

Contents

Foreword	xxi
Preface	xxiii
Synopsis	xxv
About this Book	xxvii
Contributor Bio-Paul Beckmann	xxix
Acknowledgments	xxxii
Terms and Abbreviations	xxxiii
Conventions	xxxv

CHAPTER 1 Introduction to ARM[®] Cortex[®]-M Processors.....	1
1.1. What are the ARM[®] Cortex[®]-M processors?	2
1.1.1. The Cortex [®] -M3 and Cortex-M4 processors	2
1.1.2. The Cortex [®] -M processor family	3
1.1.3. Differences between a processor and a microcontroller	4
1.1.4. ARM [®] and the microcontroller vendors	5
1.1.5. Selecting Cortex [®] -M3 and Cortex-M4 microcontrollers	6
1.2. Advantages of the Cortex[®]-M processors	8
1.2.1. Low power	8
1.2.2. Performance	9
1.2.3. Energy efficiency	9
1.2.4. Code density	9
1.2.5. Interrupts	9
1.2.6. Ease of use, C friendly	9
1.2.7. Scalability	10
1.2.8. Debug features	10
1.2.9. OS support	10
1.2.10. Versatile system features	10
1.2.11. Software portability and reusability	10
1.2.12. Choices (devices, tools, OS, etc.)	10
1.3. Applications of the ARM[®] Cortex[®]-M processors	11
1.4. Resources for using ARM[®] processors and ARM	
microcontrollers	12
1.4.1. What can you find on the ARM [®] website	12
1.4.2. Documentation from the microcontroller vendors	12
1.4.3. Documentation from tools vendors	14
1.4.4. Other resources	14
1.5. Background and history	15
1.5.1. A brief history of ARM [®]	15
1.5.2. ARM [®] processor evolution	16
1.5.3. Architecture versions and Thumb [®] ISA	18

1.5.4. Processor naming.....	22
1.5.5. About the ARM® ecosystem.....	23
CHAPTER 2 Introduction to Embedded Software Development ...	25
2.1. What are inside typical ARM® microcontrollers?.....	25
2.2. What you need to start.....	26
2.2.1. Development suites.....	26
2.2.2. Development boards.....	27
2.2.3. Debug adaptor.....	27
2.2.4. Software device driver.....	29
2.2.5. Examples.....	29
2.2.6. Documentation and other resources.....	30
2.2.7. Other equipment.....	30
2.3. Software development flow.....	30
2.4. Compiling your applications.....	32
2.5. Software flow.....	36
2.5.1. Polling.....	36
2.5.2. Interrupt driven.....	36
2.5.3. Multi-tasking systems.....	38
2.6. Data types in C programming.....	39
2.7. Inputs, outputs, and peripherals accesses.....	40
2.8. Microcontroller interfaces.....	45
2.9. The Cortex® microcontroller software interface standard (CMSIS).....	46
2.9.1. Introduction of CMSIS.....	46
2.9.2. Areas of standardization in CMSIS-Core.....	48
2.9.3. Organization of CMSIS-Core.....	50
2.9.4. How do I use CMSIS-Core?.....	50
2.9.5. Benefits of CMSIS-Core.....	53
2.9.6. Various versions of CMSIS.....	54
CHAPTER 3 Technical Overview	57
3.1. General information about the Cortex®-M3 and Cortex-M4 processors.....	57
3.1.1. Processor type.....	57
3.1.2. Processor architecture.....	58
3.1.3. Instruction set.....	59
3.1.4. Block diagram.....	61
3.1.5. Memory system.....	63
3.1.6. Interrupt and exception support.....	64
3.2. Features of the Cortex®-M3 and Cortex-M4 processors.....	64
3.2.1. Performance.....	65
3.2.2. Code density.....	65

