

Contents

<i>Foreword</i>	<i>xxi</i>
<i>Preface</i>	<i>xxiii</i>
<i>Acknowledgment</i>	<i>xxv</i>
<i>Terms and Abbreviations</i>	<i>xxvii</i>
<i>Conventions</i>	<i>xxix</i>
<i>References</i>	<i>xxxi</i>
Chapter 1: Introduction	1
1.1 Welcome to the World of Embedded Processors	1
1.1.1 Where Are the Processors Used?	1
1.1.2 Processor, CPU, Core, Microprocessor, and All These Names	2
1.1.3 Programming on Embedded Systems	3
1.1.4 What Type of Skills Do I Need to Start Learning Microcontroller Programming?	4
1.2 Understanding Different Types of Processors	4
1.2.1 Why We Need Various Types of Processors	4
1.2.2 Overview of the ARM Processor Families	5
1.2.3 Blurring the Boundaries	8
1.2.4 ARM Cortex-M Processor Series	8
1.2.5 Quick Glance on the ARM Cortex-M0 and Cortex-M0+ Processor	12
1.2.6 From Cortex-M0 Processor to Cortex-M0+ Processor	13
1.2.7 Applications of the Cortex-M0 and Cortex-M0+ Processor	17
1.3 What Is Inside a Microcontroller	18
1.3.1 Typical Elements Inside a Microcontroller	18
1.3.2 Characteristics of Processors for Microcontroller Applications	20
1.3.3 Silicon Technologies	22
1.4 There is Something About ARM®	22
1.4.1 Do ARM Make Chips?	22
1.4.2 What Else Does ARM Make?	23
1.4.3 Why Do Not Chip Vendors Do Their Own Processor Designs?	23
1.4.4 What is Special About the ARM Ecosystem?	24

1.5 Resources on Using ARM [®] Processors and ARM Microcontrollers	25
1.5.1 On the ARM Web Pages	25
1.5.2 Resources from Microcontroller Vendors	26
1.5.3 Resources from Tool Vendors	28
1.5.4 Other Resources	28
Chapter 2: Technical Overview	29
2.1 What are the Cortex [®] -M0 and Cortex-M0+ Processors?	29
2.2 Block Diagrams	31
2.3 Typical Systems	34
2.4 What Is ARMv6-M Architecture?	37
2.5 Software Portability Between Cortex [®] -M Processors	38
2.6 The Advantages of the ARM [®] Cortex [®] -M0 and Cortex-M0+ Processor	40
2.6.1 Low Power and Energy Efficiency	40
2.6.2 High Code Density	41
2.6.3 Low Interrupt Latency and Deterministic Behavior	42
2.6.4 Ease of Use	42
2.6.5 System-Level Features and OS Support Features	42
2.6.6 Comprehensive Debug Features	43
2.6.7 Configurability, Flexibility, and Scalability	43
2.6.8 Software Portability and Reusability	44
2.6.9 Wide Range of Product Choices	44
2.6.10 Wide Ecosystem Support	45
2.7 Applications of the Cortex [®] -M0 and Cortex-M0+ Processors	45
2.7.1 Microcontrollers	45
2.7.2 Sensors	46
2.7.3 Sensor Hubs	47
2.7.4 Power Management IC	47
2.7.5 ASSPs, ASICs	47
2.7.6 Subsystems in System on Chips	47
2.8 Why Using a 32-Bit Processor for Microcontroller Applications?	48
2.8.1 Performance	48
2.8.2 Code Density	49
2.8.3 Other Benefits of ARM Architectures	52
2.8.4 Software Reusability	53
Chapter 3: Introduction to Embedded Software Development	55
3.1 Welcome to Embedded System Programming	55
3.2 Some Basic Concepts	55
3.2.1 Reset	55
3.2.2 Clocks	56
3.2.3 Voltage Level	57
3.2.4 Inputs and Outputs	57
3.2.5 Introduction to Embedded Software Program Flows	58
3.2.6 Programming Language Choices	63

