Для всех ценителей творческого подхода
при решении технических задач

Содержание

От издательства 13

Предисловие к первому изданию 14

Предисловие ко второму изданию 21

Предисловие к третьему изданию 24

Предисловие к четвертому изданию 28

Обновленные товарные знаки и благодарности 32

Аббревиатуры 34

Список литературы 39

Часть I. Языковые технологии для Real-Time C++ 41

Глава 1. Введение в Real-Time C++ 42

1.1. Программа LED 42
1.2. Синтаксис C++ 45
1.3. Типы классов 45
1.4. Члены 48
1.5. Объекты и экземпляры 50
1.6. #include 52
1.7. Пространства имен 52
1.8. Стандартная библиотека C++ 54
1.9. Подпрограмма main() 55
1.10. Доступ к низкоуровневым регистрам 56
1.11. Константа времени компиляции 57
Список литературы 58

Глава 2. Работа с программой на C++ в реальном времени на электронной плате 59

2.1. Целевое аппаратное оборудование 59
2.2. Создание и прошивка программы LED 60
2.3. Добавление таймера для визуального наблюдения переключения светодиода 64
2.4. Запуск и сброс программы LED 66
2.5. Определение и обработка ошибок и предупреждений 67
2.6. Достижение надлежащей эффективности 69
Список литературы 72

Глава 3. Простой старт для работы с Real-Time C++ 73
3.1. Объявление локальных переменных при использовании 74
3.2. Целочисленные типы фиксированной разрядности и пример с простыми числами 74
3.3. Тип bool 79
3.4. Организация с помощью пространства имен 80
3.5. Базовые классы 82
3.6. Базовые шаблоны 83
3.7. nullptr на замену NULL 85
3.8. Обобщенные константные выражения с constexpr 86
3.9. static_assert 87
3.10. Использование <limits> 88
3.11. std::array 88
3.12. Базовые алгоритмы STL 89
3.13. <numeric> 90
3.14. atomic_load() и atomic_store() 91
3.15. Разделители чисел 91
3.16. Двоичные литералы 92
3.17. Пользовательские литералы 93
3.18. Использование alignof и alignas 96
3.19. Спецификатор final 96
3.20. Псевдоним как альтернатива typedef 98
3.21. Ограничение диапазонов указателей с помощью 99
3.22. Генерация случайных чисел с помощью <random> 99
Список литературы 102

Глава 4. Объектно-ориентированные методики для микроконтроллеров 104
4.1. Объектно-ориентированное программирование 104
4.2. Объекты и инкапсуляция 109
4.3. Наследование 111
4.4. Динамический полиморфизм и подробный пример со светодиодами 112
4.5. Реальная нагрузка при динамическом полиморфизме 119
4.6. Чисто виртуальные и абстрактные 120
4.7. Соотношения классов 121
4.8. Некопируемые классы 123
4.9. Константные методы 124
4.10. Статические константные члены класса 128
4.11. Друзья класса 129
4.12. Недоступность виртуальных функций в конструкторе базового класса 131
Список литературы 134

Глава 5. Шаблоны C++ для микроконтроллеров 135
5.1. Шаблонные функции 135
5.2. Масштабируемость шаблонов, повторное использование кода и эффективность 137

5.3. Функции-члены шаблонов 139
5.4. Типы шаблонных классов 142
5.5. Параметры шаблона по умолчанию 143
5.6. Специализация шаблонов 144
5.7. Статический полиморфизм 146
5.8. Использование STL с микроконтроллерами 148
5.9. Шаблоны с переменным количеством аргументов 151
5.10. Метапрограммирование на основе шаблонов 153
5.11. Кортежи и обобщенное метапрограммирование 157
5.12. Шаблоны переменных 160
5.13. Шаблоны целочисленных последовательностей 163
Список литературы 166

Глава 6. Оптимизированное программирование на C++ для микроконтроллеров 167

6.1. Использование настроек оптимизации компилятора 167
6.2. Информация о производительности микроконтроллера 171
6.3. Информация о сложности алгоритма 172
6.4. Использование ассемблерных листингов 174
6.5. Использование файлов карты 175
6.6. Суть искажения и восстановления имен 176
6.7. Когда стоит прибегать к ассемблеру, а когда не стоит 178
6.8. Использование осмысленных комментариев 180
6.9. Упрощение кода с помощью typedef и псевдонимов 180
6.10. Использование нативных целочисленных типов 183
6.11. Использование умножения степени двойки 185
6.12. Возможное замещение умножения сдвигом и сложением 186
6.13. Использование преимуществ аппаратного обеспечения 187
6.14. Возможность использования ПЗУ 189
6.15. Минимизация фрейма прерывания 193
6.16. Использование настраиваемого управления памятью 196
6.17. Последовательное использование STL 197
6.18. Использование лямбда-выражений 199
6.19. Использование шаблонов и масштабируемость 200
6.20. Использование метапрограммирования для развертывания циклов 201
6.21. Возможные издержки на информацию о типах во время выполнения (RTTI) 202
Список литературы 204