3.2.3. Low power	66
3.2.4. Memory system	67
3.2.5. Memory protection unit.....	67
3.2.6. Interrupt handling	68
3.2.7. OS support and system level features	69
3.2.8. Cortex [®] -M4 specific features	69
3.2.9. Ease of use.....	70
3.2.10. Debug support.....	71
3.2.11. Scalability	72
3.2.12. Compatibility	73
CHAPTER 4 Architecture	75
4.1. Introduction to the architecture	76
4.2. Programmer's model.....	76
4.2.1. Operation modes and states.....	76
4.2.2. Registers.....	78
4.2.3. Special registers	81
4.2.4. Floating point registers	90
4.3. Behavior of the application program status register (APSR)	92
4.3.1. Integer status flags	93
4.3.2. Q status flag	94
4.3.3. GE bits	95
4.4. Memory system.....	97
4.4.1. Memory system features	97
4.4.2. Memory map.....	98
4.4.3. Stack memory	99
4.4.4. Memory protection unit (MPU)	103
4.5. Exceptions and interrupts	104
4.5.1. What are exceptions?	104
4.5.2. Nested vectored interrupt controller (NVIC).....	106
4.5.3. Vector table	107
4.5.4. Fault handling	108
4.6. System control block (SCB).....	109
4.7. Debug	109
4.8. Reset and reset sequence	113
CHAPTER 5 Instruction Set	117
5.1. Background to the instruction set in ARM [®] Cortex [®] -M processors	118
5.2. Comparison of the instruction set in ARM [®] Cortex [®] -M processors	120
5.3. Understanding the assembly language syntax.....	123
5.4. Use of a suffix in instructions.....	128

5.5.	Unified assembly language (UAL)	129
5.6.	Instruction set.....	131
5.6.1.	Moving data within the processor.....	132
5.6.2.	Memory access instructions	134
5.6.3.	Arithmetic operations	146
5.6.4.	Logic operations	148
5.6.5.	Shift and rotate instructions	148
5.6.6.	Data conversion operations (extend and reverse ordering).....	150
5.6.7.	Bit-field processing instructions.....	152
5.6.8.	Compare and test.....	154
5.6.9.	Program flow control.....	154
5.6.10.	Saturation operations.....	164
5.6.11.	Exception-related instructions.....	165
5.6.12.	Sleep mode-related instructions.....	168
5.6.13.	Memory barrier instructions.....	169
5.6.14.	Other instructions	170
5.6.15.	Unsupported instructions.....	172
5.7.	Cortex [®] -M4-specific instructions	173
5.7.1.	Overview of enhanced DSP extension in Cortex-M4.....	173
5.7.2.	SIMD and saturating instructions.....	175
5.7.3.	Multiply and MAC instructions	175
5.7.4.	Packing and unpacking.....	179
5.7.5.	Floating point instructions.....	181
5.8.	Barrel shifter	184
5.9.	Accessing special instructions and special registers in programming.....	189
5.9.1.	Overview.....	189
5.9.2.	Intrinsic functions.....	190
5.9.3.	Inline assembler and embedded assembler.....	190
5.9.4.	Using other compiler-specific features.....	191
5.9.5.	Access of special registers	191
CHAPTER 6	Memory System.....	193
6.1.	Overview of memory system features.....	193
6.2.	Memory map.....	194
6.3.	Connecting the processor to memory and peripherals.....	194
6.4.	Memory requirements.....	202
6.5.	Memory endianness	202
6.6.	Data alignment and unaligned data access support	205
6.7.	Bit-band operations.....	206
6.7.1.	Overview	206
6.7.2.	Advantages of bit-band operations	210

6.7.3. Bit-band operation of different data sizes	211
6.7.4. Bit-band operations in C programs.....	216
6.8. Default memory access permissions	217
6.9. Memory access attributes	217
6.10. Exclusive accesses	220
6.11. Memory barriers	223
6.12. Memory system in a microcontroller.....	224