3.3 Introduction to ARM [®] Cortex [®] -M Programming	64
3.3.1 C Programming—Data Types	64
3.3.2 Accessing Peripherals in C	65
3.3.3 What Is Inside a Program Image?	69
3.3.4 Data in SRAM	71
3.3.5 What Happens When a Microcontroller Starts?	73
3.4 Software Development Flow	74
3.5 Cortex [®] Microcontroller Software Interface Standard	78
3.5.1 Introduction of CMSIS	78
3.5.2 What Are Standardized in CMSIS-CORE?	80
3.5.3 Organization of the CMSIS-CORE	81
3.5.4 Using CMSIS-CORE	81
3.5.5 Benefits of CMSIS	83
3.6 Other Information on Software Development	85
Chapter 4: Architecture	87
4.1 Overview of ARMv6-M Architecture	87
4.1.1 What Architecture Means	87
4.1.2 Background of the ARMv6-M Architecture	87
4.2 Programmer's Model	89
4.2.1 Operation Modes and States	89
4.2.2 Registers and Special Registers	90
4.2.3 Behaviors of the APSR	96
4.3 Memory System	97
4.3.1 Overview	97
4.3.2 Single Cycle I/O Interface	99
4.3.3 Memory Protection Unit	99
4.4 Stack Memory Operations	100
4.5 Exceptions and Interrupts	102
4.6 Nested Vectored Interrupt Controller	104
4.6.1 Flexible Interrupt Management	104
4.6.2 Nested Interrupt Support	104
4.6.3 Vectored Exception Entry	104
4.6.4 Interrupt Masking	105
4.7 System Control Block	105
4.8 Debug System	105
4.9 Program Image and Start-up Sequence	106
Chapter 5: Instruction Set	109
5.1 What Is Instruction Set	109
5.2 Background of ARM [®] and Thumb [®] Instruction Set	110
5.3 Assembly Basics	113
5.3.1 Quick Glance at Assembly Syntax	113
5.3.2 Use of a Suffix	117
5.3.3 Unified Assembler Language (UAL)	118

5.4 Instruction List.....	119
5.4.1 Moving Data within the Processor	120
5.4.2 Memory Accesses.....	122
5.4.3 Stack Memory Accesses.....	126
5.4.4 Arithmetic Operations	127
5.4.5 Logic Operations	131
5.4.6 Shift and Rotate Operations.....	132
5.4.7 Extend and Reverse Ordering Operations	135
5.4.8 Program Flow Control.....	137
5.4.9 Memory Barrier Instructions.....	139
5.4.10 Exception-Related Instructions	141
5.4.11 Sleep Mode Feature-Related Instructions.....	142
5.4.12 Other Instructions	143
5.5 Pseudo Instructions.....	144
Chapter 6: Instruction Usage Examples.....	147
6.1 Overview	147
6.2 Program Control	147
6.2.1 If-then-else	147
6.2.2 Loop	148
6.2.3 More on the Branch Instructions	148
6.2.4 Typical Usages of Branch Conditions	148
6.2.5 Function Calls and Function Returns	150
6.2.6 Branch Table.....	151
6.3 Data Accesses	153
6.3.1 Simple Data Accesses	153
6.3.2 Example of Using Memory Access Instruction	154
6.4 Data Type Conversion	157
6.4.1 Conversion of Data Size.....	157
6.4.2 Endian Conversion.....	158
6.5 Data Processing	158
6.5.1 64-Bit/128-Bit Add.....	158
6.5.2 64-Bit/128-Bit Sub	159
6.5.3 Integer Divide	159
6.5.4 Unsigned Integer Square Root	161
6.5.5 Bit and Bit Field Computations.....	162
Chapter 7: Memory System.....	165
7.1 Memory Systems in Microcontrollers.....	165
7.2 Bus Systems in the Cortex [®] -M0 and Cortex-M0+ Processors.....	166
7.3 Memory Map	167
7.3.1 Overview.....	167
7.3.2 Code Region (0x00000000–0x1FFFFFFF).....	168
7.3.3 SRAM Region (0x20000000–0x3FFFFFFF).....	168
7.3.4 Peripheral Region (0x40000000–0x5FFFFFFF).....	169