Часть II. Компоненты для Real-Time C++ 205

Глава 7. Доступ к регистрам микроконтроллера 206

7.1. Определение постоянных адресов регистров 206
7.2. Использование шаблонов для доступа к регистрам 208
7.3. Типовые шаблоны для доступа к регистрам 211
7.4. Структуры с побитовым отображением 213
Список литературы 216

Глава 8. Правильный старт 217
8.1. Код запуска 217
8.2. Инициализация ОЗУ 220
8.3. Инициализация статических конструкторов 222
8.4. Взаимосвязь между компоновщиком и предзапуском 224
8.5. Принципы статической инициализации 226
8.6. Избегайте использования неинициализированных объектов 228
8.7. Переход к main() и отсутствие возможности возврата 230
8.8. Что происходит после main()? 231
Список литературы 231

Глава 9. Низкоуровневые аппаратные драйверы на C++ 232
9.1. Шаблон класса драйвера контактов порта ввода-вывода 232
9.2. Программирование прерываний в C++ 234
9.3. Реализация системного тика 239
9.4. Класс шаблона программного ШИМ 242
9.5. Класс драйверов последовательного интерфейса SPI™ 246
9.6. Мониторинг загрузки центрального процессора 249
9.7. Управление семисегментным дисплеем 252
9.8. Анимация RGB-светодиода 259
Список литературы 264

Глава 10. Настройка функций управления памятью 266
10.1. Особенности работы с динамической памятью 266
10.2. Использование Placement-new 268
10.3. Аллокаторы и контейнеры STL 269
10.4. Стандартный аллокатор 270
10.5. Создание специализированного ring_allocator 271
10.6. Использование ring_allocator и других аллокаторов 274
10.7. Распознавание и устранение ограничений по памяти 276
10.8. Память вне микросхемы и вычисления 100 001 знака числа π 278
10.9. Использование большого объема ОЗУ с одноплатным компьютером на базе Arm® 292
Список литературы 303

Глава 11. Многозадачность в C++ 305
11.1. Многозадачные планировщики 305
11.2. Синхронизация задач 307
11.3. Блок управления задачами 308
11.4. Список задач 310
11.5. Планировщик 311
11.6. Расширенная многозадачность 312
11.7. Вытесняющая многозадачность 314
11.8. Библиотека поддержки потоков C++ 316
Список литературы 318

Часть III. Математика и утилиты для Real-Time C++ 319

Глава 12. Математика с плавающей запятой 320

12.1. Арифметика с плавающей запятой 320
12.2. Математические константы 323
12.3. Элементарные функции 325
12.4. Специальные функции 326
12.5. Математика с комплексными значениями 335
12.6. Оценка функций при компиляции с помощью constexpr 339
12.7. Обобщенное числовое программирование 342
Список литературы 349

Глава 13. Математика с фиксированной запятой 351

13.1. Типы данных с фиксированной запятой 351
13.2. Масштабируемый шаблонный класс с фиксированной запятой 354
13.3. Использование класса fixed_point 358
13.4. Элементарные трансцендентные функции с фиксированной запятой 360
13.5. Специализация std::numeric_limits 370
Список литературы 372

Глава 14. Высокопроизводительные цифровые фильтры 373

14.1. Фильтр с плавающей запятой 1-го порядка 373
14.2. Целочисленный фильтр 1-го порядка 376
14.3. Целочисленные FIR-фильтры ***N***-го порядка 380
14.4. Некоторые проверенные примеры фильтров 384
Список литературы 389

Глава 15. Утилиты C++ 390

15.1. Структура nothing 390
15.2. Класс noncopyable 392
15.3. Шаблонный класс timer 394
15.4. Линейная интерполяция 397
15.5. Шаблонный класс circular_buffer 400
15.6. Библиотека Boost 403
Список литературы 404

