GDKY

河北工大科雅能源科技股份有限公司

GDKY/JS-GF-03- ES-03-2019-A-1

嵌入式软件开发规范

2019-07-01 发布　　2019-07-01 实施

电子技术部软件组发布

前　言

程序编码是技术和艺术的结合，既灵活又严谨，充满了创造性和奇思妙想。然而工业软件设计是由团队协作完成的，而非程序员个人艺术展示舞台。每个程序员都有自己的编程习惯和编码风格，如果缺乏统一的规范约束，一个产品软件的最终程序风格迥异，可读性，可维护性均较差，不仅给理解代码带来障碍，也增加后期维护扩展的工作量。此外，不规范的设计和编码行为往往还会隐含设计错误。

为规范软件设计和编码行为，提高团队协作开发效率和可靠性，增强代码的可读性，可维护性，提高编码质量与效率，保障软件整体的品质与可持续开发扩展性，特拟定此规范。本规范包含嵌入式软件产品设计规范，编码规范，管理维护。

本规范是嵌入式软件开发规范，参考了相关国际和国家标准、国内公司已实行的软件规范，力求软件团队协作更加有序和稳定。团队开发需要一个逐步标准化，规范化的过程，在这个过程中，团队得到锻炼，成员能力得到提高，产品质量得以控制。

目　　录

前　　言 II
目　　录 III
1 规范说明 1
1.1　规范目的 1
1.2　适用范围 1
1.3　引用标准 1
1.4　规范术语 1
1.5　文档版本 2
2　设计规范 3
2.1　设计流程 3
2.1.1　需求分析 3
2.1.2　功能设计 3
2.1.3　软件设计 4
2.1.4　软件编码 4
2.1.5　验证测试 4
2.2　代码组织 5
2.2.1　源码目录 5
2.2.2　命名方法 7
3　编码规范 8
3.1　程序命名 8
3.2　程序注释 10
3.3　程序排版 13
3.4　程序规范 13
4　软件维护 16
4.1　版本管理 16
4.2　版本控制 16
4.3　测试管理(BUG 提交、分配、解决等) 16
4.4　项目管理(工作量化、时间管理、项目归档等) 16
附录 A 常用名称缩写 18
参考文献 19

1 规范说明

1.1 规范目的

统一嵌入式软件开发的设计规范；软件编码风格和规范性；促进团队成员的沟通，减少误解；提升软件代码的可读性，可维护性和可扩展性；提高嵌入式软件开发的效率和质量。

1.2 适用范围

参与嵌入式软件编程的所有人员都必须遵守本嵌入式软件开发规范。其中“编码规范”部分规则适用于嵌入式 C 语言程序，使用其他编程语言可以酌情参考。引用第三方标准的源代码不受限于此规范，在编写此类标准源代码的驱动或接口程序时，可以优先遵循所使用的标准源代码规范，但基于该标准源代码 API 编写的应用程序必须遵循此规范。

1.3 引用标准

下列标准中的条款通过本标准的引用而成为本标准的条款。凡是注日期的引用文件，其随后所有的修改版（不包括勘误的内容）或修订版均不适用于本标准。凡是不注日期的引用文件，其最新版本适用于本标准。如引用标准与本规范冲突，以本规范为准。

ISO/IEC 9899:1990 Programming language-C C99 标准

1.4 规范术语

本嵌入式软件开发规范，采用以下的术语描述：

- **规则**：在软件开发过程中强制必须遵守的行为规范。
- **建议**：软件开发过程中必须加以考虑的行为规范。
- **说明**：对此规则或建议进行必要的解释。
- **示例**：对此规则或建议从正或反两个方面给出例子。

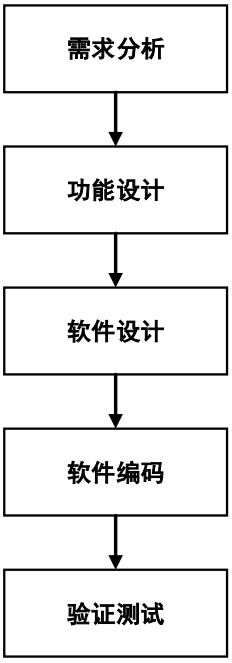
1.5 文档版本

版本	日期	说明	修订人
V1.00	2016-01-15	初次编写	邹增晖
V1.01	2016-01-19	讨论修订部分规范要求	邹增晖
V1.02	2019-07-01	修改添加部分规范说明	邹增晖

