Embedded C Coding convention

## General appearance

* No tab
* No trailing space
* 1 tab = 4 spaces
* A line cannot longer than 80 characters

## Scope

Variable, function, type, macro should have scope as small as possible.

* If variable, macro, type and function don’t need to put in h file, don’t do that
* If variable, function only access within a c file, declare only in c file with *static* keyword
* If variable don’t need to declare as global variable, declare it as local variable
* If variable don’t need to declare with *extern* keyword, don’t do that

## Naming Convention and rules

### Global Variable:

[Module Name]\_[VariableName]

* Camel case for VariableName.
* Module Name should less than or equal 4 characters

char Fat\_MyVariable;

### Local Variable:

Local variables used inside function should be defined at the beginning of function.

Name of variable should be meaning and it more than one character.

Exception: with below case can accept:

for (i =0; i<n; i++) => variable ‘i’ is accept

{

for (j=1; j<5; j++) => variable ‘j’ is not accept

…….

}

One local variable get its own line but not add several local variables into one line like

uint32\_t a, b, c; /\* Wrong format. \*/

### Function name

[Module Name]\_[FunctionName]

void Srec\_MyFunction(char MyParameter)

### Type

[TypeName]Type

typedef int StudentMaxType

### File name

[Module Name].c

Fat.c

[Module Name]\_[Something].c

Fat\_Callback.c

[Module Name].h

Fat.h

[Module Name]\_[Something].h

Fat\_Types.h

**Exception: main.c is accepted**

## Comments

Always use "/\*\*/" for code comments.

Examples:

int i;

for (;;)

{

i++; /\* Increase i. \*/

}

## Hard Coding Numbers

All hard coding unsigned numbers should be postfixed with 'U'. MISRA 2012 Rule 7.2: A 'u' or 'U' suffix shall be applied to all integer constants that are represented in an unsigned type

Examples:

Hex: 0x1234ABCDU

Dec: 1234U

## Preprocessor

* Macro should use UPPER case letters

#define MAX\_STUDENT (3U)

#define SUM(a,b) ((a)+(b))

* Comments should be appended after the #endif:

#if I\_AM\_A\_MACRO

#endif /\* I\_AM\_A\_MACRO \*/

* Add a blank line at the end of the file.
* Header files guard macro

#ifndef \_HEADER\_FILENAME\_

#define \_HEADER\_FILENAME\_

#endif /\* \_HEADER\_FILENAME\_ \*/

## General comment style

Add macros, enumeration types, structure types, inside:

**Add macro definition inside:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Definitions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Add global variables definition inside:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Variables

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Add static API prototype in C file inside:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Prototypes

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Add function implementation inside:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Code

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Add public API in header files inside:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* API

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

* For header file, the sequence is:
  + Include
  + Definition
  + Type
  + Extern variable
  + API
  + End of file
* For C file, the sequence is:
  + Include
  + Definitions
  + Type
  + Prototypes
  + Variables
  + Codes
  + End of file

With prototype and API, should be add comment to explain about function, input, output, return as below sample:

/\*\*

\* @brief Read all address, data in the file, following template.

\* With each error line, print all in that line, attach the error name.

\* @param[in] fptr: The pointer of the record file

\* @param[in] Array: Store the elements of each line

\* @param[in] error\_Arr: The array stores the value of all error check functions

\* @param[out] None

\* @param[inout] None

\* @returns None

\*/

**void read\_File(FILE \*fptr, uint8\_t Array[], uint8\_t error\_Arr[]);**