Глава 16. Расширение стандартной библиотеки C++ и STL 405

16.1. Определение пользовательского контейнера dynamic_array 405
16.2. Реализация и использование dynamic_array 407
16.3. Создание элементов библиотеки C++ при отсутствии таковых 411
16.4. Рекомендации по реализации элементов библиотеки C++ и STL 411
16.5. Поддержка now() для часов <chrono> с высоким разрешением 419
16.6. Шаблоны расширенных комплексных чисел 421
16.7. Встраиваемый класс больших целых чисел 424
16.8. Настройка <random> 427

16.9. Автономная реализация 438
Список литературы 439

Глава 17. Использование кода на языке C в C++ 441
17.1. Доступ к коду на языке C в C++ 441
17.2. Использование имеющейся библиотеки CRC на языке C 442
17.3. Обертывание библиотеки CRC на основе C с помощью классов C++ 444
17.4. Возвращаясь к исследованиям эффективности и оптимизации 446
Список литературы 448

Глава 18. Дополнительная литература 449
18.1. Список дополнительной литературы 449
Список литературы 451

Приложение A. Учебник по C++ реального времени 454
A.1. Операторы приведения C++ 454
A.2. Синтаксис унифицированной инициализации 455
A.3. Перегрузка 457
A.4. Подтверждение во время компиляции 458
A.5. Числовые ограничения 458
A.6. Контейнеры STL 462
A.7. Итераторы STL 464
A.8. Алгоритмы STL 466
A.9. Лямбда-выражения 470
A.10. Списки инициализаторов 471
A.11. Вывод типов и объявление типов с помощью auto и decltype 472
A.12. Диапазонный цикл for(:) 473
A.13. Кортеж 474
A.14. Регулярные выражения 477
A.15. Библиотека <type_traits> 479
A.16. Использование std::any и std::variant 482
A.17. Структурированные объявления связей 484
A.18. Трехстороннее сравнение 485
Список литературы 486

Приложение B. Надежная среда C++ для работы в реальном времени 487
B.1. Решение проблем, связанных с применением C++ в режиме реального времени 487
B.2. Программная архитектура 490
B.3. Определение и соблюдение ограничений времени выполнения 491
Список литературы 492

Приложение C. Сборка и установка кросс-компиляторов GNU GCC 493
C.1. Требования GCC 493
C.2. Начало работы 494

C.3. Сборка GMP 495
C.4. Сборка MPFR 496
C.5. Сборка MPC 497
C.6. Сборка PPL 498
C.7. Сборка ISL 499
C.8. Сборка бинарных утилит для кросс-компилятора 499
C.9. Сборка кросс-компилятора 500
C.10. Использование кросс-компилятора 501
Список литературы 502

Приложение D. Сборка схемы микроконтроллера 504
D.1. Принципиальная схема 504
D.2. Сборка схемы на макетной плате 506
Список литературы 507

Словарь 508

Предметный указатель 509

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте *www.dmkpress.com*, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу *dmkpress@gmail.com*; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу *http://dmkpress.com/authors/publish_book/* или напишите в издательство по адресу *dmkpress@gmail.com*.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу *dmkpress@gmail.com*. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты *dmkpress@gmail.com*.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Предисловие к первому изданию

Эта книга представляет собой практическое руководство по программированию встроенных микроконтроллерных систем реального времени на языке C++. Язык C++ обладает мощными объектно-ориентированными и шаблонными функциями, которые позволяют оптимизировать проектирование и переносимость программного обеспечения, одновременно снижая сложность кода и риск возникновения ошибок. В то же время C++ отлично подходит для компиляции высококачественного оригинального кода. Это уникальное и эффективное сочетание делает C++ идеальным языком для программирования микроконтроллерных систем, которые должны быть компактными, высокопроизводительными и надежными в плане требований безопасности.

Целевой аудиторией этой книги могут быть любители, студенты и профессионалы, интересующиеся программированием на C++ в режиме реального времени. Читателю необходимо быть знакомым с C или другим языком программирования и, в идеальном случае, иметь некоторый опыт использования микроконтроллерной электроники, а также иметь представление о проблемах производительности и габаритов, характерных для программирования встроенных систем.