2 设计规范

本部分是软件开发过程的高层描述。它规定了软件设计的必要过程，用来和所有涉及软件设计各方就基本开发过程达成共识。

2.1 设计流程



2.1.1 需求分析

此阶段根据项目需求进行可行性分析，充分理解要解决的问题及需要达到的目标。充分理解和询问项目的需求，包括产品需要实现的功能、产品预期效果、未来扩展和应用环境等。

2.1.2 功能设计

此阶段在对需求分析充分无遗漏的基础上，讨论确定如何在给定的约束条件下解决项目问题和完成目标。与硬件工程师讨论确定硬件平台，包括处理器、硬件部件、外部接口等；软件平台，包括操作系统、编程语言、开发工具、调试工具、软件组件等；并且需要讨论产品方便后期测试和生产。

规则 2.1.1：此阶段结束需**提交功能设计**相关文档，内容包括软件功能、软件框架、操作系统、编程语言、开发工具、调试工具、软件组件等。使用的软件工具和软件组件必须说明相应的版本和应用环境，一旦确定，项目整个开发过程中的所有相关人员必须使用相同的开发环境。

2.1.3 软件设计

此阶段是对软件开发工作任务进行分析确定，制定计划，估算工作量，讨论确定软件程序架构、程序模块、模块接口、实现方法、通信协议、任务安排、开发时间等。当发生需求更改时，必须修订软件开发计划。此阶段的讨论项目软件开发相关人员参加，在通信协议制定时，需要涉及的相关部门开发人员参与。嵌入式程序开发任务主要分为以下几个模块：系统集成、硬件驱动、通信协议、数据采集、数据存储和配置软件等几个部分，在多人协作开发时，需要讨论确定各个模块之间的交互方式。

规则 2.1.2：此阶段结束需要**提交软件设计**相关文档，内容包括通信协议，任务安排，预计开发时间以及多人协作时各程序模块的接口 API 说明等。

2.1.4 软件编码

此阶段是根据任务安排，软件开发人员编码和调试过程。此阶段在多人协作开发时，主要涉及各程序模块之间交互接口的调试和配合，以及最终的系统集成调试。

规则 2.1.3：此阶段结束后参与软件编程的相关人员需要提供程序概要流程图，多人协作时各人员需要提供程序模块的接口说明，系统集成人员需要编写最终的功能说明，项目负责人负责把关各成员代码规范及质量，协调各成员之间配合方式，以及各成员代码集成验证。

2.1.5 验证测试

此阶段主要为软件功能的测试，根据测试中问题修改和调试。包括可靠性、稳定性、实用性和检验生产的便宜性等。

2.2 代码组织

2.2.1 源码目录

规则 2.2.1：同一项目软件开发所有人员的源代码目录和文件命名必须始终一致，源代码目录按照规则有序放置，需根据实际项目文件组织内容，修改对应项目目录说明文件。

规则 2.2.2：新项目应尽量参考目前正在使用的项目目录组织形式，其模板为“嵌入式规范标准库”，如不能满足新项目需要而更改添加目录组织形式，必须在项目根目录添加说明。

