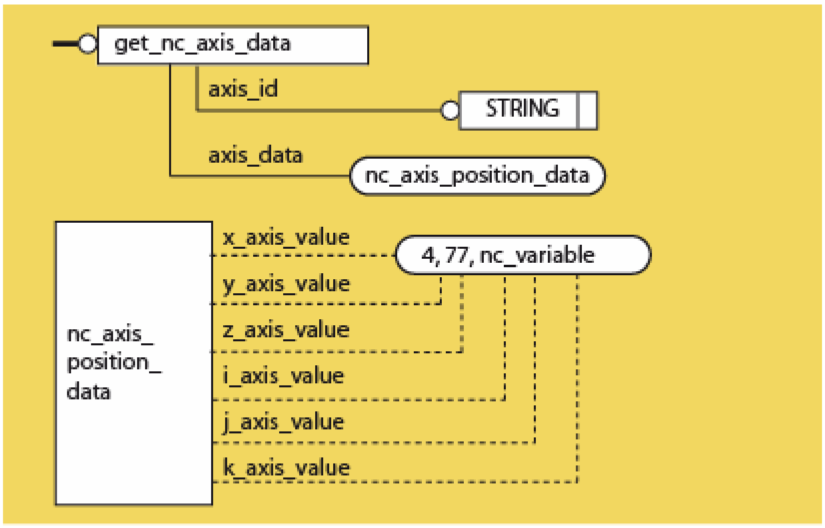
a. If data will be recorded in AP-238,

The first option is an approach similar to bounded curve (no nc_variables), and the "get_nc_axis_data" is linked to a axis_data structure of measure_with_unit values (to hold length or angle measures).



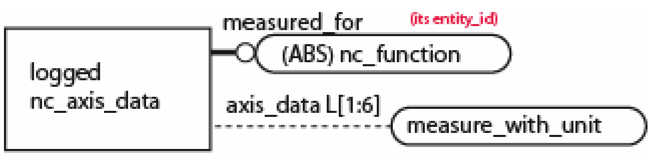
The second option is similar, but redefining the linked data structures as a set/array of nc_variable (real numeric type). A "nc_axis_position_data" has been defined. Maybe it will be possible to add extra "static*" axis information …

(I mean static, because it is supposed that these nc functions, as group I, will be used when axis motion is stopped)

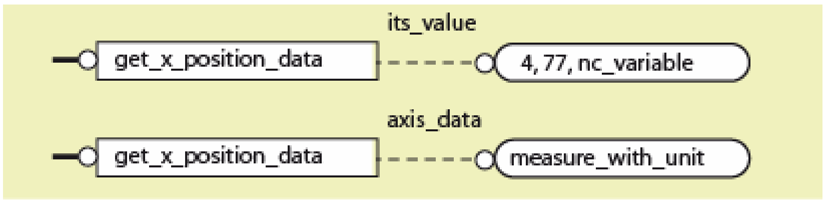


b. If data is not recorded in AP-238 and will be logged into separate a file.

It will be a task of the HMI/POST to take the necessary "actions" to include in the log file the necessary data to link the AP-238 nc_function and the resulting value, for example, including in the value the nc_function entity id and the meaning of the recorded values.



c. If we do not want to group these functions and follow the approach of six different nc_functions, options will be similar (as shown below for get_x_position_location)