Об этой книге

Эта книга представляет собой междисциплинарное издание, которое охватывает широкий спектр различных тем. Реальные примеры сочетаются с краткими описаниями, что помогает сформировать интуитивно понятную и простую методологию программирования микроконтроллеров на C++. Основное внимание уделяется вопросам эффективности, а многочисленные примеры подкрепляются измерениями производительности в реальном времени и анализом размеров, которые позволяют количественно оценить истинную ценность кода вплоть до последнего байта и микросекунды.

Все главы книги посвящены использованию C++ в минималистическом варианте, без лишних сложностей и без применения каких-либо библиотек, кроме указанных в стандарте языка. Такой подход значительно упрощает переносимость.

Книга состоит из трех частей и нескольких приложений. Все три части взаимосвязаны и объединены общей целью – предоставить читателю набор последовательных и эффективных методов C++, которые можно использовать при работе с широким спектром встроенных микроконтроллеров.

Часть I закладывает основу для работы с C++ в режиме реального времени, освещая языковые технологии. Темы включают в себя начало работы с C++ в режиме реального времени, объектно-ориентированные методы, программирование с использованием шаблонов и оптимизацию. Наиболее практичными являются первые три главы, которые призваны способствовать повышению квалификации специалистов по C++ для программирования в режиме реального времени. Глава 6 играет уникальную и важную роль, поскольку полностью посвящена методам оптимизации применительно к программированию микроконтроллеров на C++.

Часть II содержит подробное описание различных компонентов C++, широко используемых в программировании микроконтроллеров. Эти компоненты можно использовать в том виде, в котором они представлены, или адаптировать для других проектов. В этой части книги используются некоторые из наиболее мощных элементов языка C++, такие как типы классов, шаблоны и STL, позволяющие разрабатывать компоненты для доступа к регистрам микроконтроллеров, низкоуровневые драйверы, настраиваемое управление памятью, встроенные контейнеры, многозадачность и т. д.

В части III описываются математические методы и общие утилиты, которые можно использовать для решения повторяющихся проблем в C++ реального времени.

Приложения включают учебник по языку C++, информацию о среде разработки C++ реального времени, а также инструкции по созданию кросс-компиляторов GNU GCC и схему микроконтроллера.

C++ является богатейшим языком с огромным количеством функций и тонкостей, описание которых может заполнить собой несколько книжных полок. Однако в данной книге основное внимание уделяется именно использованию C++ в среде микроконтроллеров реального времени. В связи с этим учебные материалы по языку C++ изложены лаконично, а информация об аппаратном обеспечении микроконтроллеров и компиляторах включена только в той мере, в какой она необходима для описания примеров. В главе 18 приведен список рекомендуемой дополнительной литературы для тех, кто ищет вспомогательную информацию по C++, стандартной библиотеке C++ и STL, проектированию программного обеспечения, рекомендациям по кодированию на C++, набору инструментов для встроенных систем и аппаратному обеспечению микроконтроллеров.

При выражении физических величин используется система единиц MKS (метр, килограмм, секунда).

Сопутствующий код, цели и инструменты

Сопутствующий код охватывает три вводных проекта и один справочный проект. Вводные проекты посвящены различным аспектам материалов, представленных в главах 1 и 2. Справочный проект имеет более широкий охват и включает в себя множество методов из всех глав.

Сопутствующий код доступен по адресу:

http://github.com/ckormanyos/real-time-cpp.

Техники C++, описанные в этой книге, ориентированы непосредственно на работу с микроконтроллерами *малого* и *среднего* размера. В данном случае малый и средний размер соответствуют следующим приблизительным диапазонам размера и производительности:

- 4 Кбайта … 1 Мбайт программного кода;
- 256 байт … 128 Кбайт ОЗУ;
- 8-битный … 32-битный процессор;
- частота процессора 8–200 МГц.

Однако большинство описываемых в этой книге методов являются масштабируемыми. Благодаря этому их можно с одинаковым успехом применять как в устройствах большего, так и меньшего размера, включая персональные компьютеры и рабочие станции. В частности, их можно использовать в случаях, когда приложение жестко ограничено по производительности и габаритам.

В качестве основного целевого устройства для тестирования и проверки примеров кода в этой книге использовался популярный 8-битный микроконтроллер с тактовой частотой 16 МГц. Некоторые тесты также проводились с использованием хорошо известного 32-битного микроконтроллера с тактовой частотой 24 МГц. И 8-битный, и 32-битный микроконтроллеры были специально подобраны для изучения методов C++ в рамках широкого диапазона производительности микроконтроллеров.