7.3.5 RAM Region (0x60000000–0x9FFFFFFF)	169
7.3.6 Device Region (0xA0000000–0xDFFFFFFF)	169
7.3.7 Internal Private Peripheral Bus (0xE0000000–0xE00FFFFFFF)	169
7.3.8 Reserved Memory Space (0xE0100000–0xFFFFFFFF).....	170
7.3.9 System Level Design.....	170
7.4 Program Memory, Boot Loader, and Memory Remapping	170
7.4.1 Program Memory and Boot Loader.....	170
7.4.2 Memory Remap	172
7.5 Data Memory	173
7.6 Little Endian and Big Endian Support	174
7.7 Data Type.....	175
7.8 Memory Attributes and Memory Access Permission.....	177
7.9 Effect of Hardware Behavior to Programming	180
7.9.1 Data Alignment.....	180
7.9.2 Access to Invalid Addresses.....	181
7.9.3 Use of Multiple Load and Store Instructions.....	181
7.9.4 Wait States	182
Chapter 8: Exceptions and Interrupts	185
8.1 What are Exceptions and Interrupts?.....	185
8.2 Exception Types on the Cortex [®] -M0 and Cortex-M0+ Processors.....	187
8.2.1 Overview	187
8.2.2 Non-Maskable Interrupt	187
8.2.3 HardFault	188
8.2.4 SVCcall (Supervisor Call).....	188
8.2.5 Pendable Service Call.....	188
8.2.6 System Tick Timer	188
8.2.7 Interrupts	189
8.3 Brief Overview of the NVIC.....	189
8.4 Definition of Exception Priority Levels.....	190
8.5 Vector Table	192
8.6 Exception Sequence Overview.....	194
8.6.1 Acceptance of Exception Request	194
8.6.2 Stacking and Unstacking.....	194
8.6.3 Exception Return Instruction	195
8.6.4 Tail Chaining	196
8.6.5 Late Arrival.....	196
8.7 EXC_RETURN.....	197
8.8 NVIC Control Registers for Interrupt Control	200
8.8.1 Overview of NVIC Control Registers	200
8.8.2 Interrupt Enable and Clear Enable	200
8.8.3 Interrupt Pending Set and Clear Register.....	202
8.8.4 Interrupt Priority Level.....	204
8.9 Exception Masking Register (PRIMASK).....	206

8.10	Interrupt Inputs and Pending Behavior.....	207
8.10.1	Simple Interrupt Process	207
8.10.2	Simple Pulse Interrupt Handling.....	208
8.10.3	Canceling of Interrupt Pending Status Before the Interrupt Is Serviced.....	209
8.10.4	Clearing of Pending Status While Peripheral Still Asserting IRQ.....	209
8.10.5	IRQ Remains High When ISR Completed.....	210
8.10.6	Multiple IRQ Pulses Before Entering ISR	210
8.10.7	IRQ Pulse During ISR Execution.....	210
8.10.8	IRQ Assertion for a Disabled Interrupt.....	211
8.11	Details of Exception Entry Sequence	212
8.11.1	Stacking.....	212
8.11.2	Vector Fetch and Update PC.....	214
8.11.3	Registers Update.....	214
8.12	Details of Exception Exit Sequence	215
8.12.1	Unstacking of Registers.....	215
8.12.2	Fetch and Execute From Return Address.....	215
8.13	Interrupt Latency	215
Chapter 9: System Control and Low-Power Features.....		219
9.1	Brief Introduction of System Control Registers.....	219
9.2	Registers in the SCBs.....	220
9.2.1	List of Registers in the SCB	220
9.2.2	CPU ID Base Register.....	220
9.2.3	Control Registers for System Exceptions Management.....	221
9.2.4	Vector Table Offset Register	223
9.2.5	Application Interrupt and Reset Control Register.....	224
9.2.6	System Control Register.....	225
9.2.7	Configuration and Control Register	225
9.2.8	System Handler Control and State Register	226
9.3	Using the Self-Reset Feature.....	226
9.4	Using the Vector Table Relocation Feature.....	228
9.5	Low-Power Features	230
9.5.1	Overview	230
9.5.2	Sleep Modes.....	231
9.5.3	Wait-for-Event and Wait-for-Interrupt	232
9.5.4	Wake-up Conditions	235
9.5.5	Sleep-On-Exit Feature	237
9.5.6	Wake-up Interrupt Controller	239
Chapter 10: Operating System Support Features.....		243
10.1	Overview of OS Support Features	243
10.2	Introduction to Operating Systems in Embedded World	243
10.3	The SysTick Timer	245
10.3.1	SysTick Registers	246