示例 2.2.1：嵌入式实时操作系统软件项目开发源代码目录。

※目录说明：此目录为公共库的全部目录，不同项目使用的目录和组件多少和类别不同，根据具体需求删除不需要的目录。

```
|
|-components 标准组件源代码
|       |-cpulibs CPU 标准驱动库
|       |    |
|       |    |-CMSIS   ARM Cortex-M 标准组件
|       |    |-STM32F4xx_StdPeriph_Driver STM32F40x 系列标准驱动库源代码
|       |
|       |-filesystem 文件系统
|       |    |
|       |    |-Reliance-Edge 嵌入式可靠存储小型文件系统，带掉电保护
|       |
|       |-protocols 标准应用层协议源代码
|       |    |
|       |    |-Modbus Modbus RTU/ASCII 主从机协议源代码
|       |
|       |-rtos 操作系统源代码
|       |    |
|       |    |-FreeRTOS 免费开源的嵌入式实时操作系统内核：http://www.freertos.org/
|       |
|       |-stdlibc C 标准库头文件，方便查看函数声明
|       |
|       |-usrlibc 自定义常用标准库函数
|
|-documents 工程相关文档(功能说明,通信协议,数据存储说明等)
|
|-examples   基于公共库功能实例演示项目(eg: RTOS, Ehternet, NB-IoT 应用功能演示等)
|
|-projects   开发环境工程目录(基于此开发环境模板工程)
|      |
|      |-IAR-430(V5)   IAR MSP430 开发环境，软件版本为 V5.x
|      |-IAR-STM8(V6) IAR STM8 开发环境，软件版本为 V6.x
|      |-IAR-ARM(V7)   IAR ARM 开发环境，软件版本为 V7.x
|      |-MDK-ARM(uV4) MDK ARM 开发环境，软件版本为 V4.x
|      |-MDK-ARM(uV5) MDK ARM 开发环境，软件版本为 V5.x
|
|-sources 项目应用源代码
|      |
|      |-includes.h 项目公共头文件
|      |-FreeRTOSConfig.h RTOS 配置头文件
|      |-main.c 程序执行入口文件
|      |-app 应用层源代码
|      |-app.c/h 主应用程序汇总文件
|      |-app_cfg.h 应用程序配置文件
|      |-app_hook.c 应用程序回调函数钩子文件
|      |  |
|      |  |-elog 轻量级日志打印输出源码
|      |  |-nb-iot NB-IoT 驱动和网络状态机源码
|      |
|      |-bsp 硬件驱动层源代码
|          |
|          |-stm32f40x 外部硬件驱动源代码
|          |-stm32l1xx 外部硬件驱动源代码
|
|-tools 项目开发相关工具软件
|
|-si4.si4project   SourceInsight4 源代码编辑查看软件工程
```

2.2.2 命名方法

规则 2.2.2：源代码目录和文件命名使用硬件模块名称或者程序功能名称。禁止使用汉字、拼音作为命名，名称中模块分隔使用‘_’下划线作为分隔符，禁止出现其他符号。命名需能够体现出具体内容和作用，禁止使用无意义或与实际内容不关联内容命名，尽量使用全小写名称。

示例 2.2.2：源代码目录和文件命名方法。

menu ： 目录，包含显示菜单应用程序目录； menu_edit.c ：文件，包含液晶显示菜单“文字编辑栏”的源代码

3 编码规范

3.1 程序命名

常用的程序命名方法主要有 3 种：匈牙利命名法(Microsoft)、骆驼命名法(TI)和帕斯卡命名法，目前程序开发大多采用几种命名方法组合的形式，如 Microsoft 大多采用匈牙利命名（变量命名）+帕斯卡命名（函数命名）组合，TI 采用骆驼命名(变量命名)+帕斯卡命名(函数命名)，linux 采用 K&R 风格，全小写字母+下划线’_’分隔方法。

规则 3.1.1：编写应用程序时，全局变量命名使用匈牙利命名法，首字母(作用域范围)后面需用下划线’_’分隔。函数命名使用帕斯卡命名法，使用每个单词首字母大写方式分隔。

规则 3.1.2：命名应遵循下列基本原则：

- 尽量简单、清晰、通俗；
- 使用英文命名，不使用中文或拼音命名；如确实使用拼音，注释必须说明缩写全称。
- 尽量选用通用词汇；
- 尽量使用完整单词或词组，避免使用简称；不常规简称，源码定义或文档中须有对应全称说明；
- 尽量准确表达其含义；
- 尽量避免同时使用易混淆的字母或数字，如 1 与 l、0 和 o；
- 禁止使用只靠字母大小写区分的多个名称；

规则 3.1.3：当名称太长时应使用缩写，缩写原则为：

- 应使用标准或常用的缩写（附录 A），如长度 length(len)，最大 maximum(max)；
- 同一项目内同一名称缩写应一致且规范，禁止各处缩写各不相同；
- 名称缩写可以使用单词首字母或前几个字母，如通道 Channel(ch)，操作系统 Operating System(OS)；
- 名称缩写可以去掉不在词头的元音字母，如屏幕 Screen（scrn）；
- 名称由多个单词组成，缩写可以使用有意义的简写或去掉无用的后缀，如失败次数计数 Count of Failure(FailCnt)；
- 项目中所有不在附录 A 中的名称缩写必须在项目相关文件内统一列出；