Все примеры и тесты C++ в этой книге, а также сопутствующий код были скомпилированы с помощью GNU GCC версий 4.6.2 и 4.7.0. Некоторые примеры и тесты также были скомпилированы с помощью других компиляторов для ПК.

В тексте используется самая последняя спецификация C++11 в ISO/IEC 14882:2011. На момент написания этой книги спецификация C++11 была совершенно новой. Появление C++11 значительно повысило эффективность и удобство использования C++. Это существенно повлияет на программирование на C++. Поэтому подготовленный читатель наверняка захочет быть в курсе лучших практик C++11 по мере их развития сообществом разработчиков.

Примечания по стилю кодирования

Во всех примерах этой книги и в сопутствующем коде используется единый стиль кодирования.

Примеры кода набраны с помощью моноширинного шрифта. Ключевые слова языка C++ и встроенные типы набраны тем же шрифтом, но выделены полужирным начертанием. Например:

```
constexpr int version = 7;
```

Как правило, имена всех символов, таких как переменные, типы классов, члены и подпрограммы, записываются строчными буквами. Для разделения слов и сокращений в именах используется один подчеркивающий символ (_). Например, переменная `system-tick`, выраженная в этом стиле, показана в примере кода ниже.

```
unsigned long system_tick;
```

Использование префиксов, суффиксов или аббревиатур для передачи информации о типе в имени, иногда называемое *венгерской нотацией* (Hungarian notation), в этой книге не применяется. Лишние префиксы, суффиксы и аббревиатуры в венгерской нотации могут затруднять определение имени символа, а без них имена символов становятся более интуитивно понятными и наглядными. Например:

```
std::uint16_t name_of_a_symbol;
```

Имена для использования в общедоступных областях предпочтительно используются в длинных и описательных вариантах, а не в коротких и сокращенных. В данном случае ясность выражения предпочтительнее лаконичности. Однако символы, используемые для локальных параметров подпрограмм или прикладных подробностей реализации с очевидным значением, зачастую обозначаются лаконичными или сокращенными именами.

Такой стиль именования используется, например, в приведенной ниже глобальной подпрограмме. Она возвращает значение **`float`**, равное квадрату евклидова расстояния от начала координат точки в двумерном декартовом пространстве $\mathbb{R}^2$.

```
float squared_euclidean_distance(const float& x,
                                 const float& y)
{
  return (x * x) + (y * y);
}
```

В языке C++ активно используются ссылки, поскольку они могут быть эффективны для работы с небольшими микроконтроллерами. Рассмотрим 8-битный микроконтроллер. Копирование параметров подпрограмм или их перемещение в стек для данных шириной более 8 разрядов может оказаться

трудоемким процессом. Этот объем работы можно потенциально сократить с помощью ссылок. Например, в предыдущем примере кода параметры подпрограммы с плавающей запятой `x` и `y`, каждый из которых имеет ширину 4 байта, были переданы в подпрограмму по ссылке (т. е. **`const float`**&).

Типы целых чисел фиксированного размера, определенные в пространстве имен `std` стандартной библиотеки C++, такие как `std::uint8_t`, `std::uint16_t`, `std::uint32_t` и т. п., преимущественно используются взамен простых встроенных типов, таких как **`char`**, **`short`**, **`int`** и т. д. Это позволяет улучшить наглядность и портируемость кода. Ниже приведен пример беззнакового ответа на вход в систему, который содержит ровно 8 бит.

```
std::uint8_t login_response;
```

Примеры кода обычно включают один или несколько заголовков стандартной библиотеки C++, такие как `<algorithm>`, `<array>`, `<cstdint>`, `<limits>`, `<tuple>`, `<vector>` и т. д. Как правило, примеры кода, которым требуются заголовки библиотек, не включают в себя соответствующие заголовки в явном виде.

Например, приведенное выше объявление `login_response` фактически требует `<cstdint>` для определения `std::uint8_t`. Однако файл библиотеки при этом не включается. Как правило, в примерах кода основное внимание уделяется самому коду, а не включению заголовков библиотек.

Несложно угадать или запомнить, например, что `std::array` находится в `<array>`, а `std::vector` находится в `<vector>`. Однако сложнее угадать или запомнить, что `std::size_t` находится в `<cstddef>`, а `std::accumulate()` находится в `<numeric>`. Однако с помощью онлайн-справки и других ресурсов, а также после небольшой практики определение частей стандартной библиотеки, которые можно найти в тех или иных заголовках, становится рутинной задачей.