10.3.2	Setting up SysTick.....	248
10.3.3	Using SysTick Timer for Timing Measurement.....	250
10.3.4	Using SysTick Timer in Single Shot Mode	251
10.4	Process Stack and PSP	252
10.5	SVCALL Exception	256
10.6	PendSV.....	258
10.7	Advanced Topics: Using SVC and PendSV in Programming	259
10.7.1	Using the SVC Exception	260
10.7.2	Using the PendSV Exception.....	265
10.8	Advanced Topics: Context Switching in Action	267
Chapter 11: Fault Handling.....		279
11.1	Fault Exception Overview	279
11.2	What Can Cause a Fault?.....	279
11.3	Analyze a Fault.....	280
11.4	Accidental Switching to ARM® State	282
11.5	Error Handling in Real Applications	283
11.6	Error Handling During Software Development.....	283
11.7	Lockup	286
11.7.1	Causes of Lockup	286
11.7.2	What Happens During a Lockup?	288
11.8	Preventing Lockup.....	288
11.9	Comparison with Fault Handling in ARMv7-M Architecture.....	289
Chapter 12: Memory Protection Unit.....		291
12.1	What is MPU?	291
12.2	MPU Use Cases	292
12.3	Technical Introduction	294
12.4	MPU Registers	294
12.4.1	MPU Type Register	295
12.4.2	MPU Control Register.....	296
12.4.3	MPU Region Number Register	297
12.4.4	MPU Region Base Address Register	297
12.4.5	MPU Region Base Attribute and Size Register	298
12.5	Setting Up the MPU	302
12.6	Memory Barrier and MPU Configuration.....	308
12.7	Using Sub-Region Disable	309
12.7.1	Allow Efficient Memory Separation.....	309
12.7.2	Reduce the Total Number of Regions Needed.....	310
12.8	Considerations When Using MPU.....	310
12.8.1	Program Code	311
12.8.2	Data Memory	311
12.9	Comparing with the MPU in the Cortex®-M3/M4/M7 Processors	312

Chapter 13: Debug Features	315
13.1 Software Development and Debug Features	315
13.2 Debug Interface	317
13.2.1 JTAG and Serial Wire Debug Communication Protocol	317
13.2.2 Cortex-M Processor and CoreSight™ Debug Architecture	319
13.2.3 Design Considerations with Debug Interface	320
13.3 Debug Features Overview	320
13.4 Debug System	321
13.5 Halt Mode and Debug Events	321
13.6 Instruction Tracing Support Using the MTB	324
Chapter 14: Getting Started with the Keil Microcontroller Development Kit	329
14.1 Introduction to Keil Microcontroller Development Kit	329
14.1.1 Overview	329
14.1.2 The Tools	330
14.1.3 Advantages of Using Keil MDK	330
14.1.4 Installation	331
14.2 Typical Program Compilation Flow	331
14.3 Introduction of the Hardware	334
14.3.1 Freescale Freedom Board (FRDM-KL25Z)	334
14.3.2 STMicroelectronics STM32L0 Discovery	335
14.3.3 STMicroelectronics STM32F0 Discovery	336
14.3.4 NXP LPC1114FN28	336
14.4 Getting Started with μ Vision® IDE	338
14.4.1 What Are Needed to Start	338
14.4.2 Starting Keil MDK	338
14.4.3 Project Setup Steps for Freescale FRDM-KL25Z	339
14.4.4 Project Setup Steps for STMicroelectronics STM32L0 Discovery	351
14.4.5 Project Setup Steps for STMicroelectronics STM32F0 Discovery	362
14.4.6 Project Setup Steps for NXP LPC1114FN28	376
14.5 Using the IDE and the Debugger	387
14.6 Under the Hood	391
14.6.1 CMSIS Files	391
14.6.2 Clock Setup	391
14.6.3 Stack and Heap Setup	391
14.6.4 Compilation	392
14.7 Customizations of the Project Environment	393
14.7.1 Target Options	393
14.7.2 Optimization Options	396
14.7.3 Runtime Environment Options	398
14.7.4 Project Management	398
14.8 Using the Simulator	400
14.9 Execution in SRAM	401
14.10 Using MTB for Instruction Trace	404

