METUCube Code Generations Requirements

May 10, 2025

Here you can find the necessary requirements for the codes that will be embedded. Note that these rules are not enforced for ground software, although always recommended for safe application development. The requirements are summarized from a NASA document on safe coding, which can be found in References section.

1. Code in C++ or C if possible. Never Use Python. MATLAB is applicable
2. Never use dynamic arrays. If you must use a list, pre-allocate its length. Never use functions like array.push()
3. A list can not contain two different data types. Decide one, and proceed with it
4. Do not use object oriented programming, or MATLAB Classes
5. Do not use recursive functions. Try not call a function when inside a function. Should needed, test the use cases to make sure there is not a bug that causes an infinite loop
6. If you need to use a struct or a dict, always initialize upfront with necessary items defined with their data types. Note that it is not recommended. Use lists instead, and do not nest them unless they are staticly defined

   ```
   % Allowed: Predefined struct
   config = struct('sensor_id', 0, 'buffer', zeros(1, 10));

   % Forbidden: Adding fields dynamically
   config.newField = 5; %
   ```

7. Try not to use MATLAB functions as long as possible. Should needed, check whether it is supported for code generation or not using this link. For example, commonly used **adjoint** function is not supported.
8. Try not to use MATLAB toolboxes. Should needed, check whether it is supported for code generation or not using this link
9. When defining variables, make sure their data types are defined such as double, or int16. MATLAB code generator assigns them, but they not be possible. If this is not applicable, document needed data types for variables you used, so we can make sure everything work as expected.

10. Do not use while loops except the main entry point. Should needed, add a bound counter.

```
maxIterations = 100; % Defined as a compile-time constant
iter = 0;
while (iter < maxIterations) && ~done
    iter = iter + 1;
    % ... logic ...
end

% OR ---------------------------------

% Safer alternative
for attempt = 1:10
    if readSensor()
        break;
    end
end
```

11. Avoid conditions relying on floating-point math. Note that the conditions in your computer may not occur in embedded systems.

```
if(abs(x - y) > 1e-6) % Non-deterministic condition, not allowed
```

12. Always explicitly cast variables when changing types. Avoid relying on MATLAB's implicit type coercion.

```
% Forbidden: Implicit cast
x = y + 5; % If y is int16, 5 is double → risky!

% Allowed: Explicit cast
x = int16(y + int16(5));
```

13. Avoid global variables. If unavoidable, declare them as **persistent** with explicit initialization:

```
function y = myFunc()
    persistent count
    if isempty(count)
        count = int32(0); % Explicit init
    end
    count = count + 1;
    y = count;
end
```

14. If you need to use timer or an interrupt driven code, contact with the team who will embed the code as soon as possible. Try to avoid though, since it will require RTOS applications

15. Your functions should return status codes in addition to their returned data. You are expected to write code for each error case to determine and isolate any errors. If you need global help in reporting and logging these errors, contact with the team who will embed the code as soon as possible. Define clear error codes for your sub-systems, and document them.

References

- NASA JPL C Coding Standard.
- MISRA C++ 2008 Guidelines.
- MATLAB Coder Documentation.