В случаях, когда включению заголовочного файла уделяется особое внимание, соответствующие строки **`#include`** могут быть прописаны в явном виде. Например:

```
#include <cstdint>
std::uint8_t login_response;
```

Пространства имен используются достаточно часто. Тем не менее, как правило, директива **`using`** не используется для вставки символов из пространств имен в глобальное пространство имен. Это означает, что все пространство имен должно быть введено с именем символа в нем. Это, опять же, способствует однозначности, а не краткости.

Например, в приведенном ниже беззнаковом 16-битном счетчике используется тип из пространства имен `std`. Поскольку директива "**`using namespace std`**" не применяется, имя пространства имен (`std`) включено в тип в явном виде.

```
std::uint16_t counter;
```

Суффиксы обычно добавляются к литеральным константам. Когда суффикс добавляется к литеральной константе, его опциональным регистром является верхний. Например:

```
constexpr float pi = 3.14159265358979323846F;

constexpr std::uint8_t login_key = 0x55U;
```

На стиль кодирования значительное влияние оказали некоторые устоявшиеся правила написания кода на C++. Однако ради лаконичности и наглядности не все правила соблюдались до конца.

Одним из явных проявлений влияния правил написания кода является тщательное использование приведения типов в стиле C++ при преобразовании встроенных типов. Например, в следующем коде выполняется приведение типа **float** к типу беззнакового целого числа в явном виде.

```
float f = 3.14159265358979323846F;

std::uint8_t u = static_cast<std::uint8_t>(f);
```

Хотя явные приведения типов, подобные этому, не всегда являются обязательными, они помогают устранить неоднозначность и исключить возможные ошибки интерпретации, вызванные преобразованием целых чисел.

Еще одним проявлением влияния рекомендаций по кодированию на программный код является упорядочение членов класса в соответствии с их уровнем доступа в классе. Например, приведенный ниже класс `communication` представляет собой базовый класс в иерархии объектов связи. Члены в определении класса упорядочены в соответствии с уровнем доступа. Так, в частности:

```
class communication
{
public:
  virtual ~communication();

  virtual bool send(const std::uint8_t) const;
  virtual bool recv(std::uint8_t&);

protected:
  communication();

private:
  bool recv_ready;
  std::uint8_t recv_buffer;
};
```

Время от времени используются макросы (макроопределения) препроцессора в стиле C. Макросы препроцессора пишутся полностью в верхнем регистре. Слова в именах макросов препроцессора разделяются подчеркиваниями. Например, приведенный ниже макрос препроцессора `MAKE_WORD()` создает беззнаковое 16-разрядное слово из двух беззнаковых 8-разрядных составляющих.

```
#define MAKE_WORD(lo, hi) \
(uint16_t) (((uint16_t) (hi) << 8) | (lo))
```

Благодарности

Прежде всего мне хотелось бы поблагодарить свою жену и дочь за поддержку в написании этой книги, а также за создание спокойной и заботливой атмосферы, в которой мне было комфортно работать. Спасибо вам за вашу поддержку и время. Я вам очень благодарен.

Я также хотел бы выразить признательность своим родным, друзьям и коллегам, которых слишком много, чтобы перечислить всех, но которые внесли свой вклад в этот проект своими инновационными идеями, поддержкой, дружбой и сотрудничеством.

Спасибо членам комитета по стандартам C++, Boost, волонтерам GCC и всем разработчикам в динамичном сообществе C++ и встроенных систем. Благодаря вашим усилиям, зачастую совершенно безвозмездным, C++ вырос до беспрецедентного уровня выразительности, благодаря чему объектно-ориентированное и обобщенное программирование стало более эффективным и простым, чем когда-либо.

Совместная работа со Springer Verlag оказалась очень приятным опытом. Я благодарю своего редактора, который первым оценил достоинства этой работы и поддерживал меня на протяжении всего процесса создания книги. Я также благодарю команду редакторов и всех сотрудников Springer Verlag за их профессионализм и квалифицированную поддержку.

- Схема целевого оборудования, описанная в этой книге и изображенная в главе 2 и приложении D, была разработана и собрана на беспаечной макетной плате Кристофером Корманьосом.
- Фотографии целевого оборудования, описанного в этой книге и изображенного в главе 2 и приложении D, были сделаны Кристофером Корманьосом.

Кристофер Корманьос
Ройтлинген, Германия
Сиэтл, Вашингтон, США
Сентябрь 2012 г.