**Table of Contents**

* [Назначение документа](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%BD%D0%B0%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0)
* [Область действия](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%BE%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C_%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%8F)
* [Цели соглашения](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%86%D0%B5%D0%BB%D0%B8_%D1%81%D0%BE%D0%B3%D0%BB%D0%B0%D1%88%D0%B5%D0%BD%D0%B8%D1%8F)
  + [Соглашения](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%81%D0%BE%D0%B3%D0%BB%D0%B0%D1%88%D0%B5%D0%BD%D0%B8%D1%8F)
  + [Базовые принципы](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%B1%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B5_%D0%BF%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF%D1%8B)
* [Исходные файлы](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B5_%D1%84%D0%B0%D0%B9%D0%BB%D1%8B)
  + [Ширина строки](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%88%D0%B8%D1%80%D0%B8%D0%BD%D0%B0_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8)
  + [Правила написания инклудов](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D0%BB%D0%B0_%D0%BD%D0%B0%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D1%8F_%D0%B8%D0%BD%D0%BA%D0%BB%D1%83%D0%B4%D0%BE%D0%B2)
  + [Заголовок файла](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BE%D0%BA_%D1%84%D0%B0%D0%B9%D0%BB%D0%B0)
  + [Порядок кода в файлах](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%B0_%D0%B2_%D1%84%D0%B0%D0%B9%D0%BB%D0%B0%D1%85)
* [Соглашение об именах](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%81%D0%BE%D0%B3%D0%BB%D0%B0%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BE%D0%B1_%D0%B8%D0%BC%D0%B5%D0%BD%D0%B0%D1%85)
  + [Имена файлов](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%B8%D0%BC%D0%B5%D0%BD%D0%B0_%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2)
  + [Имена папок](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%B8%D0%BC%D0%B5%D0%BD%D0%B0_%D0%BF%D0%B0%D0%BF%D0%BE%D0%BA)
  + [Макросы](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%BC%D0%B0%D0%BA%D1%80%D0%BE%D1%81%D1%8B)
  + [Все переменные, const-константы, экземпляры структур и объединений](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%B2%D1%81%D0%B5_%D0%BF%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_const-%D0%BA%D0%BE%D0%BD%D1%81%D1%82%D0%B0%D0%BD%D1%82%D1%8B_%D1%8D%D0%BA%D0%B7%D0%B5%D0%BC%D0%BF%D0%BB%D1%8F%D1%80%D1%8B_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80_%D0%B8_%D0%BE%D0%B1%D1%8A%D0%B5%D0%B4%D0%B8%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9)
  + [Все функции](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%B2%D1%81%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8)
  + [Пользовательские типы данных - typedef, типы структур и объединений](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B5_%D1%82%D0%B8%D0%BF%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_-_typedef_%D1%82%D0%B8%D0%BF%D1%8B_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80_%D0%B8_%D0%BE%D0%B1%D1%8A%D0%B5%D0%B4%D0%B8%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9)
  + [Рекомендованные сокращения в названиях](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%80%D0%B5%D0%BA%D0%BE%D0%BC%D0%B5%D0%BD%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%BE%D0%BA%D1%80%D0%B0%D1%89%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2_%D0%BD%D0%B0%D0%B7%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%D1%85)
* [Типы данных](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%82%D0%B8%D0%BF%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)
  + [Стандартные типы данных](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D1%8B%D0%B5_%D1%82%D0%B8%D0%BF%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)
* [Форматирование](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
  + [Строки кода](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8_%D0%BA%D0%BE%D0%B4%D0%B0)
  + [Объявление переменных](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%BE%D0%B1%D1%8A%D1%8F%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D1%85)
  + [Секции кода](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%81%D0%B5%D0%BA%D1%86%D0%B8%D0%B8_%D0%BA%D0%BE%D0%B4%D0%B0)
  + [Пробелы](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%BF%D1%80%D0%BE%D0%B1%D0%B5%D0%BB%D1%8B)
* [Конструкции](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%BA%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D0%B8)
  + [Циклы без тела](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%86%D0%B8%D0%BA%D0%BB%D1%8B_%D0%B1%D0%B5%D0%B7_%D1%82%D0%B5%D0%BB%D0%B0)
  + [Стиль расстановки скобок](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D1%81%D1%82%D0%B8%D0%BB%D1%8C_%D1%80%D0%B0%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B8_%D1%81%D0%BA%D0%BE%D0%B1%D0%BE%D0%BA)
* [Комментарии](http://wiki.dep111.rtc.local/code_conventions:embedded_c#%D0%BA%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8)