CHAPTER 7 Exceptions and Interrupts 229

7.1. Overview of exceptions and interrupts.....	230
7.2. Exception types.....	231
7.3. Overview of interrupt management.....	233
7.4. Definitions of priority	235
7.5. Vector table and vector table relocation.....	242
7.6. Interrupt inputs and pending behaviors.....	246
7.7. Exception sequence overview.....	250
7.7.1. Acceptance of exception request	250
7.7.2. Exception entrance sequence	250
7.7.3. Exception handler execution.....	251
7.7.4. Exception return.....	251
7.8. Details of NVIC registers for interrupt control.....	252
7.8.1. Summary	252
7.8.2. Interrupt enable registers.....	253
7.8.3. Interrupt set pending and clear pending.....	253
7.8.4. Active status	255
7.8.5. Priority level.....	255
7.8.6. Software trigger interrupt register	257
7.8.7. Interrupt controller type register.....	258
7.9. Details of SCB registers for exception and interrupt control	259
7.9.1. Summary of the SCB registers	259
7.9.2. Interrupt control and state register (ICSR).....	259
7.9.3. Vector table offset register (VTOR).....	259
7.9.4. Application interrupt and reset control register (AIRCR).....	259
7.9.5. System handler priority registers (SCB->SHP[0 to 11]).....	263
7.9.6. System handler control and state register (SCB->SHCSR).....	264
7.10. Details of special registers for exception or interrupt masking.....	265
7.10.1. PRIMASK.....	265
7.10.2. FAULTMASK	266
7.10.3. BASEPRI.....	267

7.11.	Example procedures in setting up interrupts.....	268
7.11.1.	Simple cases.....	268
7.11.2.	With vector table relocation.....	269
7.12.	Software interrupts.....	270
7.13.	Tips and hints.....	271
CHAPTER 8 Exception Handling in Detail.....		273
8.1.	Introduction.....	273
8.1.1.	About this chapter.....	273
8.1.2.	Exception handler in C.....	274
8.1.3.	Stack frames.....	274
8.1.4.	EXC_RETURN.....	278
8.2.	Exception sequences.....	279
8.2.1.	Exception entrance and stacking.....	279
8.2.2.	Exception return and unstacking.....	281
8.3.	Interrupt latency and exception handling optimization.....	281
8.3.1.	What is interrupt latency?.....	281
8.3.2.	Interrupts at multiple-cycle instructions.....	284
8.3.3.	Tail chaining.....	284
8.3.4.	Late arrival.....	285
8.3.5.	Pop preemption.....	286
8.3.6.	Lazy stacking.....	286
CHAPTER 9 Low Power and System Control Features.....		289
9.1.	Low power designs.....	290
9.1.1.	What does low power mean in microcontrollers?.....	290
9.1.2.	Low power system requirements.....	292
9.1.3.	Low power characteristics of the Cortex [®] -M3 and Cortex-M4 processors.....	293
9.2.	Low power features.....	293
9.2.1.	Sleep modes.....	293
9.2.2.	System control register (SCR).....	294
9.2.3.	Entering sleep modes.....	295
9.2.4.	Wake-up conditions.....	296
9.2.5.	Sleep-on-Exit feature.....	297
9.2.6.	Send event on pend (SEVONPEND).....	300
9.2.7.	Sleep extension/wake-up delay.....	300
9.2.8.	Wake-up interrupt controller (WIC).....	300
9.2.9.	Event communication interface.....	302
9.3.	Using WFI and WFE instructions in programming.....	307
9.3.1.	When to use WFI.....	307
9.3.2.	Using WFE.....	308