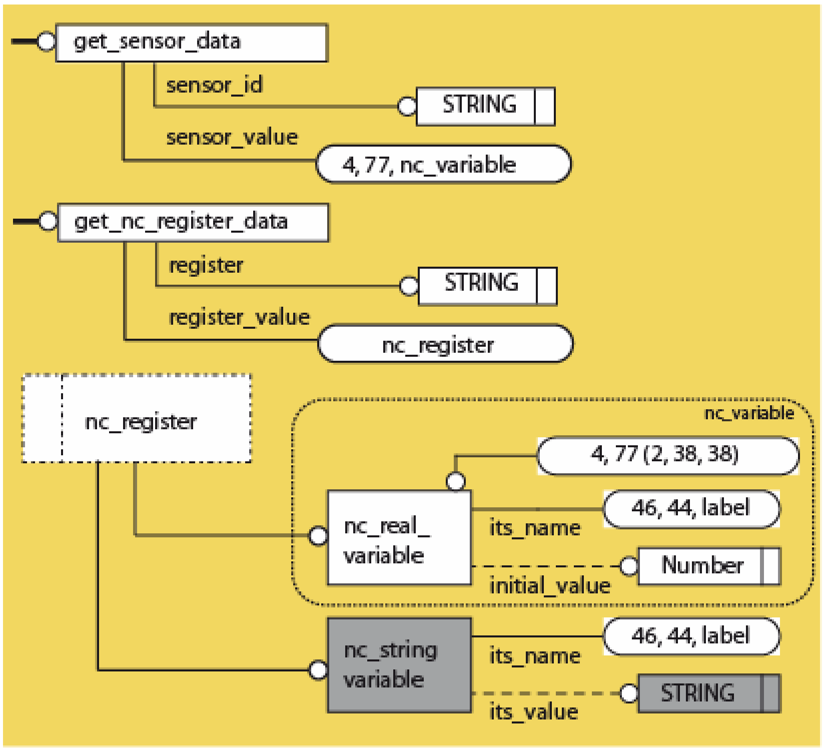


2.8 get_sensor_data*

The nc_function name can be “split” or not into **get_sensor_data:** the purpose of this function will be to collect nc sensor data (it could be through internal nc_register, as long as they are REAL values), and **get_nc_register_data** to collect data form other internal nc_register not expressed as real/numeric data types.

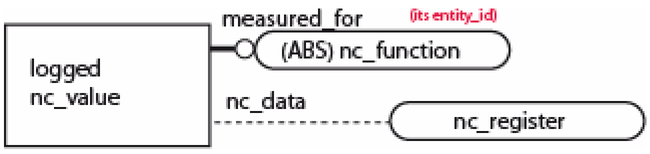
a. If data will be recorded in AP-238,

The proposed option is similar for both functions, but for the “get_nc_register” function it will be necessary to define a STRING variable, following a similar approach as it was used to define the real type nc_variable. A “nc_register” type has been defined as a select of nc_real_variable (wich is a nc_variable, and a defined nc_string variable).



d. If data is not recorded in AP-238 and will be logged into separate a file.

It will be a task of the HMI/POST to take the necessary “actions” to include in the log file the necessary data to link the AP-238 nc_function and the resulting value, for example, including in the log before the value the nc_function entity id and the meaning of the recorded values. An optional attribute for this function could be a STRING name indicating the file name to write the result.



5. ”GLOBAL” GROUP I functions.

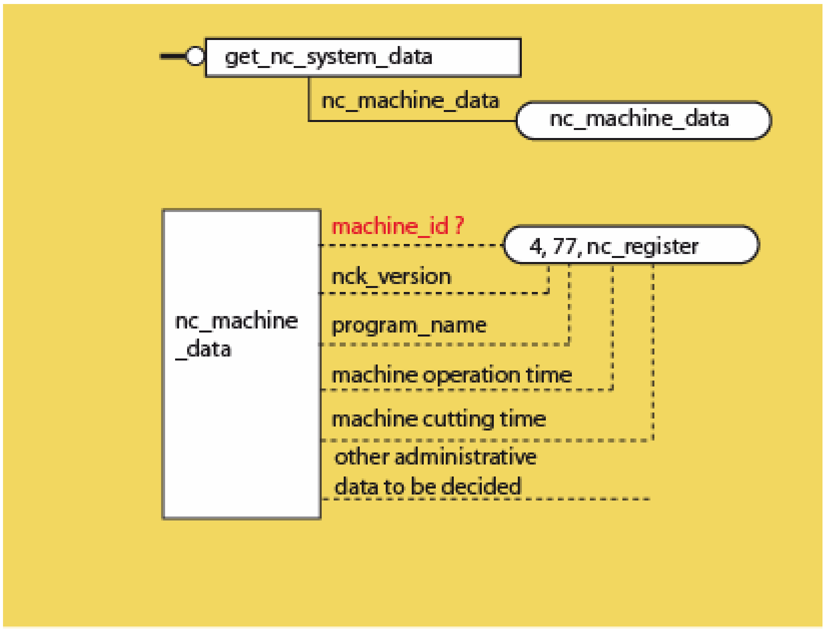
(Global = execute nc_function and get a series of values during workplan execution.)

