**AIISRR** Artificial Intelligence Infinity Software Reliability Regulation

AIISRR-C-: 2010

Regulations for the use of the C language in critical system

1. 开发工具
   1. 代码编辑器

就目前而言AII建议采用较新版本的Microsoft Visual Studio和其环境增强插件WTS Visual Assist X作为代码编辑器并且采用Source Insight作为代码浏览工具。其余编辑器如VIM, EMACS, eclipse以及Ultra Editor也可以使用，但是必须配置成与VS+VA兼容的设置。出于开发效率和项目兼容性考虑，Microsoft Visual Studio 2008 + WTS VAX是目前AII大部分项目的标准开发工具。

* 1. 编译器

**Windows Platform:** 唯有MS Visual Studio Compiler 和 Intel Compiler可保证一直用于一般性项目开发。任何一个MS Visual Studio Compiler和Intel Compiler版本必须经过AII技术委员会定期认证方可用于特定项目开发。可在Windows平台运行的其余编译器必须经过AII技术委员会认证方可用于项目开发。

**Linux Desktop Platform:** 唯有GCC可被AII技术委员会定期认证并可用于一般性项目开发。一些基于Intel平台的项目可采用Intel编译器Linux版本，但必须经过AII技术委员会认证方可用于项目开发。

**Linux Embedded Platform:** 任何项目不得使用开源Linux编译器，商业版编译器必须经过AII技术委员会认证方可用于项目开发。当前而言Windriver System Linux解决方案以及QNX解决方案可用于AII的嵌入式项目开发。

**vxWorks Embedded Platform:** 唯有vxWorks6.0以上的版本才能用于AII嵌入式项目开发。

**MAC OS X Platform:** 唯有xcode开发平台才能用于AII的项目开发。

* 1. 项目版本管理

AII recommends Subversion for small scale projects and git for large scale projects. AII will introduce commercial version management solution later.

1. Project Structure
   1. General Folder Structure

The project folder should contain at least three directories: build, src, doc, whatever it is used for any type of project and in any development phase.

The folder “build” contains project definition files which are solution setting files in Visual Studio of Windows platform, Makefile in GCC platform or other project setting files of other platform. Different platforms must settle on different folders and name them with platform name. Standard name must be “linux”, “win32”, “macos”, “vxWorks”.

The folder “src” contains all the C source code and header files. The internal structure of “src” folder are slight different in different project type.

The folder “doc” contains all the development documents of the project. The development documents should be cataloged into different folder. Only MS office format and PDF are acceptable .

* 1. Application Integration Development

The folder structure of Application Integration Development must compliant with General Folder Structure. Other directories can be added depends on project requirements.

In folder “src”, creates an individual sub-folder for each module in which contains C code file and internal header files. The header files (in most of the cases just one head file for each module) which define interfaces and open data structure definition or objects must settle on public “include” folder of such project. The project only has one public “include” folder which contains the open header files of all the modules.

* 1. Library Development

The Library Development Project contains folders: “build”, “src”, “include”, “lib”, “doc”. The structure of folder “build” and “doc” are compliant with General Folder Structure.

The folder “src” contains all the C source code files and internal header files of each module.

The folder “include” contains all the public header files of each module.

The folder “lib” contains all the compiled dynamic link library files on every possible supported platform. The DLL files of every platform settle on individual sub-folder with platform name which must be standardized as: “linux”, “win32”, “macos”, “vxWorks”.

* 1. Module Development

The Module Development Project contains folder: “build”, “src”, “doc”.

1. Name Regulation
   1. 命名宗旨

对于命名的规定出于统一代码风格的目的，基于美观实用的原则。虽然本文档对命名的规则给出了大部分建设性意见，但是命名还是允许很大的自由度和创意性。本部分除了强制遵守的内容外，其余建议在实施时视具体情况允许自由创造。

* 1. 文件命名规则

1. 模块公开接口访问头文件名称必须和模块名称一致。模块名称由相关设计文档决定并与包含模块代码文件的文件夹名称一致。（强制）
2. 项目工程之下的所有文件夹以及文件名包括代码文件文档文件的名称中不允许含有空格。名称一律采用英文小写字母，不允许采用汉字或者其他语言字符命名（特殊代码文件可能含有特殊的转义字符如object-C代码文件名中可能含有’@’，这些不在本规则限定范围内）。（强制）
3. 代码文件名命名采用小写字母简称，不同意群不必采用下划线’\_’或者大写字母隔离。（建议）
4. 代码文件名必须以英文为基础，反映本代码文件的功能性，不得采用汉语拼音或者与汉语拼音相关的简称。（强制）
5. 无论是那种语言撰写的文档，其文件名必须采用英文命名。（强制）
6. 文档文件名可以包含暗示语言种类的名称字母，如en, chs, cht, 等等。（建议）
   1. 代码命名规则
      1. 函数名命名规则
7. 函数名称命名必须符合C语言语法规范。（强制）
8. 函数名前31个字母必须唯一。（强制）
9. 函数命名必须以英文为基础，反映本函数功能性，不得采用汉语拼音或者与之相关的简称。（强制）
10. 函数名一般采用英文小写字母，不同意群采用下划线’\_’隔开。（建议）
11. 在相同层次作用域，函数名称不得与其他结构体或者变量名称同名。（强制）

例如：

typedef struct

{

…

}circle\_def;

int circle\_def(…)

{

…

}

以上代码在编译时并不会报错，但是很容易引起使用者的混淆。也造成代码静态检测辅助工具的失效。

* + 1. 变量名命名规则

1. 任何变量的名称必须符合C语言语法规范。（强制）
2. 任何变量名的前31个字母必须唯一。（强制）
3. 任何变量命名必须以英文为基础，反映本变量的含义，不得采用汉语拼音或者与之相关的简称。（强制）
4. 任何变量名一般采用英文字母，不同意群采用下划线’\_’隔开。（建议）
5. 变量的最后一个意群可以通过一定的字母简称反映变量的类型。（建议）如：

void\* var1\_ptr; /\* 指针变量 \*/

uint32\_t var2\_ui32; /\* 32位无符号整型 \*/

int32\_t index\_array[100]; /\* 数组 \*/

1. 函数内部定义的动态本地变量，须以字母开头，不得采用下划线’\_’或者数字开头。（强制）
2. 相邻两级作用域不得出现同名称的标识符。(强制)。如：

static int32\_t var\_i32;

int32 function (void)

{

int32\_t var\_i32;

}

typedef struct a\_struct

{

…

};

int32\_t function (void)

{

struct a\_struct a\_struct;

…

}

1. 结构体类型定义一般采用如下范例：

typedef struct

{

…

}STRUCT\_NAME, \*STRUCT\_NAME\_PTR;

或者

typedef struct \_node\_name

{

…

struct \_node\_name\* next\_node;

struct \_node\_name\* previous\_node;

}STRUCT\_NAME,\*STRUCT\_NAME\_PTR;

其中结构体类型名称一般采用大写，不同意群之间采用下划线’\_’隔开。（建议）

* + 1. 宏定义命名规则

1. 任何宏定义的名称必须符合C语言语法规范。（强制）
2. 任何宏定义的名称前31个字母必须唯一。（强制）
3. 任何宏定义命名必须以英文为基础，反映本变量的含义，不得采用汉语拼音或者与之相关的简称。（强制）
4. 宏定义名称一般采用英文大写字母，不同意群采用下划线’\_’隔开。（建议）
   * 1. 类型定义命名规则