Chapter 15: Getting Started with IAR Embedded Workbench for ARM®	409
15.1 Overview of IAR Embedded Workbench for ARM®	409
15.2 Typical Program Compilation Flow	410
15.3 Creating a Simple Blinky Project	412
15.4 Project Options	420
15.5 Using MTB Instruction Trace with IAR EWARM	421
15.6 Hints and Tips	422
Chapter 16: Getting Started with gcc (GNU Compiler Collection)	427
16.1 About the GNU Compiler Collection Tool Chain	427
16.2 About the Examples in This Chapter	427
16.3 Typical Development Flow	428
16.4 Creating a Simple Blinky Project	431
16.5 Overview of the Command Line Options	433
16.6 Flash Programming	436
16.7 Using Keil® MDK-ARM™ with GNU Tools for ARM® Embedded Processors	438
16.8 Using CoCoX CoIDE with GNU Tools for ARM® Embedded Processors	445
16.8.1 Overview and Setup	445
16.8.2 Create a New Project	447
16.8.3 Using the IDE and the Debugger	454
Chapter 17: Getting Started with mbed™	459
17.1 What is mbed™	459
17.2 How the mbed™ System Works	460
17.3 Advantages of mbed™	462
17.4 Setting Up Your FRDM-KL25Z Board and mbed™ Account	463
17.4.1 Check Out mbed Web Page	463
17.4.2 Register for an Account with mbed	463
17.4.3 Additional Setup for the Personal Computer	463
17.5 Creating a Blinky Program	465
17.5.1 Simple Version with Just Red LED On/Off	465
17.5.2 LED with Pulse Width Modulation Control	467
17.6 Common Peripheral Objects Support	467
17.7 Using printf	468
17.8 Application Example—A Model Railway Controller	471
17.9 Interrupts	476
17.10 Hints and Tips	478
Chapter 18: Programming Examples	479
18.1 Producing Output with Universal Asynchronous Receiver/Transmitter	479
18.1.1 Overview of Universal Asynchronous Receiver/Transmitter Communication	479

18.1.2	Overview of UART Configurations on Microcontroller	482
18.1.3	Programming the UART on FRDM-KL25Z.....	482
18.1.4	Programming the UART on STM32L0 Discovery.....	484
18.1.5	Programming the UART on STM32F0 Discovery.....	486
18.1.6	Programming the UART on LPC1114FN28.....	487
18.2	Handling printf.....	490
18.2.1	Overview	490
18.2.2	Retargeting with Keil® MDK	491
18.2.3	Retargeting with IAR EWARM.....	492
18.2.4	Retargeting with GNU Compiler Collection.....	493
18.2.5	Semihosting with IAR EWARM.....	494
18.2.6	Semihosting with CoIDE.....	495
18.3	Developing Your Own Input and Output Functions.....	495
18.3.1	Why Reinventing the Wheel?	495
18.3.2	Other Interfaces	500
18.3.3	Other Hints and Tips About scanf.....	501
18.4	Interrupt Programming Examples	502
18.4.1	General Overview of Interrupt Handling.....	502
18.4.2	Overview of Interrupt Control Functions	502
18.5	Application Example—Another Controller for a Model Train	504
18.6	Different Versions of CMSIS-CORE.....	508
Chapter 19: Ultralow-Power Designs.....		511
19.1	Examples of Using Low-Power Features	511
19.1.1	Overview	511
19.1.2	Entering Sleep Modes	511
19.1.3	WFE versus WFI.....	513
19.1.4	Using Sleep-On-Exit Feature	514
19.1.5	Using Send-Event-on-Pend Feature	515
19.1.6	Using Wake-up Interrupt Controller	516
19.1.7	Using Event Communication Interface.....	517
19.2	Requirements of Low-Power Designs.....	520
19.3	Where Does the Power Go?.....	521
19.4	Developing Low-Power Applications	523
19.4.1	Overview of Low-Power Design Practices.....	523
19.4.2	Various Approaches to Reduce Power.....	524
19.4.3	Selecting the Right Approach	525
19.5	Debug Considerations.....	527
19.5.1	Debug and Low-Power.....	527
19.5.2	“Safe Mode” for Debug and Flash Programming.....	527
19.5.3	Debug Interface and Low-Voltage Pins.....	527
19.6	Benchmarking of Low-Power Devices	528
19.6.1	Background of ULPBench™.....	528
19.6.2	Overview of the ULPBench-CP.....	528