The first question with this approach is where to insert these functions in the AP-238 executbale structure: **At the beginning ?, if not inserted at the beginning, what happens ?.**

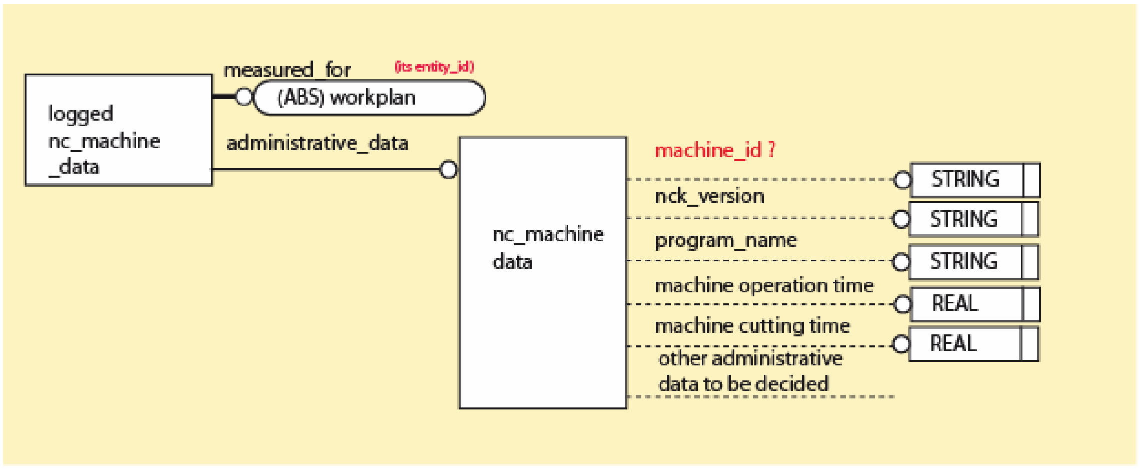
2.9 get_cnc_system_data

The “get_machine_id” nc_function name can be changed into **get_cnc_system_data:** the purpose of this function will be to collect the actual controller administrative data. It has been placed here because this function can be used for example collect data about the total program execution time or the total cutting cycle time, there are two possible approaches:

1. The function is placed always at the end of the workplan (before the program stops): At the end of the program data is dumped into the log file the predefined machine data.
2. The function is placed anywhere, but logging is delayed to the program end, so some sort of FLAGS have to be programmed internally to indicate that at the end of the program data is going to be logged…