9.4.	Developing low power applications	309
9.4.1.	Reducing the active power.....	310
9.4.2.	Reduction of active cycles.....	311
9.4.3.	Sleep mode current reduction	311
9.5.	The SysTick timer.....	312
9.5.1.	Why have a SysTick timer.....	312
9.5.2.	Operations of the SysTick timer.....	313
9.5.3.	Using the SysTick timer	314
9.5.4.	Other considerations.....	318
9.6.	Self-reset	318
9.7.	CPU ID base register	319
9.8.	Configuration control register.....	320
9.8.1.	Overview of CCR.....	320
9.8.2.	STKALIGN bit.....	320
9.8.3.	BFHFNMIGN bit.....	321
9.8.4.	DIV_0_TRP bit	322
9.8.5.	UNALIGN_TRP bit	322
9.8.6.	USERSETMPEND bit	322
9.8.7.	NONBASETHRDENA bit.....	323
9.9.	Auxiliary control register.....	323
9.10.	Co-processor access control register	325
CHAPTER 10	OS Support Features	327
10.1.	Overview of OS support features.....	327
10.2.	Shadowed stack pointer.....	328
10.3.	SVC exception.....	330
10.4.	PendSV exception.....	336
10.5.	Context switching in action.....	339
10.6.	Exclusive accesses and embedded OS	352
CHAPTER 11	Memory Protection Unit (MPU).....	355
11.1.	Overview of the MPU	355
11.1.1.	About the MPU.....	355
11.1.2.	Using the MPU	356
11.2.	MPU registers.....	357
11.2.1.	MPU type register.....	357
11.2.2.	MPU control register.....	357
11.2.3.	MPU region number register	360
11.2.4.	MPU region base address register	360
11.2.5.	MPU region base attribute and size register	361
11.2.6.	MPU alias registers	364
11.3.	Setting up the MPU.....	365
11.4.	Memory barrier and MPU configuration	373

- 11.5. Using sub-region disable 374
 - 11.5.1. Allow efficient memory separation 374
 - 11.5.2. Reduce the total number of regions needed 374
- 11.6. Considerations when using MPU 376
 - 11.6.1. Program code 376
 - 11.6.2. Data memory 376
 - 11.6.3. Peripherals 377
- 11.7. Other usages of the MPU 377
- 11.8. Comparing with the MPU in the Cortex[®]-M0+ processor 378

CHAPTER 12 Fault Exceptions and Fault Handling 379

- 12.1. Overview of fault exceptions 380
- 12.2. Causes of faults 382
 - 12.2.1. Memory management (MemManage) faults 382
 - 12.2.2. Bus faults 382
 - 12.2.3. Usage faults 384
 - 12.2.4. HardFaults 384
- 12.3. Enabling fault handlers 385
 - 12.3.1. MemManage fault 385
 - 12.3.2. Bus fault 385
 - 12.3.3. Usage fault 385
 - 12.3.4. HardFault 386
- 12.4. Fault status registers and fault address registers 386
 - 12.4.1. Summary 386
 - 12.4.2. Information for MemManage fault 387
 - 12.4.3. Information for bus fault 388
 - 12.4.4. Information for usage fault 388
 - 12.4.5. HardFault status register 389
 - 12.4.6. Debug fault status register (DFSR) 389
 - 12.4.7. Fault address registers MMFAR and BFAR 390
 - 12.4.8. Auxiliary fault status register 391
- 12.5. Analyzing faults 392
- 12.6. Faults related to exception handling 396
 - 12.6.1. Stacking 396
 - 12.6.2. Unstacking 396
 - 12.6.3. Lazy stacking 396
 - 12.6.4. Vector fetches 397
 - 12.6.5. Invalid returns 397
 - 12.6.6. Priority levels and stacking or unstacking faults 397
- 12.7. Lockup 399
 - 12.7.1. What is lockup? 399
 - 12.7.2. Avoiding lockup 400