规则 3.1.4：全局变量命名采用匈牙利命名法，格式为**[作用域范围前缀]_基本类型＋名称，**其中，作用域范围前缀以小写字母表示且可选，基本类型以小写字母表示且必选。局部变量采用全小写字母或驼峰命名法。除常用的循环变量外，禁止使用单字母作为变量名。

示例 3.1.4：

常用前缀符

前缀符	含义	示例
g	全局变量	g_stSystem, g_cDevType， g_strDevName
s	静态变量	s_nCurCnt，s_strUserName，s_pSysTime
m	成员变量	m_nUserNum， m_stUserName
h	句柄变量	g_hnFileHandle，g_hnSocket，g_hTaskHandle
p	指针变量	g_psReadBuff，g_pstrRetStr，g_ppTarget
a	数组变量	g_anPorts，g_asSendBuffers，g_apWrkBuffs

常用基本类型符

前缀符	含义	示例
b	Bool	bOK，bQuit，bFind
c/ch	Char	cFlag，cDisplay
s/str	Char []	sUserName，strDevName
n/i	Int	nCnt，nPort
l	Long	lFileSize，lOffset，lCount
d	Double	dTotalHeat
f	Float	fTemperature
u	Unsigned	uiCnt
w	Word	与 unsigned int 等价
dw	Double word	与 unsigned long 等价
e/em	Enum	eKeyValue
st	Struct	stSystem

规则 3.1.5：常量和宏定义首个单词必须全大写，并且应尽量使用全部大写命名。同类宏定义必须有一个使用注释说明其含义和取值范围。宏函数可以用小写字母。

规则 3.1.6：函数命名应使用帕斯卡命名法，并且采用**[文件名/模块名]+动词+名词**形式。禁止在名称中包含除字母、数字和下划线’_’之外的符号，公共库函数可以采用全部小写+下划线”_”(K&R)命名。

建议 3.1.2：名称中最好不要包含数字和下划线’_’，使用单词首字母大写区分。

示例 3.1.5：KeyInit(void); LedOn(uint8_t led);

规则 3.1.6：函数参数命名参考变量命名方法，对指针类型变量名称开头应添加 p 指示。

示例 3.1.6：

```
bool IsSpaceStr(char *sInStr,   uint32_t   nMaxLen, uint32_t *plRetOffset);
```

3.2 程序注释

程序中增加注释的目的是帮助对程序的阅读理解，不宜太多和太少，一般代码注释量占源文件20%-30%左右，注释应当准确、易懂、简洁（参考 ST 标准库，以及 CycloneTCP 网络栈）。

规则 3.2.1：注释应当与源代码保持一致，应及时删除无关注释。注释标识符与前面内容和后面内容至少间隔 1 个空格。变量注释可与变量在同一行内，函数内语句和函数调用注释必须在对应语句上方，增加可读性，与其上面的语句代码至少间隔 1 行。

规则 3.2.2：头文件和源文件开头必须添加文件注释，主要包括版权声明、文件名称、功能说明、日期和版本，编写人和版本修订记录等。**注释必须遵循自动化注释文档规范**，可以方便将注释自动形成文档，如 doxygen。

示例 3.2.2：

```
/**
  ******************************************************************************
  *                    Copyright(C) 2018-2028 GDKY All Rights Reserved
  *
  * @file        template.c
  * @author      ZouZH
  * @version     V1.00
  * @date        22-April-2016
  * @brief       template.
  ******************************************************************************
  * @history
  *
```

规则 3.2.3：同一个项目组成员必须采用一致的注释方式。所有函数、宏定义、全局变量、结构体变量类型、枚举类型定义、自定义类型必须在变量和类型定义处增加必要的注释。