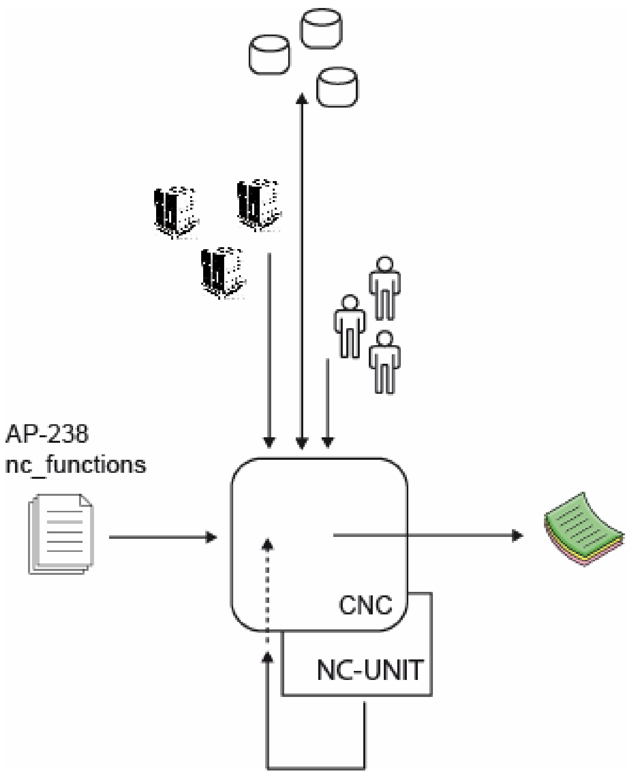
(data inside AP-238)



(data outside Ap-238 file)

2.9 get_operator_data

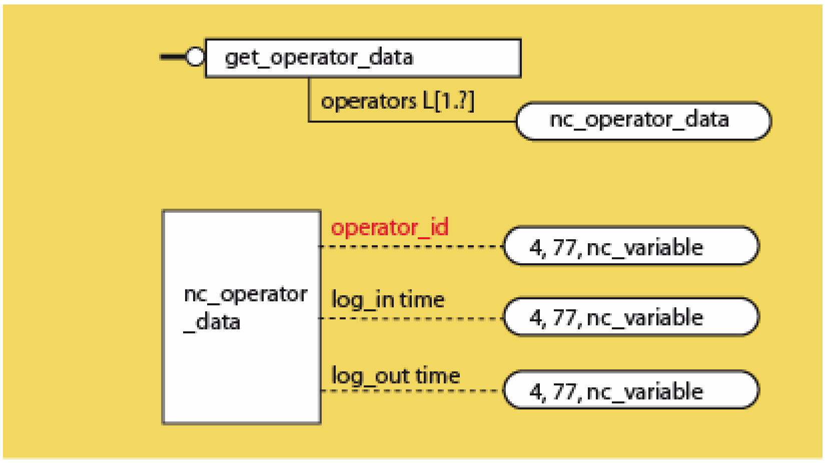
The "get_operator_id" nc_function name can be changed into **get_operator_data.** The purpose of this function will be to collect operator data during workplan execution. It is placed here IF its execution can be considered as **workingstep asynchronous**, meaning that operator changes can happen during workingstep execution.
Also, this function, the same as the following ones (for materials, events and tool_data) maybe will need to interact with other "inputs" not coming form the NC as seen in the figure below, so as operator logging is controlled but other systems or external databases, is it this automation inside Ap-238 scope (AP-238 CNC architecture)?



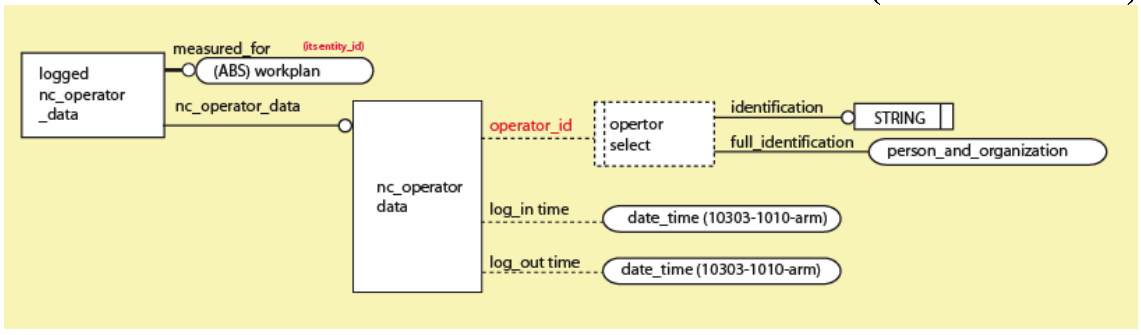
Another important observation for these functions is that "asynchronous events" such as operator log-in/log-out, and machine events are that their essential data is the "event time", so to connect this data with CNC machinig traceability (workingstep/features), it will be needed a complete trace of workingsteo execution time.