12.8.	Fault handlers	401
12.8.1.	HardFault handler for debug purpose.....	401
12.8.2.	Fault mask	405
12.9.	Additional information	406
12.9.1.	Running a system with two stacks	406
12.9.2.	Detect stack overflow.....	407
CHAPTER 13	Floating Point Operations	409
13.1.	About floating point data.....	410
13.1.1.	Introduction	410
13.1.2.	Single-precision floating point numbers.....	410
13.1.3.	Half-precision floating point numbers	412
13.1.4.	Double-precision floating point numbers	414
13.1.5.	Floating point support in Cortex [®] -M processors	415
13.2.	Cortex [®] -M4 floating point unit (FPU).....	416
13.2.1.	Floating point unit overview	416
13.2.2.	Floating point registers overview	417
13.2.3.	CPACR register	417
13.2.4.	Floating point register bank	419
13.2.5.	Floating point status and control register (FPSCR)	420
13.2.6.	Floating point context control register (FPU->FPCCR)	420
13.2.7.	Floating point context address register (FPU->FPCAR)	420
13.2.8.	Floating point default status control register (FPU->FPDSCR).....	425
13.2.9.	Media and floating point feature registers (FPU->MVFR0, FPU->MVFR1)	426
13.3.	Lazy stacking in detail	426
13.3.1.	Key elements of the lazy stacking feature	426
13.3.2.	Scenario #1: No floating point context in interrupted task.....	427
13.3.3.	Scenario #2: Floating point context in interrupted task but not in ISR	428
13.3.4.	Scenario #3: Floating point context in interrupted task and in ISR.....	428
13.3.5.	Scenario #4: Nested interrupt with floating point context in the second handler.....	429
13.3.6.	Scenario #5: Nested interrupt with floating point context in the both handlers	430
13.3.7.	Interrupt of lazy stacking	430
13.3.8.	Interrupt of floating point instructions.....	431

13.4.	Using the floating point unit	432
13.4.1.	Floating point support in CMSIS-Core	432
13.4.2.	Floating point programming in C	433
13.4.3.	Compiler command line options	433
13.4.4.	ABI Options: Hard-vfp and Soft-vfp	434
13.4.5.	Special FPU modes	436
13.5.	Floating point exceptions	438
13.6.	Hints and tips	441
13.6.1.	Run-time libraries for microcontrollers	441
13.6.2.	Debug operation	441
CHAPTER 14	Introduction to the Debug and Trace Features	443
14.1.	Debug and trace features overview	444
14.1.1.	What are debug features?	444
14.1.2.	What are trace features?	445
14.1.3.	Debug and trace features summaries	447
14.2.	Debug architecture	449
14.2.1.	CoreSight™ debug architecture	449
14.2.2.	Processor debug interface	450
14.2.3.	Debug Port (DP), Access Port (AP), and Debug Access Port (DAP)	452
14.2.4.	Trace interface	454
14.2.5.	CoreSight characteristics	454
14.3.	Debug modes	456
14.4.	Debug events	459
14.5.	Breakpoint feature	460
14.6.	Debug components introduction	462
14.6.1.	Processor debug support	462
14.6.2.	Flash patch and breakpoint (FPB) unit	467
14.6.3.	Data watchpoint and trace (DWT) unit	470
14.6.4.	Instrumentation trace macrocell (ITM)	473
14.6.5.	Embedded trace macrocell (ETM)	476
14.6.6.	Trace port interface unit (TPIU)	478
14.6.7.	ROM table	479
14.6.8.	AHB access port (AHB-AP)	481
14.7.	Debug operations	484
14.7.1.	Debug connection	484
14.7.2.	Flash programming	485
14.7.3.	Breakpoints	485
CHAPTER 15	Getting Started with Keil Microcontroller Development Kit for ARM®	487
15.1.	Overview	487
15.2.	Typical program compilation flow	488

15.3.	Getting started with μ Vision	491
15.4.	Project options	509
15.4.1.	Device option	509
15.4.2.	Target options	509
15.4.3.	Output options	509
15.4.4.	Listing options	513
15.4.5.	User options	513
15.4.6.	C/C++ options	513
15.4.7.	Assembler options	518
15.4.8.	Linker options	518
15.4.9.	Debug options	518
15.4.10.	Utilities options	518
15.5.	Using the IDE and the debugger	518
15.6.	Using the instruction set simulator	524
15.7.	Running programs from SRAM	530
15.8.	Optimization options	534
15.9.	Other hints and tips	539
15.9.1.	Stack and heap memory size configurations	539
15.9.2.	Other information	539

