Contents

Part I Language Technologies for Real-Time C++

1 Getting Started with Real-Time C++ 3
- 1.1 The LED Program 3
- 1.2 The Syntax of C++ 6
- 1.3 Class Types 6
- 1.4 Members 9
- 1.5 Objects and Instances 11
- 1.6 `#include` 12
- 1.7 Namespaces 13
- 1.8 C++ Standard Library 15
- 1.9 The `main()` Subroutine 15
- 1.10 Low-Level Register Access 16
- 1.11 Compile-Time Constant 17

References 18

2 Working with a Real-Time C++ Program on a Board 19
- 2.1 The Target Hardware 19
- 2.2 Build and Flash the LED Program 20
- 2.3 Adding Timing for Visible LED Toggling 23
- 2.4 Run and Reset the LED Program 25
- 2.5 Recognizing and Handling Errors and Warnings 25
- 2.6 Reaching the Right Efficiency 28

References 30

3 An Easy Jump-Start in Real-Time C++ 33
- 3.1 Declare Locals when Used 33
- 3.2 Fixed-Size Integer Types 34
- 3.3 The `bool` Type 35
- 3.4 Organization with Namespaces 36
- 3.5 Basic Classes 37
- 3.6 Basic Templates 38

3.7 `nullptr` Replaces `NULL` 39
3.8 Generalized Constant Expressions with `constexpr` 40
3.9 `static_assert()` 41
3.10 Using `<limits>` 41
3.11 `std::array` 42
3.12 Basic STL Algorithms 43
3.13 `<numeric>` 43
3.14 `atomic_load()` and `atomic_store()` 44

4 Object-Oriented Techniques for Microcontrollers 47
4.1 Object Oriented Programming 47
4.2 Objects and Encapsulation 52
4.3 Inheritance 53
4.4 Dynamic Polymorphism 55
4.5 The Real Overhead of Dynamic Polymorphism 56
4.6 Pure Virtual and Abstract 57
4.7 Class Relationships 58
4.8 Non-copyable Classes 60
4.9 Constant Methods 61
4.10 Static Constant Integral Members 65
4.11 Class Friends 65
4.12 Virtual Is Unavailable in the Base Class Constructor 67
References 70

5 C++ Templates for Microcontrollers 71
5.1 Template Functions 71
5.2 Template Scalability, Code Re-use and Efficiency 73
5.3 Template Member Functions 75
5.4 Template Class Types 78
5.5 Template Default Parameters 79
5.6 Template Specialization 80
5.7 Static Polymorphism 82
5.8 Using the STL with Microcontrollers 84
5.9 Variadic Templates 86
5.10 Template Metaprogramming 89
5.11 Tuples and Generic Metaprogramming 91
References 95

6 Optimized C++ Programming for Microcontrollers 97
6.1 Use Compiler Optimization Settings 97
6.2 Know the Microcontroller's Performance 99
6.3 Know an Algorithm's Complexity 100
6.4 Use Assembly Listings 102
6.5 Use Map Files 102
6.6 Understand Name Mangling and De-mangling 103
6.7 Know When to Use Assembly and When Not To 105
6.8 Use Comments Sparingly 106

6.9 Simplify Code with `typedef` 107
6.10 Use Native Integer Types 109
6.11 Use Scaling with Powers of Two 111
6.12 Replace Multiply with Shift-and-Add 112
6.13 Consider Advantageous Hardware Dimensioning 112
6.14 Consider ROM-Ability 114
6.15 Minimize the Interrupt Frame 116
6.16 Use Custom Memory Management 118
6.17 Use the STL Consistently 119
6.18 Use Lambda Expressions 120
6.19 Use Templates and Scalability 122
6.20 Use Metaprogramming to Unroll Loops 122
References 123

Part II Components for Real-Time C++

7 Accessing Microcontroller Registers 127
7.1 Defining Constant Register Addresses 127
7.2 Using Templates for Register Access 129
7.3 Generic Templates for Register Access 131
7.4 Bit-Mapped Structures 134
Reference 136

8 The Right Start 137
8.1 The Startup Code 137
8.2 Initializing RAM 139
8.3 Initializing the Static Constructors 141
8.4 The Connection Between the Linker and Startup 143
8.5 Understand Static Initialization Rules 145
8.6 Avoid Using Uninitialized Objects 146
8.7 Jump to `main()` and Never `return` 148
8.8 When in `main()`, What Comes Next? 149
References 150

