# Matlab, Simulink, and C programming guidelines

This document describes some straight-forward guidelines related to software development to assure consistency and standardization when producing software functions and simulation models. The intention is to have a minimum of quality assurance when developing a software library in a common effort from our research group.

## First of all!

In *Matlab preferences 🡪 Simulink preferences 🡪 General*: Select **“Simulation cache folder”** and **“Code generation folder”** to a local folder on your computer and not the working folder under the Git repository. This avoids the annoying “slprj” folders, etc., to be included in the library.

## Color coding of Simulink blocks

* **GREEN:** Sources (inputs ports, constants, etc.)
* **RED:** Sinks (output ports, terminators, to workspace blocks, etc.)
* YELLOW: All remaining blocks, except....

…some blocks have special colors:

* **CYAN: Memory blocks:** *Integrators, unit delays, memory, transfer functions, state space blocks, etc.*
* **GREY: Logics blocks**: *Logic action ports, subsystem enable, etc.*
* **ORANGE: GOTO blocks.**
* **MAGENTA: FROM blocks.**

## Naming convention

* Give all subsystems and input/output ports describing names (name should indicate function of the system or signal).
* Hide names of all commonly used blocks (gains, product, switch, integrators, limiters, etc.).
* Use the camelCaseNamingConvention for naming of subsystems and input/output ports (start with lowercase letter).

## Library blocks

* Mask library blocks with proper description of block functionality and with author and creation/revision date of the library block.
* All external parameters used in the library block should be listed as parameters in the library mask.
* Create documentation for the library block (template to be made).

## Other

* Programming flow should go from left to right (as much as practically possible). This means that input ports should be placed to the left and output ports should be placed to the right.
* Use constants and product blocks instead of gains (for maintaining tunability of parameters after code generation).

## MATLAB function header:

% [output1, output2, etc.] = FUNCTION(input1, input2, etc.)

%

% <description of function>

%

% Inputs:

% input 1 - <description>

% input 2 - <description>

% etc.

%

% Outputs:

% output 1 - <description>

% output 2 - <description>

%

% Copyright: NTNU, Trondheim, Norway

% Created: <date> <author>

% Revised: <date> <author> <description>

% <date> <author> <description>

%

## Simulink C-coded S-function header:

/\*

sFunctionName.c

<Description of function/block>

Input parameters:

Parameter 0 - <Description>

Parameter 1 - <Description>

etc.

Input ports and signals (size):

I0 - <Port description>:

0: <Signal description>

1: <Signal description>

etc.

I1 - <Port description>:

0: <Signal description>

etc.

Output ports and signals (size):

O0 - <Port description>:

0: <Signal description>

1: <Signal description>

etc.

O1 - <Port description>:

0: <Signal description>

etc.

Compile with: > mex -O sFunctionName.c <library files>

Copyright: NTNU, Trondheim, Norway

Created: <date> <author>

Revised: <date> <author> <description>

<date> <author> <description>

\*/

## Simulink library block (masked)

**Mask documentation 🡪 Mask description**

Name and description of block

where

input 1: <name and dimension>

input 2: < name and dimension >

% Roger Skjetne - 08.03.2014

**Mask Icon & Ports**

Label each input and output port, e.g.:

port\_label('input', 1, 'xl\_name')

port\_label('input', 2, 'x2\_name')

port\_label('output', 1, 'y1\_name')

port\_label('output', 2, 'y2\_name')

## Experiences

GOTO and FROM blocks should use global visibility and cannot be located within:

* Enabled Subsystem.
* Subsystem with “Treat as atomic unit” on.

The “printf” function may cause Matlab to crash when used within an s-function thread.

*-any new tips: please include them here!*