19.7	Example of Using Low-Power Features on Freescale KL25Z.....	532
19.7.1	Objective	532
19.7.2	Test Setup.....	532
19.7.3	Low-Power Modes on KL25Z	533
19.7.4	Clocking Arrangement	533
19.7.5	The Test Setup.....	535
19.7.6	Measurement Results.....	540
19.8	Example of Using Low-Power Feature on LPC1114.....	542
19.8.1	Overview of LPC1114FN28.....	542
19.8.2	First Experiment—Running at 12 MHz with Internal and External Crystal	545
19.8.3	Second Experiment—Running at Reduced Frequencies of 1 MHz and 100 KHz	548
19.8.4	Additional Improvements	549
19.8.5	Using Deep Sleep on LPC1114.....	550
Chapter 20: Programming with Embedded OS		559
20.1	Introduction.....	559
20.1.1	Background	559
20.1.2	Embedded OS and RTOS.....	559
20.1.3	Why Use an Embedded OS?.....	560
20.1.4	Role of CMSIS-RTOS.....	560
20.1.5	About the Keil® RTX Kernel.....	562
20.1.6	Setting Up a Simple RTX Example with Keil MDK	563
20.2	Overview of the RTX Kernel.....	567
20.2.1	Thread	567
20.2.2	RTX Configurations.....	569
20.2.3	A Closer Look at the First Example.....	569
20.2.4	Interthread Communication Overview.....	573
20.2.5	Signal Event Communication.....	574
20.2.6	Mutual Exclusive (Mutex).....	578
20.2.7	Semaphore.....	580
20.2.8	Message Queue.....	583
20.2.9	Mail Queue	585
20.2.10	Memory Pool Management Feature.....	588
20.2.11	Generic Wait Function and Time-Out Value.....	590
20.2.12	Timer Feature.....	590
20.2.13	Adding SVC Services for Unprivileged Threads	593
20.3	Using RTX in an Application	597
20.4	Debugging an Application with RTX.....	600
20.5	Trouble Shooting	601
20.5.1	Stack Size Requirements.....	602
20.5.2	Privileged Level.....	602
20.5.3	Utilize OS Error Reporting Support	603

