**Coding Style Conventions:**

*Motor Control Firmware (MCFW)*

|  |  |  |
| --- | --- | --- |
| **Authors** | **Name** | **Department** |
| 01 | Hamid Behjati | IFAM PSS DCDC RD |

|  |  |  |
| --- | --- | --- |
| **Revisions** | **Date** | **Description** |
| 01 | 03-13-2023 | Initial draft |

Table of Contents

[1 Scope 3](#_Toc166157931)

[2 Folder Style 4](#_Toc166157932)

[3 C/C++ Style 5](#_Toc166157933)

[3.1 File Names 5](#_Toc166157934)

[3.2 Header File Struct Type Definitions 5](#_Toc166157935)

[3.3 Header File Functions 5](#_Toc166157936)

[3.4 Private Static Functions Inside “.c” Files 5](#_Toc166157937)

[3.5 Constants and Enum Types 6](#_Toc166157938)

[3.6 Variables and Struct Instances 6](#_Toc166157939)

[3.7 C++ Classes 6](#_Toc166157940)

[3.8 C++ Class Members 6](#_Toc166157941)

[3.9 Macros and Defines 7](#_Toc166157942)

[4 MATLAB Style 8](#_Toc166157943)

# Scope

This document presents the coding style conventions for MCFW.

It is important to follow these conventions to make the code *easily readable* and *maintainable*.

It must be noted that these conventions only apply to the core application code.

The lower level code (HALs, PDLs, FreeRTOS, etc.) does not follow a unified coding style e.g.

* *HALs*: no difference between the style of file names (cybsp.h), structs (wdt\_obj), and functions (cyhal\_wdt\_init)

**#include** "cybsp.h"

result = cyhal\_wdt\_init(&wdt\_obj, cyhal\_wdt\_get\_max\_timeout\_ms());

* *FreeRTOS*: our style is more similar to FreeRTOS which is the industry standard

**#include** "FreeRTOS.h"

xTaskCreate(task\_blink, "blink task", configMINIMAL\_STACK\_SIZE, NULL, configMAX\_PRIORITIES - 7, NULL);

vTaskStartScheduler();

# Folder Style

For the folder names should be in PascalCase e.g. “OperationalCode”.

# C/C++ Style

## File Names

#include "StateMachine.h"

File names should be PascalCase.

## Header File Struct Type Definitions

"StateMachine.h":

typedef struct

{

STATE\_t states[State\_ID\_Max];

STATE\_ID\_t current;

STATE\_ID\_t next;

STATE\_VARS\_t vars;

} STATE\_MACHINE\_t;

extern STATE\_MACHINE\_t sm;

If file name is “XxxYyy.h”, the structure type definitions in the header file should start with “XXXX\_YYY\_...” and end in “\_t”

## Header File Functions

"StateMachine.h":

void STATE\_MACHINE\_Init();

void STATE\_MACHINE\_RunISR0();

void STATE\_MACHINE\_RunISR1();

If file name is “XxxYyy.h”, the functions in the header file should start with “XXX\_YYY\_...” (file name) and end in “\_WwwZzz” (function description).

## Private Static Functions Inside “.c” Files

"StateMachine.c":

static void EmptyFcn() {};

static void (\*CommonISR0Wrap)() = EmptyFcn;

static void (\*CommonISR1Wrap)() = EmptyFcn;

The static functions should not include the header file’s name, only the function description i.e. “WwwZzz()”.

## Constants and Enum Types

typedef enum

{

Direct = 0,

Observer = 1

} FB\_t;

const uint32\_t Run\_Time\_Ticks = static\_cast<uint32\_t>(Run\_Time\_Sec \* params.sys.samp.fs0);

Constants and enum types should be in “Xxx\_Yyy\_Zzz” format.

## Variables and Struct Instances

float i\_cmd\_ext; // External, RFO

QD\_t i\_qd\_r\_ref; // After applying phase advance, RFO

vector<float>::const\_iterator input\_ss\_last = input.end();

Variables and struct instances should be in “xxx\_yyy” format (snake\_case).

## C++ Classes

template<class T>

class CDiscreteFourierTransform

{

public:

CDiscreteFourierTransform(vector<T>& signal);

~CDiscreteFourierTransform();

vector<complex<T>> ExecuteAll();

private:

vector<complex<T>> m\_result;

};

C++ classes should start with “C...” i.e. “CDiscreteFourierTransform”.

## C++ Class Members

class CBisectionRoots

{

public:

using EquationFunc = float (\*)(float);

private:

EquationFunc m\_equation;

MINMAX\_t m\_lim;

float m\_error\_max;

};

We should distinguish between class members and non-members by including “m\_...” in the member variable definition.

## Macros and Defines

#define SCALE\_PI\_TO\_INT32 ((float)(INT32\_MAX) / M\_PI)

Macros and defines should be “XXX\_YYY” format.

# MATLAB Style

MATLAB filenames, constants, variables, structs, and functions should follow the same conventions as C/C++ e.g.

%% File Names (PascalCase)

CleanWorkspace.m

%% Constants

MATLAB\_Overwrite\_Params = true;

%% Variables and Structs (snake\_case)

c\_sin = sqrt(1-c\_cos^2);

m.sys.ts = p.sys.samp.tsim;

%% Functions (PascalCase)

CleanWorkspace;