CHAPTER 16 Getting Started with the IAR Embedded

	Workbench for ARM[®]	541
16.1.	Overview of the IAR embedded workbench for ARM [®]	541
16.2.	Typical program compilation flow	542
16.3.	Creating a simple blinky project	544
16.4.	Project options	553
16.5.	Hints and tips	553

CHAPTER 17 Getting Started with the GNU Compiler

	Collection (gcc)	561
17.1.	The GNU Compiler Collection (gcc) toolchain	561
17.2.	Typical development flow	562
17.3.	Creating a simple blinky project	565
17.4.	Overview of the command line options	566
17.5.	Flash programming	567
17.5.1.	Using Keil MDK-ARM	569
17.5.2.	Using third-party flash programming utilities	569
17.6.	Using Keil [™] MDK-ARM with GNU tools for ARM Embedded Processors	570
17.7.	Using CoIDE with GNU tools for ARM [®] Embedded Processors	572
17.8.	Commercial gcc-based development suites	577
17.8.1.	Atollic TrueSTUDIO for ARM [®]	580
17.8.2.	Red Suite	582
17.8.3.	CrossWorks for ARM [®]	582

CHAPTER 18	Input and Output Software Examples	583
18.1.	Producing outputs	583
18.2.	Re-targeting to the Instrumentation Trace Macrocell (ITM) ..	584
18.2.1.	Overview	584
18.2.2.	Keil™ MDK-ARM.....	584
18.2.3.	IAR Embedded Workbench	588
18.2.4.	GCC.....	590
18.3.	Semi-hosting	591
18.4.	Re-targeting to peripherals	595
CHAPTER 19	Using Embedded Operating Systems	605
19.1.	Introduction to embedded OSs.....	605
19.1.1.	What are embedded OSs?.....	605
19.1.2.	When to use an embedded OS.....	606
19.1.3.	Role of CMSIS-RTOS	607
19.2.	Keil™ RTX Real-Time Kernel.....	609
19.2.1.	About RTX.....	609
19.2.2.	Features overview	609
19.2.3.	RTX and CMSIS-RTOS.....	610
19.2.4.	Thread.....	611
19.3.	CMSIS-OS examples.....	613
19.3.1.	Simple CMSIS-RTOS with two threads.....	613
19.3.2.	Inter-thread communication overview	619
19.3.3.	Signal event communication.....	621
19.3.4.	Mutual Exclusive (Mutex).....	625
19.3.5.	Semaphore.....	626
19.3.6.	Message queue	630
19.3.7.	Mail queue.....	632
19.3.8.	Memory pool management feature.....	635
19.3.9.	Generic wait function and time-out value	637
19.3.10.	Timer feature.....	637
19.3.11.	Access privileged devices	640
19.4.	OS-aware debugging	642
19.5.	Troubleshooting	643
19.5.1.	Stack size and stack alignment.....	643
19.5.2.	Privileged level.....	644
19.5.3.	Miscellaneous.....	645
CHAPTER 20	Assembly and Mixed Language Projects	647
20.1.	Use of assembly code in projects.....	647
20.2.	Interaction between C and assembly	648
20.3.	Structure of an assembly function	650
20.4.	Examples.....	652