示例 3.2.3：

头文件注释示例

```
/**
  ******************************************************************************
  *               Copyright(C) 2015-2025 GDKY  All Rights Reserved
  *
  * @file     record_cfg.h
  * @author   ZouZH
  * @version  V1.00
  * @date     27-Nov-2015
  * @brief    数据存储通用平台相关配置和定义文件.
  ******************************************************************************
  */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __RECORD_CFG_H
#define __RECORD_CFG_H

#ifdef __cplusplus
 extern "C" {
#endif /* __cplusplus */

/* INCLUDES ------------------------------------------------------------------ */
#include <stdint.h>

/* TYPEDEFS ------------------------------------------------------------------ */

/**
 *  @brief 系统参数 24Bytes
 */
__packed typedef struct
{
  uint32_t version;          /*!< 版本号: xx.xx.xx.xx(BCD Code ) */
  uint8_t  bcc;              /*!< BCC 校验 */
  uint16_t crc;              /*!< CRC 校验 */

}SysParam_t;

/* MACROS  ------------------------------------------------------------------- */

/* CONSTANTS  ---------------------------------------------------------------- */

/**
 * 用户存储参数配置
 */
#define RECORD_TEMP_MAX_NBR       8  /* 热量表最大数目 */
#define RECORD_HW_MAX_NBR         3  /* 最多设备数, 0-255 */

/* GLOBAL VARIABLES ---------------------------------------------------------- */

/* GLOBAL FUNCTIONS ---------------------------------------------------------- */

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif /* __RECORD_CFG_H */

/*************************** END OF FILE ***************************************/
```

源文件注释示例

```
/**
  ******************************************************************************
  *               Copyright(C) 2015-2025 GDKY  All Rights Reserved
  *
  * @file     record.c
  * @author   ZouZH
  * @version  V1.01
  * @date     07-Dec-2015
  * @brief    数据存储通用平台应用层接口.
  ******************************************************************************
  * @history
  * 2015-11-04 V1.00 ZouZH 初次创建
  * 2015-12-07 V1.01 ZouZH 修正读取和写入数据项地址计算错误
  */


/* INCLUDES ------------------------------------------------------------------ */
#include "record.h"

/* TYPEDEFS ------------------------------------------------------------------ */

/* MACROS  ------------------------------------------------------------------- */

/* CONSTANTS  ---------------------------------------------------------------- */

/* GLOBAL VARIABLES ---------------------------------------------------------- */

/**
 * 系统参数
 */
SysParam_t g_stSysParam;

/* GLOBAL FUNCTIONS ---------------------------------------------------------- */

/* LOCAL VARIABLES ----------------------------------------------------------- */

/* LOCAL FUNCTIONS ----------------------------------------------------------- */

/**
 * @brief  读取数据
 *
 * @param  REC_DT_x 数据类型(1-REC_DT_MAX)
 * @param  REC_DI_x 数据条项(0-REC_DI_MAX)
 * @param  itemIdx  数据条项索引(0-65535)
 * @param  pvBuf    数据项缓存
 *
 * @retval 0 成功，其他失败
 */
RecordErr_t Record_Read(RecordDataType_t REC_DT_x, RecordDataItem_t REC_DI_x,
                        uint16_t itemIdx, void *pvBuf)
{
  RecordErr_t recErr = REC_OK;
  return recErr;
}
```

3.3 程序排版

规则 3.3.1：程序代码排版应采用缩进编写，每层一个制表位(TAB，缩进 2 个或 4 个空格)，同一个项目应用程序缩进风格应始终保持一致。

规则 3.3.2：每行只定义一个变量或者一个语句。每个函数定义结束之后都要加空行。在一个函数体内，逻辑上密切相关的语句之间不加空行，其他地方加空行分隔。

规则 3.3.3：太长的语句(>120 字符)或表达式应分成多行书写。语句应在逗号或者低优先级操作符处划分新行，操作符放在新行之首。划分的新行应适当缩进，使排版整齐，易读。

规则 3.3.4：相邻两个功能相对独立的代码块和变量说明之间应使用空行隔开。

规则 3.3.5：比较操作符，赋值操作符，算术操作符，逻辑操作符，位域操作符等双目运算符前后应加空格与操作数隔开。单目运算符如“!”、“~”，“++”、”--“等不应加空格与操作数隔开。逗号、分号后面应加空格，但前面不应加空格。函数名与括号间、结构或类与成员间、左括号后、右括号前、不应加空格。像“[] “、“.”、“->”这类操作符前后不加空格

规则 3.3.6：复杂的表达式应使用括号以避免二义性，不应过分依赖运算符优先级。位运算时应使用括号。宏定义为表达式时，最外部和每个变量必须使用括号。

建议 3.3.1：对结构体数据成员或者在含义上相关的数据统一赋值时和定义多个宏定义时，应保持等号‘=和赋值的数据对齐，增加可读性。

规则 3.3.7：if、for、do、while、case、switch、default 等语句（包含表达式）独占一行，且 if、for、do、while 等语句的执行部分尽量加括号{}。