For example, if we have the logging time for operators working during a workpiece machining, and we know workingstep 4 was wrong, but we don't know when WS4 started and ended, we have no way to know which operator was working during WS4. A possible alternative would be to add WS information to the operator log, but can we guarantee the operator database, or logging system (maybe a remote/distributed one) knows which WS is in execution?

1. The function is placed always at the end of the workplan (before the program stops): At the end of the program data is dumped into the log file the predefined machine data.
2. The function is placed anywhere, but logging is delayed to the program end, so some sort of FLAGS have to be programmed internally to indicate that at the end of the program operator data must be logged…



(data inside AP-238)



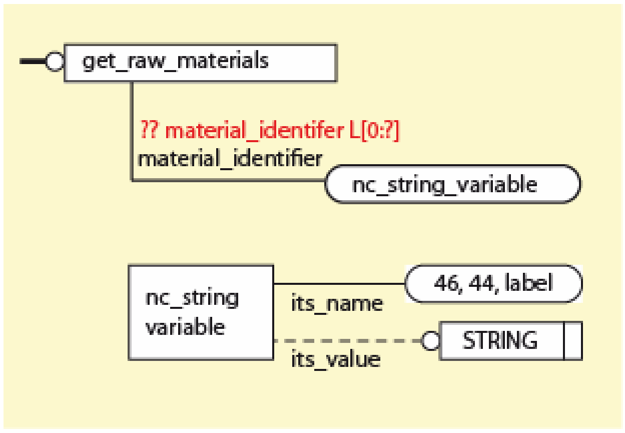
(data outised AP-238)

2.10 get_raw_materials

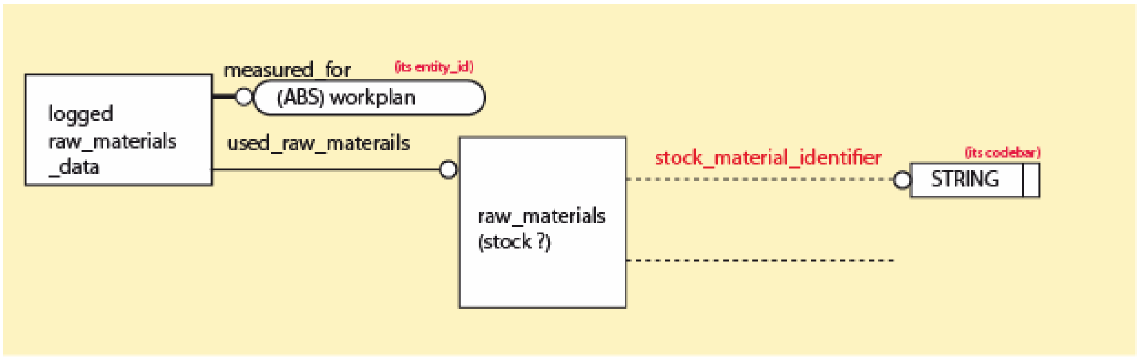
The purpose of this function will be to collect raw_materials data employed for part machining.

It is allowed to have more than one stock piece ?

Should we consider coolant_types and other resources as raw materials?



(data inside AP-238)



(data outside Ap-238)

2.13 get_tool_data*

2.14 get_execution_times

2.15 get_machine_events

6. LAST QUESTION

<u>**Do we need an input nc_function ? (similar to display message but to force the operator to enter some data ? → CNC architecture dependant (HMI))**</u>