9 Low-Level Hardware Drivers in C++ 151
9.1 An I/O Port Pin Driver Template Class 151
9.2 Programming Interrupts in C++ 154
9.3 Implementing a System-Tick 158
9.4 A Software PWM Template Class 161
9.5 A Serial SPI™ Driver Class 165
9.6 CPU-Load Monitors 169
References 171

10 Custom Memory Management 173
10.1 Dynamic Memory Considerations 173
10.2 Using Placement-`new` 174

10.3 Allocators and STL Containers 176
10.4 The Standard Allocator 177
10.5 Writing a Specialized `ring_allocator` 178
10.6 Using `ring_allocator` and Other Allocators 181
10.7 Recognizing and Handling Memory Limitations 183
References 185

11 C++ Multitasking 187
11.1 Multitasking Schedulers 187
11.2 Task Timing 188
11.3 The Task Control Block 189
11.4 The Task List 192
11.5 The Scheduler 193
11.6 Extended Multitasking 194
11.7 Preemptive Multitasking 196
11.8 The C++ Thread Support Library 197
References 198

Part III Mathematics and Utilities for Real-Time C++

12 Floating-Point Mathematics 201
12.1 Floating-Point Arithmetic 201
12.2 Mathematical Constants 204
12.3 Elementary Functions 205
12.4 Special Functions 206
12.5 Complex-Valued Mathematics 212
12.6 Compile-Time Evaluation of Functions with `constexpr` 215
12.7 Generic Numeric Programming 218
References 223

13 Fixed-Point Mathematics 225
13.1 Fixed-Point Data Types 225
13.2 A Scalable Fixed-Point Template Class 227
13.3 Using the `fixed_point` Class 231
13.4 Fixed-Point Elementary Transcendental Functions 234
13.5 A Specialization of `std::numeric_Limits` 244
References 246

14 High-Performance Digital Filters 247
14.1 A Floating-Point Order–1 Filter 247
14.2 An Order–1 Integer Filter 250
14.3 Order–*N* Integer FIR Filters 254
14.4 Some Worked-Out Filter Examples 258
References 263

15 C++ Utilities 265
15.1 The `nothing` Structure 265
15.2 The `noncopyable` Class 268

15.3 A Template `timer` Class 270
15.4 Linear Interpolation 272
15.5 A Circular Buffer Template Class 276
15.6 The Boost Library 278
References 279

16 Extending the C++ Standard Library and the STL 281
16.1 Defining the Custom `dynamic_array` Container 281
16.2 Implementing and Using `dynamic_array` 283
16.3 Writing Parts of the C++ Library if None Is Available 287
16.4 Implementation Notes for Parts of the C++ Library and STL 287
16.5 Providing `now()` for `<chrono>`'s High-Resolution Clock 293
Reference 295

17 Additional Reading 297
17.1 Literature List 297
References 298

Appendices

A A Tutorial for Real-Time C++ 303
A.1 C++ Cast Operators 303
A.2 Uniform Initialization Syntax 304
A.3 Overloading 306
A.4 Compile-Time Assert 306
A.5 Numeric Limits 307
A.6 STL Containers 310
A.7 STL Iterators 312
A.8 STL Algorithms 314
A.9 Lambda Expressions 318
A.10 Initializer Lists 319
A.11 Type Inference 320
A.12 Range-Based `for(:)` 321
A.13 Tuple 321
A.14 Regular Expressions 323
References 325

B A Robust Real-Time C++ Environment 327
B.1 Addressing the Challenges of Real-Time C++ 327
B.2 Software Architecture 329
B.3 Establishing and Adhering to Runtime Limits 330
References 331

C Building and Installing GNU GCC Cross Compilers 333
C.1 The GCC Prerequisites 333
C.2 Getting Started 334

C.3 Building GMP 334
C.4 Building MPFR 335
C.5 Building MPC 336
C.6 Building PPL 337
C.7 Building the Binary Utilities for the Cross Compiler 338
C.8 Building the Cross Compiler 339
C.9 Using the Cross Compiler 341
References 342

D Building a Microcontroller Circuit 343
D.1 The Circuit Schematic 343
D.2 Assembling the Circuit on a Breadboard 344
References 346

Glossary 347

Index 349