3.4 程序规范

规则 3.4.1：禁止在头文件中定义变量。

规则 3.4.2：宏表达式的变量应使用括号，使用宏时应避免传递表达式给宏变量。禁止使用宏替换关键字。含有多个语句的宏应该被包含在一个 do-while 代码块里。

规则 3.4.3：单条语句构成的宏不应使用分号作为宏结束；多条语句构成宏时，应使用花括号括宏体。

规则 3.4.4：禁止使用与全局变量、外部变量同名的局部变量；禁止使用关键字、保留字、函数名作为变量名。

规则 3.4.5：函数参数为指针变量时，如果指针指向内容不作为返回值时，应加 const 修饰，防止程序意外修改。

规则 3.4.6：switch 参数必须为整型量，所有 switch 必须有 default 处理，case 语句不为空时，必须在末

尾使用 break。

规则 3.4.7：尽量避免使用 goto 语句，如果使用只允许 goto 流程为一个方向，禁止交叉跳转。

规则 3.4.8：用#include <filename.h>格式来引用标准库的头文件，用#include “filename.h”格式来引用非标准库的头文件。包含不同级别的头文件要用空行分割。

规则 3.4.9：自定义头文件中，要用#ifndef/#define/#endif 结构产生预处理块防止重复包含，并且与#else 和#endif 对应的宏名要标记完整。

规则 3.4.10：非外部使用的全局静态变量和静态函数应当显式用 static 修饰和声明。

建议 3.4.1：总在定义变量时指定其初始值。避免使用没有初始化的变量。

建议 3.4.2：尽可能少使用全局变量。多个任务访问同一个全局变量时，需注意临界区保护。中断服务程序或多任务使用的变量，应注意加上 volatile 修饰。

规则 3.4.3：if 语句中，整型变量用“==”或“！=”直接与 0 比较；不可将浮点变量用“==”或“！=”与任何数字比较；应当将指针变量用“==”或“！=”与 NULL 比较。

规则 3.4.4：需要对外公开的常量、变量和函数放在头文件中，不需要对外公开的放在定义文件的头部。

规则 3.4.5：函数的参数书写要完整，不能只写参数的类型而省略参数名字。如果函数没有参数，则用 void 填充。函数返回值的类型也同样必须显式声明。

规则 3.4.6：指针的数学运算只能用在指向数组或数组元素的指针上。指针减法只能用在指向同一数组中元素的指针上。>、>=、<、<= 不应用在指针类型上，除非指针指向同一数组。

建议 3.4.6：代码中的关键数值和常量数值尽量使用有意义的标识符代替，增加可读性和可维护性。

建议 3.4.7：函数功能应尽量单一。函数尽量不要定义静态变量，使函数不具有可重入性。

建议 3.4.8：能在循环体外完成的处理，应放置在循环体外。多重循环的，应将大循环放置在最内层，以减少 CPU 跨切循环层的次数。如果循环体内存在逻辑判断，并且循环次数很大，宜将逻辑判断移到循环体的外面。

建议 3.4.9：对于所有有物理含义的变量、常量，如果其命名不是充分自注释的，在声明时都必须加以注释，说明其物理含义。变量、常量、宏的注释应放在其上方相邻位置或右方。

建议 3.4.10：注意运算符的优先级，并用括号明确表达式的操作顺序，尽量避免使用默认优先级。

建议 3.4.11：程序模块间编程应层次化，尽量降低耦合性，增加程序功能模块的复用性和可移植性。

建议 3.4.12：程序结构的设计要尽量考虑向前兼容和以后的版本升级，并为某些未来可能的应用保留余地。

建议 3.4.13： 尽量检查函数输入参数的有效性和合法性，特别是指针和数据参数，并使用类似 ASSERT

方式在调试阶段查找错误。

建议 3.4.14：当定义变量或函数指针时，”*” 应靠近变量或函数名，而非靠近类型名。

建议 3.4.15：不可在 for 循环体内修改循环变量，防止 for 循环失去控制。for 语句的循环控制变量的取值采用“半开半闭区间”写法。

建议 3.4.16：定义几个有关联的常量的时候，应首选枚举类型。使用枚举类型应当显示标明起始值，并且加入前缀与类型，防止未知引用。

建议 3.4.17：函数的返回出口尽量只有一个，便于调试查错。

建议 3.4.18：编译过程中每条 Warning 都必须得到处理。在进行强制数据类型转换时要谨慎。

建议 3.4.19：代码中应尽量采用分级调试接口，但不能重复定义同类调试接口。尽量避免直接使用 printk 或 printf。

4 软件维护

软件版本管理贯穿整个产品开发的始末，它的主要作用是把各阶段工作的划分更加明确化，使整个软件开发过程明晰，便于软件开发过程中检查和回溯，开发经验的总结和积累，同时可以避免软件开发过程中版本的混乱。