**Назначение документа**

Документ предназначен для стандартизации основных приемов оформления кода и использования элементов языка С.

**Область действия**

Действие документа распространяется на подразделения 111 ЦНИИ РТК, занимающиеся разработкой программного обеспечения. Данное соглашение следует в обязательном порядке применять при разработке новых модулей и программных продуктов. При поддержке старых продуктов следует соблюдать стиль ранее написанного кода.

**Цели соглашения**

Целями данного соглашения являются:

1. Повышение читаемости кода для облегчения поддержки и передачи кода между разработчиками;
2. Уменьшение вероятности ошибок при разработке.

**Соглашения**

**Запреты**

* Оператор GOTO
* Объявление переменных и структур как extern
* Использование malloc и массивов переменной длины
* Символ табуляции
* Стражи включения заголовочных файлов (#ifdef)
* Тела циклов и ветвлений без фигурных скобок
* Транслит в названиях
* Извращения и неуточненное поведение (триграфы, многосимвольные литералы и т.д.)

**Базовые принципы**

Фундаментальная цель этих стандартов состоит в том, чтобы поддерживать сопровождение кода. Это означает, что код должен быть читабельным, понятным, тестируемым и переносимым. Для достижения этого, необходимо следовать следующим принципам:

* Следуйте духу стандарта

Старайтесь делать похожие вещи одинаково, даже если это не оговорено в стандарте.

* Весь код должен быть написан в стандарте С99
* KISS

Стремитесь к простоте кода

* Пишите точно и ясно

Избегайте неявных и малоизвестных возможностей язык.

* Будьте последовательны

Используйте одни и те же правила везде, где можно.

* Избегайте сложных выражений

Выражения, включающие много подвыражений, трудно изучать и тестировать.

* Не используйте оператор GOTO
* Не используйте malloc и массивы переменной длины

И то и другое - вызов malloc. Обращения к куче занимают неизвестное время и в зависимости от положения звезд места в куче может неожиданно не хватить. И будет hardfault.

* Модификация старого кода

Всякий раз при изменении существующего кода пытайтесь его модифицировать, соблюдая соглашения, описанные в данном документе. Это создает уверенность, что код может быть расширен снова через какое-то время.

* Прочтите “Чистый код” Роберт Мартин
* Спрашивайте, если есть сомнения

**Исходные файлы**

**Ширина строки**

Вы должны стремится ограничивать ширину исходного текста СИ 100-а символами

**Отступы**

Символ табуляции к использованию запрещен. Выравнивание кода и текста в комментариях должно осуществляться только пробелами! Длина отступа для блока кода - 4 пробела. Препроцессорные директивы #if, #ifdef, #elif и #else тоже порождают отступ!

int test( void )

{

int a;

int b;

if ( a - b < 0 )

{

return a;

}

return b;

}

**Правила написания инклудов**

Одна папка с файлами образует модуль. Все файлы .с внутри модуля должны инклудить фалйы .h внутри модуля с помощью #include “file.h” - в кавычках и без путей. В таком случае модуль становится переносимым, его код не нужно менять, если вы назовете папку по-другому или передвините ее в дереве проекта.

Все системные инклуды (т.е. которые НЕ лежат в папке с проектом) инклудятся через угловые скобки - #include <stdint.h>

Все остальные инклуды (за исключением косячных стронних) инклудятся через кавычки с указанием пути, начиная от папки src - #include “UART/uart.h”. В проекте должно быть прописано как можно меньше путей (в идеале - только к src). Для SPL приходится делать исключение.