20.4.1.	Simple example with ARM [®] toolchains (Keil [™] MDK-ARM, DS-5)	652
20.4.2.	Simple example with GNU tools for ARM-embedded processors	654
20.4.3.	Accessing special registers	656
20.4.4.	Data memory	656
20.4.5.	Hello world	658
20.4.6.	Displaying values in hexadecimal and decimal	660
20.4.7.	NVIC interrupt control	662
20.4.8.	Unsigned integer square root	664
20.5.	Mixed language projects	665
20.5.1.	Calling a C function from assembly	665
20.5.2.	Calling an assembly function from C	666
20.5.3.	Embedded assembler (Keil [™] MDK-ARM/ARM [®] DS-5 [™] professional)	667
20.5.4.	Inline assembler	667
20.6.	Intrinsic functions	669
20.7.	Idiom recognition	670
 CHAPTER 21 ARM[®] Cortex[®]-M4 and DSP Applications		673
21.1.	DSP on a microcontroller?	674
21.2.	Dot product example	674
21.3.	Architecture of a traditional DSP processor	676
21.4.	Cortex [®] -M4 DSP instructions	680
21.4.1.	Registers and data types	681
21.4.2.	Fractional arithmetic	683
21.4.3.	SIMD data	684
21.4.4.	Load and store instructions	685
21.4.5.	Arithmetic instructions	685
21.4.6.	General Cortex [®] -M4 optimization strategies	694
21.4.7.	Instruction limitations	697
21.5.	Writing optimized DSP code for the Cortex [®] -M4	697
21.5.1.	Biquad filter	697
21.5.2.	Fast Fourier transform	703
21.5.3.	FIR filter	709
 CHAPTER 22 Using the ARM[®] CMSIS-DSP Library		717
22.1.	Overview of the library	717
22.2.	Pre-built binaries	718
22.3.	Function naming convention	718
22.4.	Getting help	719
22.5.	Example 1 – DTMF demodulation	719
22.5.1.	Generating the sine wave	720

22.5.2. Decoding using an FIR filter	720
22.5.3. Decoding using an FFT	722
22.5.4. Decoding using a Biquad filter	725
22.5.5. Example DTMF code.....	727
22.6. Example 2 – least squares motion tracking	730
22.6.1. Example least squares code	732
CHAPTER 23 Advanced Topics	737
23.1. Decisions and branches	737
23.1.1. Conditional branches.....	737
23.1.2. Complex decision tree	741
23.2. Performance considerations.....	742
23.3. Double-word stack alignment	745
23.4. Various methods for semaphore implementation	745
23.4.1. Using SVC services for semaphores	746
23.4.2. Use bit-band for semaphores	746
23.5. Non-base Thread enable.....	747
23.6. Re-entrant Interrupt Handler	750
23.7. Bit Data Handling in C	756
23.8. Startup code	759
23.9. Stack overflow detection	759
23.9.1. Stack analysis by toolchain.....	759
23.9.2. Stack analysis by trial	760
23.9.3. Stack overflow detection by stack placement	760
23.9.4. Using MPU.....	761
23.9.5. Using DWT and Debug Monitor Exception.....	761
23.9.6. Stack checking in OS context switching	762
23.10. Flash patch feature	762
23.11. Revision versions of the Cortex [®] -M3 and Cortex-M4 processors.....	766
23.11.1. Overview	766
23.11.2. Changes from Cortex [®] -M3 r0p0 to r1p0/r1p1	766
23.11.3. Changes from Cortex [®] -M3 r1p1 to r2p0.....	767
23.11.4. Changes from Cortex [®] -M3 r2p0 to r2p1.....	770
23.11.5. Changes from Cortex [®] -M4 r0p0 to r0p1.....	770
CHAPTER 24 Software Porting	771
24.1. Overview.....	772
24.2. Porting software from 8-bit/16-bit MCUs to Cortex [®] -M MCUs.....	772
24.2.1. Architectural differences.....	772
24.2.2. Common modifications	774
24.2.3. Memory size requirements.....	775

24.2.4. Non-applicable optimizations for 8-bit or 16-bit microcontrollers	776
24.2.5. Example — migrate from 8051 to ARM® Cortex®-M.....	777
24.3. Porting software from ARM7TDMI™ to Cortex®-M3/M4.....	779
24.3.1. Overview of the hardware differences.....	779
24.3.2. Assembly language files	783
24.3.3. C language files.....	786
24.3.4. Pre-compiled object files and libraries	787
24.3.5. Optimization.....	787
24.4. Porting software between different Cortex®-M processors	788
24.4.1. Differences between different Cortex®-M processors	788
24.4.2. Required software changes	792
24.4.3. Embedded OS.....	795
24.4.4. Creating portable program code for the Cortex®-M processors.....	797
References.....	799
Index	801

Appendices A–I are available on the book’s companion website