20.5.4 OS Feature Configurations	603
20.5.5 Miscellaneous	603
20.6 Other Hints and Tips	604
20.6.1 Customization of RTX_Config_CM.c.....	604
20.6.2 Thread Priority.....	604
20.6.3 A Short Waiting Time.....	604
20.6.4 Additional Information.....	605
Chapter 21: Mixed Language Projects (C/C++ with Assembly).....	607
21.1 Use of Assembly in Project Developments.....	607
21.2 Recommended Practices in Assembly Programming and AAPCS	608
21.3 Overview of an Assembly Function	610
21.3.1 ARM® Tool Chains	610
21.3.2 Gcc Tool Chains	611
21.3.3 IAR Embedded Workbench for ARM	612
21.3.4 Structure of an Assembly Function	612
21.4 Inline Assembly	613
21.4.1 ARM® Tool Chains (Keil® MDK/DS-5).....	613
21.4.2 GNU Compiler Collection	615
21.5 Embedded Assembler Feature (ARM® Tool Chain).....	616
21.6 Mixed Language Projects.....	617
21.6.1 Overview	617
21.6.2 Calling C Functions from Assembly Codes.....	617
21.6.3 Calling Assembly Functions from C Codes.....	618
21.7 Creating Assembly Projects in Keil® MDK-ARM	619
21.7.1 A Small Project	619
21.7.2 Hello World	620
21.7.3 Additional Text Output Functions.....	621
21.8 Generic Assembly Code for Interrupt Control	624
21.8.1 Enable and Disable Interrupts.....	624
21.8.2 Set and Clear Interrupt Pending Status	625
21.8.3 Setting Up Interrupt Priority Level.....	626
21.9 Other Programming Techniques for Assembly Language	628
21.9.1 Allocating Data Space for Variables.....	628
21.9.2 Complex Branch Handling.....	630
21.10 Accessing Special Instructions.....	631
21.10.1 CMSIS-CORE.....	631
21.10.2 Idiom Recognitions.....	632
Chapter 22: Software Porting.....	635
22.1 Overview	635
22.2 Porting Software from 8-Bit/16-Bit Microcontrollers to ARM® Cortex®-M.....	635
22.2.1 Common Modifications	635
22.2.2 Memory Requirements	637

22.2.3 Nonapplicable Optimizations for 8-Bit or 16-Bit Microcontrollers	638
22.2.4 Example—Migrate from 8051 to ARM Cortex-M0/Cortex-M0+	639
22.3 Differences between ARM7TDMI™ and Cortex®-M0/M0+ Processor.....	641
22.3.1 Overview of Classic ARM® Processors	641
22.3.2 Operation Mode	642
22.3.3 Registers.....	643
22.3.4 Instruction Set.....	644
22.3.5 Interrupts	644
22.4 Porting Software from ARM7TDMI™ to the Cortex®-M0/Cortex-M0+ Processors	645
22.4.1 Start-up Code and Vector Table.....	645
22.4.2 Interrupt.....	645
22.4.3 C Program Code	646
22.4.4 Assembly Code.....	647
22.4.5 Atomic Access.....	647
22.4.6 Optimizations.....	647
22.5 Differences between Various Cortex®-M Processors	648
22.5.1 Overview	648
22.5.2 Programmer's Model	649
22.5.3 NVIC and Exceptions.....	650
22.5.4 Instruction Set.....	653
22.5.5 System Level Features.....	653
22.5.6 Debug and Trace Features.....	655
22.6 General Software Modifications when Porting between Cortex®-M Processors	656
22.7 Porting Software between Cortex®-M0/M0+ and Cortex-M1	656
22.8 Porting Software between Cortex®-M0/M0+ and Cortex-M3	657
22.9 Porting Software between Cortex®-M0/M0+ and the Cortex-M4/M7 Processor.....	659
Chapter 23: Advanced Topics	661
23.1 Bit Data Handling in C Programming	661
23.2 Startup Code in C.....	663
23.3 Stack Overflow Detection.....	668
23.3.1 What is Stack Overflow?.....	668
23.3.2 Stack Analysis by Tool Chain.....	669
23.3.3 Stack Analysis by Trial	669
23.3.4 Stack Limit Using Memory Protection Unit	670
23.3.5 Stack Checking in OS Context Switching.....	670
23.4 Reentrant Interrupt Service Routine	671
23.5 Semaphore Implementation	673
23.6 Memory Ordering and Memory Barriers.....	674
Appendix A: Instruction Set Quick Reference.....	679
Appendix B: Exception Type Quick Reference.....	683

<i>Appendix C: CMSIS-CORE Quick Reference</i>	<i>685</i>
<i>Appendix D: NVIC, SCB, and SysTick Registers Quick Reference</i>	<i>691</i>
<i>Appendix E: Debug Registers Quick Reference</i>	<i>699</i>
<i>Appendix F: Debug Connector Arrangements.....</i>	<i>711</i>
<i>Appendix G: Trouble Shooting</i>	<i>715</i>
<i>Appendix H: A Breadboard Project with an ARM[®] Cortex[®]-M0 Microcontroller.....</i>	<i>729</i>
<i>Index.....</i>	<i>733</i>