**Заголовок файла**

В начале каждого файла включайте блок комментариев на **русском** языке содержащий в себе краткое описание и назначение содержимого файла, имя и фамилию основного разработчика.

Пример:

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

*Описание*

*Это пример.*

*Разработчик: Пупкин Василий*

*Заметки*

*Компилируется только под C99.*

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

Текст заголовка в свободной форме.

**Порядок кода в файлах**

**Файл реализации (.с)**

* Заголовок файла
* Раздел #include. В каждый файл .с подключаются все необходимые инклуды. В соответствующий файл .h инклудятся только файлы, необходимые для подключения этого .h в других местах
* Раздел локальные макросы (все директивы #define)
* Локальные типы данных
* Локальные прототипы функций. С модификатором static.
* Локальные переменные файла. С модификатором static.
* Глобальные переменные для проекта и геттеры/сеттеры к ним. Переменные объявляются как static! Экземпляры структур здесь же.
* Тело глобальных функций.
* Тело локальных функций. С модификатором static.

Пример файла с разделителями - [template\_file.c](http://wiki.dep111.rtc.local/_media/code_conventions:template_file.c" \o "code_conventions:template_file.c (2.2 KB))

**Заголовочный файл (.h)**

* Заголовок файла - только для stand-alone заголовочных файлов (т.е. у которых нет своего файла.c)
* Глобальные макросы
* Глобальные типы данных
* Глобальные inline функции
* Глобальные прототипы функций.

Объявление переменных и экземпляров структур как extern **запрещено**. Используйте геттер-сеттер.

**Не допускайте многократного включения заголовочного файла**

Используйте конструкцию:

#pragma once

Не используйте стражи включений через ifdef.

Отделяйте основные разделы

Каждому разделу должен предшествовать блок комментариев, как показано ниже.

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

*Типы данных*

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

Пример файла - [template\_header.h](http://wiki.dep111.rtc.local/_media/code_conventions:template_header.h" \o "code_conventions:template_header.h (1.4 KB))

**Соглашение об именах**

**Имена файлов**

С маленькой буквы. С использованием нижнего подчеркивания.

test\_my\_mind.c

break\_my\_heart.h

**Имена папок**

С маленькой буквы. С использованием нижнего подчеркивания.

**Макросы**

БОЛЬШИМИ\_БУКВАМИ\_С\_ИСПОЛЬЗОВАНИЕМ\_НИЖНЕГО\_ПОДЧЕРКИВАНИЯ.

#define BUFFER\_SIZE 20

**Все переменные, const-константы, экземпляры структур и объединений**

Начинаются со строчной буквы. Каждое следующее слово в название – с заглавной (camelCase).

int bufferSize;

**Все функции**

Начинаются со строчной буквы. Каждое следующее слово в название – с заглавной.

static int isItReal( void );

**Глобальные функции**

**При создании глобальной для проекта функции, предваряйте ее имя названием модуля с нижним подчеркиванием**

void uart\_init(void)

Это аналог namespace'ов в С.

**Функции-"методы" для структур**

Если у вас есть структура, для которой вы хотите создать функции-методы (как для классов в с++), эти функции должны выглядеть следующим образом:

* Имя функции предваряется названием структуры через нижнее подчеркивание
* Первым параметром такой функции является указатель на экземпляр структуры, к которому применяется метод (явный this). Этот параметр всегда называется **self**.

typedef struct

{

...

} MySuperStruct;

void mySuperStruct\_gyperMethod(MySuperStruct \* self, ..);

**Пользовательские типы данных - typedef, типы структур и объединений**

Начинаются с заглавной буквы. Каждое следующее слово в название – с заглавной (CamelCase).

typedef uint32\_t MySuperType;

typedef struct

{

...

} MySuperStruct;

union MySuperUnion

{

...

