

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	24
2.4	Run and Reset the LED Program	26
2.5	Recognizing and Handling Errors and Warnings	27
2.6	Reaching the Right Efficiency	29
	References	31
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	36
3.4	Organization with Namespaces	37

3.5	Basic Classes	38
3.6	Basic Templates	39
3.7	nullptr Replaces NULL	40
3.8	Generalized Constant Expressions with constexpr	41
3.9	static_assert	42
3.10	Using <limits>	43
3.11	std::array	43
3.12	Basic STL Algorithms	44
3.13	<numeric>	45
3.14	atomic_load() and atomic_store()	46
3.15	Digit Separators	46
3.16	Binary Literals	47
3.17	User-Defined Literals	48
	Reference	49
4	Object-Oriented Techniques for Microcontrollers	51
4.1	Object Oriented Programming	51
4.2	Objects and Encapsulation	56
4.3	Inheritance	57
4.4	Dynamic Polymorphism	59
4.5	The Real Overhead of Dynamic Polymorphism	60
4.6	Pure Virtual and Abstract	61
4.7	Class Relationships	62
4.8	Non-Copyable Classes	64
4.9	Constant Methods	65
4.10	Static Constant Integral Members	68
4.11	Class Friends	69
4.12	Virtual Is Unavailable in the Base Class Constructor	71
	References	74
5	C++ Templates for Microcontrollers	75
5.1	Template Functions	75
5.2	Template Scalability, Code Re-Use and Efficiency	77
5.3	Template Member Functions	79
5.4	Template Class Types	82
5.5	Template Default Parameters	83
5.6	Template Specialization	85
5.7	Static Polymorphism	86
5.8	Using the STL with Microcontrollers	89
5.9	Variadic Templates	91
5.10	Template Metaprogramming	93
5.11	Tuples and Generic Metaprogramming	96
5.12	Variable Templates	99
	References	101

6	Optimized C++ Programming for Microcontrollers	103
6.1	Use Compiler Optimization Settings	103
6.2	Know the Microcontroller's Performance	106
6.3	Know an Algorithm's Complexity	108
6.4	Use Assembly Listings	109
6.5	Use Map Files	110
6.6	Understand Name Mangling and De-Mangling	111
6.7	Know When to Use Assembly and When Not to	112
6.8	Use Comments Sparingly	114
6.9	Simplify Code with typedef	114
6.10	Use Native Integer Types	116
6.11	Use Scaling with Powers of Two	118
6.12	Potentially Replace Multiply with Shift-and-Add	119
6.13	Consider Advantageous Hardware Dimensioning	120
6.14	Consider ROM-Ability	122
6.15	Minimize the Interrupt Frame	123
6.16	Use Custom Memory Management	126
6.17	Use the STL Consistently	126
6.18	Use Lambda Expressions	128
6.19	Use Templates and Scalability	129
6.20	Use Metaprogramming to Unroll Loops	130
	References	130

Part II Components for Real-Time C++

7	Accessing Microcontroller Registers	133
7.1	Defining Constant Register Addresses	133
7.2	Using Templates for Register Access	135
7.3	Generic Templates for Register Access	137
7.4	Bit-Mapped Structures	140
	Reference	142
8	The Right Start	143
8.1	The Startup Code	143
8.2	Initializing RAM	145
8.3	Initializing the Static Constructors	147
8.4	The Connection between the Linker and Startup	149
8.5	Understand Static Initialization Rules	151
8.6	Avoid Using Uninitialized Objects	152
8.7	Jump to main() and Never return	154
8.8	When in main(), What Comes Next?	155
	References	156

9	Low-Level Hardware Drivers in C++	157
9.1	An I/O Port Pin Driver Template Class	157
9.2	Programming Interrupts in C++	160
9.3	Implementing a System-Tick	164
9.4	A Software PWM Template Class	167
9.5	A Serial SPI™ Driver Class	171
9.6	CPU-Load Monitors	176
9.7	Controlling a Seven-Segment Display	178
	References	183
10	Custom Memory Management	185
10.1	Dynamic Memory Considerations	185
10.2	Using Placement-new	186
10.3	Allocators and STL Containers	188
10.4	The Standard Allocator	189
10.5	Writing a Specialized <code>ring_allocator</code>	190
10.6	Using <code>ring_allocator</code> and Other Allocators	193
10.7	Recognizing and Handling Memory Limitations	195
	References	197
11	C++ Multitasking	199
11.1	Multitasking Schedulers	199
11.2	Task Timing	201
11.3	The Task Control Block	202
11.4	The Task List	204
11.5	The Scheduler	205
11.6	Extended Multitasking	206
11.7	Preemptive Multitasking	208
11.8	The C++ Thread Support Library	209
	References	210

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

12	Floating-Point Mathematics	213
12.1	Floating-Point Arithmetic	213
12.2	Mathematical Constants	216
12.3	Elementary Functions	218
12.4	Special Functions	219
12.5	Complex-Valued Mathematics	225
12.6	Compile-Time Evaluation of Functions with <code>constexpr</code>	228
12.7	Generic Numeric Programming	231
	References	238

13 Fixed-Point Mathematics	241
13.1 Fixed-Point Data Types	241
13.2 A Scalable Fixed-Point Template Class	244
13.3 Using the <code>fixed_point</code> Class	247
13.4 Fixed-Point Elementary Transcendental Functions	250
13.5 A Specialization of <code>std::numeric_limits</code>	260
References	262
14 High-Performance Digital Filters	263
14.1 A Floating-Point Order-1 Filter	263
14.2 An Order-1 Integer Filter	266
14.3 Order- <i>N</i> Integer FIR Filters	269
14.4 Some Worked-Out Filter Examples	274
References	279
15 C++ Utilities	281
15.1 The <code>nothing</code> Structure	281
15.2 The <code>noncopyable</code> Class	284
15.3 A Template <code>timer</code> Class	286
15.4 Linear Interpolation	289
15.5 A <code>circular_buffer</code> Template Class	292
15.6 The Boost Library	296
References	297
16 Extending the C++ Standard Library and the STL	299
16.1 Defining the Custom <code>dynamic_array</code> Container	299
16.2 Implementing and Using <code>dynamic_array</code>	301
16.3 Writing Parts of the C++ Library if None Is Available	305
16.4 Implementation Notes for Parts of the C++ Library and STL	306
16.5 Providing <code>now()</code> for <code><chrono></code> 's High-Resolution Clock	312
Reference	313
17 Additional Reading	315
17.1 Literature List	315
References	316
Appendix A: A Tutorial for Real-Time C++	319
Appendix B: A Robust Real-Time C++ Environment	345
Appendix C: Building and Installing GNU GCC Cross Compilers	351

Appendix D: Building a Microcontroller Circuit	363
Glossary	367
Index	369