

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	10
1.5	Objects and Instances	12
1.6	#include	13
1.7	Namespaces	14
1.8	C++ Standard Library	16
1.9	The <code>main()</code> Subroutine	16
1.10	Low-Level Register Access	17
1.11	Compile-Time Constant	18
	References	19
2	Working with a Real-Time C++ Program on a Board	21
2.1	The Target Hardware	21
2.2	Build and Flash the LED Program	22
2.3	Adding Timing for Visible LED Toggling	26
2.4	Run and Reset the LED Program	28
2.5	Recognizing and Handling Errors and Warnings	28
2.6	Reaching the Right Efficiency	30
	References	33
3	An Easy Jump Start in Real-Time C++	35
3.1	Declare Locals When Used	35
3.2	Fixed-Size Integer Types and Prime Number Example	36
3.3	The <code>bool</code> Type	42
3.4	Organization with Namespaces	42
3.5	Basic Classes	45
3.6	Basic Templates	46

3.7	nullptr Replaces NULL	48
3.8	Generalized Constant Expressions with constexpr	49
3.9	static_assert	51
3.10	Using <code><limits></code>	51
3.11	<code>std::array</code>	52
3.12	Basic STL Algorithms.....	53
3.13	<code><numeric></code>	54
3.14	<code>atomic_load()</code> and <code>atomic_store()</code>	55
3.15	Digit Separators.....	55
3.16	Binary Literals	56
3.17	User-Defined Literals.....	57
3.18	Using alignof and alignas	60
3.19	The Specifier final	61
3.20	Alias as an Alternative to typedef	62
3.21	Delimiting Pointer Ranges with <code></code>	64
3.22	Generating Random Numbers with <code><random></code>	64
	References.....	68
4	Object-Oriented Techniques for Microcontrollers	69
4.1	Object Oriented Programming	69
4.2	Objects and Encapsulation	74
4.3	Inheritance	76
4.4	Dynamic Polymorphism and a Detailed LED Example	77
4.5	The Real Overhead of Dynamic Polymorphism	84
4.6	Pure Virtual and Abstract.....	85
4.7	Class Relationships	86
4.8	Non-copyable Classes	88
4.9	Constant Methods	89
4.10	Static Constant Integral Members	93
4.11	Class Friends.....	95
4.12	Virtual Is Unavailable in the Base Class Constructor	97
	References.....	100
5	C++ Templates for Microcontrollers	101
5.1	Template Functions	101
5.2	Template Scalability, Code Re-Use and Efficiency	103
5.3	Template Member Functions.....	106
5.4	Template Class Types.....	109
5.5	Template Default Parameters.....	110
5.6	Template Specialization	111
5.7	Static Polymorphism	113
5.8	Using the STL with Microcontrollers.....	116
5.9	Variadic Templates	118
5.10	Template Metaprogramming	121
5.11	Tuples and Generic Metaprogramming.....	125
5.12	Variable Templates	129

5.13	Template Integer Sequences	132
	References	136
6	Optimized C++ Programming for Microcontrollers	137
6.1	Use Compiler Optimization Settings	137
6.2	Know the Microcontroller's Performance	141
6.3	Know an Algorithm's Complexity	142
6.4	Use Assembly Listings	144
6.5	Use Map Files	145
6.6	Understand Name Mangling and De-mangling	145
6.7	Know When to Use Assembly and When Not to	147
6.8	Use Sensible Comments	149
6.9	Simplify Code with typedef and Alias	149
6.10	Use Native Integer Types	152
6.11	Use Scaling with Powers of Two	154
6.12	Potentially Replace Multiply with Shift-and-Add	156
6.13	Consider Advantageous Hardware Dimensioning	156
6.14	Consider ROM-ability	158
6.15	Minimize the Interrupt Frame	163
6.16	Use Custom Memory Management	166
6.17	Use the STL Consistently	167
6.18	Use Lambda Expressions	169
6.19	Use Templates and Scalability	170
6.20	Use Metaprogramming to Unroll Loops	171
6.21	Potential Costs of Runtime Type Information (RTTI)	171
	References	174

Part II Components for Real-Time C++

7	Accessing Microcontroller Registers	177
7.1	Defining Constant Register Addresses	177
7.2	Using Templates for Register Access	179
7.3	Generic Templates for Register Access	182
7.4	Bit-Mapped Structures	185
	Reference	188
8	The Right Start	189
8.1	The Startup Code	189
8.2	Initializing RAM	192
8.3	Initializing the Static Constructors	194
8.4	The Connection Between the Linker and Startup	196
8.5	Understand Static Initialization Rules	198
8.6	Avoid Using Uninitialized Objects	200