};

**Рекомендованные сокращения в названиях**

|  |  |  |
| --- | --- | --- |
| buffer |  | buf |
| clock |  | clk |
| compare |  | cmp |
| configuration |  | cfg |
| context |  | ctx |
| current |  | cur |
| delay |  | dly |
| error |  | err |
| function |  | fnct |
| hexadecimal |  | hex |
| initialize |  | init |
| maximum |  | max |
| message |  | msg |
| minimum |  | min |
| pointer |  | ptr |
| previous |  | prev |
| priority |  | prio |
| timer |  | tmr |

**Типы данных**

**Стандартные типы данных**

Используются переопределения из стандартной библиотеки (файл stdint.h свой для каждой платформы/среды/компилятора). Они имеют вид: uint32\_t - беззнаковый целый размером 32 бита int8\_t - знаковый целый размеров 8 бит

**Тип bool**

Стандарт С99 вводит тип \_Bool. Напрямую этот тип использовать не следует. Следует подключать файл стандартной библиотеки stdbool.h, и использовать тип данных **bool**, принимающий значения **true** или **false**.

Это гарантирует идентичность bool в коде С99 и С++.

Компиляторы, которые не поддерживают стандарт С99 должны использоваться как можно реже и только в исключительных ситуациях.

**Область видимости типов данных**

Если тип данных должен использоваться только в файле реализации, то он должен быть объявлен в файле реализации. Если тип данных глобален, то он должен быть помещен в заголовочный файл модуля.

**Форматирование**

**Строки кода**

Используйте одну точку с запятой на одной строке (кроме for, lol)

Пример:

Правильно

firstStep = 0;

setPosition();

Неправильно

firstStep = 0; setPosition();

**Объявление переменных**

Не объявляйте несколько переменных на одной строке, даже если у них одинаковый тип. И особенно, если у них разный тип!

Правильно

int8\_t a = 0;

int8\_t b = 3;

Неправильно

int8\_t a, b;

Очень неправильно

int8\_t \* a, b; *// внезапно, тут у переменных тип будет разный*

**Секции кода**

Отделяйте логические секции кода пустыми строками или комментариями

**Пробелы**

**Между именем функции и круглыми скобками после него не должно быть пробелов при вызове функции**

dispInit();

**После открывающей скобки (квадратной и круглой) и перед закрывающей пробелы - по желанию**

setRobotPar( bar );

buffer[ 100 ];

**Один пробел необходим после каждой запятой для разделения аргументов**

setRobotPar( speed, position );

**Унарные операторы пишутся без пробела между ними и их операндом**

~bits;

++i;

\*ptr; *// разыменование*

sizeof( x );

**Унарный оператор \* при объявлении указателя пишется через два пробела**

int \* a;

**Бинарные операторы и тернарный оператор пишутся как минимум с одним пробелом между ними и операндом**

c1 = c2;

a + b = c;

a < b;

x << 8;

**Один пробел после точки с запятой**

for ( i = 0; i < 69; ++i )

{

}

**Ключевые слова if, else, while, for, switch, return сопровождаются одним пробелом**

if ( a > b )

while ( x > 0 )

for ( i = 0; i < 10; ++i )

switch ( x )

return ( y )

**Каждая функция состоящая более чем из 3-х строк снабжается заголовком на русском языке**

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

*Описание:*

*Аргументы:*

*Возврат:*

*Замечания:*

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

**Конструкции**

Использовать фигурные скобки – всегда

if ( a > b )

{

a = b;

}

**Циклы без тела**

Всегда снабжаются фигурными скобками (вид - на вкус пользователя)

while ( 1 )

{

}

или

while(1)

{;}

**Стиль расстановки скобок**

**Не используем стиль оформления K&R**

Не правильно

if ( ){

}

Правильно - стиль Allman

if ( )

{

}

**Комментарии**

Используются комментарии в стиле С++. Комментарии размещаются непосредственно над комментируемым участком кода или справа от него. Язык – русский.

*// массив для приема сообщения*

int8u buffer[];

int8s \* ptr; *//указатель на массив для передачи данных*