4.1 版本管理

规则 4.1.1：统一使用 Git 作为嵌入式软件开发的源代码和技术文档的版本管理系统。必须提交到每个项目指定的服务器 URL，每个成员单独一个功能分支，由项目负责人负责审查合并。

建议 4.1.1：Windows 客户端软件可以使用 SourceTree 或者 TortoiseGit。

4.2 版本控制

规则 4.2.1：提交的版本号需遵循 Vxx.xx.xxx 规则，V 前面可以增加项目名称或编号，版本号分 3 节：主版本号+次版本号+内部版本号，每节之间使用小数点分隔。主版本号为具体项目代号，次版本号为基于项目的一个功能产品，内部版本号为产品开发过程中阶段性成果。

建议 4.2.2：项目组成员在自己的分支中可以在内部版本号之后增加一节，记录个人开发调试版本。

规则 4.2.2：要及时记录软件版本修改，源代码工作目录内必须使用单独文档记录关键功能修改；并且在相关源文件内增加修改说明。

规则 4.2.3：在本地工作目录下的代码（除当前调试的外）必须是最新的版本或者是基于某个已知版本。在提交之前必须要和库里面最新的代码作对比，正确进行提交。

规则 4.2.4：版本控制范围必须包括与软件开发相关的所有文档和代码，如硬件版本、软件功能说明、通信协议、软件框图、软件流程图、工作目录说明、程序修改记录、程序使用烧写等说明。

4.3 测试管理(BUG 提交、分配、解决等)

4.4 项目管理(工作量化、时间管理、项目归档等)

附录 A 常用名称缩写

缩写、略语	全称	含义	示例
Addr	Address	地址	DataAddr
Blk	Block	块	MemBlk
Chk	Check	检查/校验	ChkCRC16
Clr	Clear	清除	ClearScreen
Cnt	Count	计数	FailCnt
Ctrl	Control	控制	CtrlReg
Cur	Current	当前	CurLog
Del	Delete	删除	DelUser
Err	Error	错误	errReturn
Ext	Extension	扩展	ExtSpace
Grp	Group	组	UserGrp
ID	Identifier	标识符	CardID
Init	Initialize	初始化	LedInit
ISR	Interrupt Service Routine	中断服务程序	USART_ISR
Max	Maximum	最大	MaxValue
Min	Minimum	最小	MinValue
Mbox	Mailbox	邮箱	MboxCreate
Mem	Memory	存储	MemAllocate
Ms	Millisecond	毫秒	DelayMs
Msg	Message	消息	MsgSend
Num	Number	数量	ByteNum
Opt	Option	选项	OptType
Prio	Priority	优先级	TaskPrio
Prv	private	私有的	PrvVar
Rdy	Ready	准备	RdyRun
Req	Request	请求	ReqOpen
Sched	Scheduler	调度	TaskSched
Sec	Second	秒	SecCnt
Sem	Semaphore	信号量	SemKeyValue
Stat	State	状态	RunStat
Stk	Stack	堆栈	TaskStk
SW	Switch	开关	SwValue
Sys	System	系统	SysStatus
Tbl	Table	表	UserTbl
Tmout	Timeout	超时	RunTmout

参考文献

[1] ISO/IEC 9899:1990 Programming language-C C99 标准

[2] MISRA-C-:2004 Guidelines for the use of the C language in critical systems 关键系统 C 语言设计指导

[3] GBT 28169-2011 嵌入式软件 C 语言编码规范

[4] 华为软件编程规范

[5] The Art of Readable Code 编写可读代码的艺术

[6] 编程精粹 Microsoft 编写优质无错 C 程序秘诀

[7] linux 编程规范

[8] uCOS-II C 语言编程规约