8.7	Jump to <code>main()</code> and Never return	202
8.8	When in <code>main()</code> , What Comes Next?	203
	References	204
9	Low-Level Hardware Drivers in C++	205
9.1	An I/O Port Pin Driver Template Class	205
9.2	Programming Interrupts in C++	207
9.3	Implementing a System Tick	212
9.4	A Software PWM Template Class	215
9.5	A Serial SPI™ Driver Class	219
9.6	CPU-Load Monitors	222
9.7	Controlling a Seven-Segment Display	225
9.8	Animating an RGB LED	232
	References	237
10	Custom Memory Management	239
10.1	Dynamic Memory Considerations	239
10.2	Using Placement- new	241
10.3	Allocators and STL Containers	242
10.4	The Standard Allocator	243
10.5	Writing a Specialized <code>ring_allocator</code>	244
10.6	Using <code>ring_allocator</code> and Other Allocators	247
10.7	Recognizing and Handling Memory Limitations	249
10.8	Off-Chip Memory and Computing 100,001 Digits of π	251
10.9	Using Ample RAM on Arm®-Based Single-Board Computer	266
	References	276
11	C++ Multitasking	279
11.1	Multitasking Schedulers	279
11.2	Task Timing	281
11.3	The Task Control Block	282
11.4	The Task List	284
11.5	The Scheduler	285
11.6	Extended Multitasking	286
11.7	Preemptive Multitasking	288
11.8	The C++ Thread Support Library	291
	References	292
 Part III Mathematics and Utilities for Real-Time C++		
12	Floating-Point Mathematics	295
12.1	Floating-Point Arithmetic	295
12.2	Mathematical Constants	298
12.3	Elementary Functions	300
12.4	Special Functions	302

12.5	Complex-Valued Mathematics	312
12.6	Compile-Time Evaluation of Functions with constexpr	316
12.7	Generic Numeric Programming	320
	References	327
13	Fixed-Point Mathematics	329
13.1	Fixed-Point Data Types	329
13.2	A Scalable Fixed-Point Template Class	332
13.3	Using the <code>fixed_point</code> Class	336
13.4	Fixed-Point Elementary Transcendental Functions	338
13.5	A Specialization of <code>std::numeric_limits</code>	349
	References	351
14	High-Performance Digital Filters	353
14.1	A Floating-Point Order-1 Filter	353
14.2	An Order-1 Integer Filter	356
14.3	Order- <i>N</i> Integer FIR Filters	360
14.4	Some Worked-Out Filter Examples	365
	References	369
15	C++ Utilities	371
15.1	The <code>nothing</code> Structure	371
15.2	The <code>noncopyable</code> Class	374
15.3	A Template <code>timer</code> Class	376
15.4	Linear Interpolation	379
15.5	A <code>circular_buffer</code> Template Class	382
15.6	The Boost Library	386
	References	387
16	Extending the C++ Standard Library and the STL	389
16.1	Defining the Custom <code>dynamic_array</code> Container	389
16.2	Implementing and Using <code>dynamic_array</code>	392
16.3	Writing Parts of the C++ Library if <code>None</code> Is Available	396
16.4	Implementation Notes for Parts of the C++ Library and STL	396
16.5	Providing <code>now()</code> for <code><chrono></code> 's High-Resolution Clock	405
16.6	Extended-Complex Number Templates	407
16.7	An Embeddable Big Integer Class	410
16.8	Customizing <code><random></code>	414
16.9	Freestanding Implementation	424
	References	425
17	Using C-Language Code in C++	427
17.1	Accessing C Language Code in C++	427
17.2	An Existing C-Language CRC Library	428
17.3	Wrapping the C-Based CRC Library with C++ Classes	430
17.4	Return to Investigations of Efficiency and Optimization	433
	References	434

18	Additional Reading	435
18.1	Literature List	435
	References	437
A	A Tutorial for Real-Time C++	439
A.1	C++ Cast Operators	439
A.2	Uniform Initialization Syntax	440
A.3	Overloading	443
A.4	Compile-Time Assert	443
A.5	Numeric Limits	444
A.6	STL Containers	448
A.7	STL Iterators	450
A.8	STL Algorithms	453
A.9	Lambda Expressions	457
A.10	Initializer Lists	458
A.11	Type Inference and Type Declaration with auto and decltype	460
A.12	Range-Based for (:)	462
A.13	Tuple	462
A.14	Regular Expressions	466
A.15	The <code><type_traits></code> Library	468
A.16	Using <code>std::any</code> and <code>std::variant</code>	471
A.17	Structured Binding Declarations	474
A.18	Three-Way Comparison	475
	References	475
B	A Robust Real-Time C++ Environment	477
B.1	Addressing the Challenges of Real-Time C++	477
B.2	Software Architecture	479
B.3	Establishing and Adhering to Runtime Limits	480
	References	481
C	Building and Installing GNU GCC Cross Compilers	483
C.1	The GCC Prerequisites	483
C.2	Getting Started	484
C.3	Building GMP	485
C.4	Building MPFR	486
C.5	Building MPC	486
C.6	Building PPL	487
C.7	Building ISL	488
C.8	Building the Binary Utilities for the Cross Compiler	488
C.9	Building the Cross Compiler	490
C.10	Using the Cross Compiler	491
	References	492

D Building a Microcontroller Circuit 493

 D.1 The Circuit Schematic 493

 D.2 Assembling the Circuit on a Breadboard 495

 References 496

Glossary 497

Index 499