



CARMINE NOVIELLO

MASTERING STM32

A step-by-step guide to the most complete
ARM Cortex-M platform, using a free
and powerful development environment
based on Eclipse and GCC

Mastering STM32

A step-by-step guide to the most complete ARM Cortex-M platform, using a free and powerful development environment based on Eclipse and GCC

Carmine Noviello

This book is for sale at <http://leanpub.com/mastering-stm32>

This version was published on 2017-07-21



Leanpub

This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2015 - 2017 Carmine Noviello

Tweet This Book!

Please help Carmine Noviello by spreading the word about this book on [Twitter!](#)

The suggested hashtag for this book is [#MasteringSTM32](#).

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

<https://twitter.com/search?q=#MasteringSTM32>

Contents

Preface	i
Why Did I Write the Book?	i
Who Is This Book For?	ii
How to Integrate This Book?	iii
How Is the Book Organized?	iv
About the Author	vii
Errata and Suggestions	viii
Book Support	viii
How to Help the Author	ix
Copyright Disclaimer	ix
Credits	ix
Acknowledgments	x
I Introduction	1
1. Introduction to STM32 MCU Portfolio	2
1.1 Introduction to ARM Based Processors	2
1.1.1 Cortex and Cortex-M Based Processors	4
1.1.1.1 Core Registers	5
1.1.1.2 Memory Map	7
1.1.1.3 Bit-Banding	9
1.1.1.4 Thumb-2 and Memory Alignment	12
1.1.1.5 Pipeline	13
1.1.1.6 Interrupts and Exceptions Handling	15
1.1.1.7 SysTimer	17
1.1.1.8 Power Modes	17
1.1.1.9 CMSIS	19
1.1.1.10 Effective Implementation of Cortex-M Features in the STM32 Portfolio	20
1.2 Introduction to STM32 Microcontrollers	21
1.2.1 Advantages of the STM32 Portfolio....	22

CONTENTS

1.2.2And Its Drawbacks	23
1.3	A Quick Look at the STM32 Subfamilies	24
1.3.1	F0	26
1.3.2	F1	27
1.3.3	F2	28
1.3.4	F3	30
1.3.5	F4	32
1.3.6	F7	33
1.3.7	H7	34
1.3.8	L0	35
1.3.9	L1	36
1.3.10	L4	38
1.3.11	W and J STM32 MCUs	39
1.3.12	How to Select the Right MCU for You?	39
1.4	The Nucleo Development Board	42
2.	Setting-Up the Tool-Chain	48
2.1	Why Choose Eclipse/GCC as Tool-Chain for STM32	49
2.1.1	Two Words About Eclipse...	50
2.1.2	... and GCC	50
2.2	Windows - Installing the Tool-Chain	51
2.2.1	Windows - Eclipse Installation	52
2.2.2	Windows - Eclipse Plug-Ins Installation	54
2.2.3	Windows - GCC ARM Embedded Installation	60
2.2.4	Windows – Build Tools Installation	62
2.2.5	Windows – OpenOCD Installation	62
2.2.6	Windows – ST Tools and Drivers Installation	62
2.2.6.1	Windows – ST-LINK Firmware Upgrade	63
2.3	Linux - Installing the Tool-Chain	64
2.3.1	Linux - Install i386 Run-Time Libraries on a 64-bit Ubuntu	65
2.3.2	Linux - Java Installation	65
2.3.3	Linux - Eclipse Installation	65
2.3.4	Linux - Eclipse Plug-Ins Installation	67
2.3.5	Linux - GCC ARM Embedded Installation	73
2.3.6	Linux - Nucleo Drivers Installation	73
2.3.6.1	Linux – ST-LINK Firmware Upgrade	74
2.3.7	Linux – OpenOCD Installation	74
2.3.8	Linux – STM32CubeMX Tool Installation	77
2.3.9	Linux – QSTLink2 Installation	79
2.4	Mac - Installing the Tool-Chain	80
2.4.1	Mac - Eclipse Installation	81
2.4.2	Mac - Eclipse Plug-Ins Installation	83
2.4.3	Mac - GCC ARM Embedded Installation	88

CONTENTS

2.4.4	Mac - Nucleo Drivers Installation	88
2.4.4.1	Mac – ST-LINK Firmware Upgrade	89
2.4.5	Mac – OpenOCD Installation	89
2.4.6	Mac STM32CubeMX Tool Installation	91
2.4.7	Mac - stlink by texane Installation	93
3.	Hello, Nucleo!	95
3.1	Get in Touch With the Eclipse IDE	95
3.2	Create a Project	99
3.3	Connecting the Nucleo to the PC	106
3.4	Flashing the Nucleo	107
3.4.1	Windows	107
3.4.2	Linux	108
3.4.3	Mac OSX	110
3.5	Understanding the Generated Code	111
4.	STM32CubeMX Tool	114
4.1	Introduction to CubeMX Tool	114
4.1.1	Pinout View	118
4.1.1.1	Chip View	118
4.1.1.2	IP Tree Pane	120
4.1.2	Clock View	122
4.1.3	Configuration View	123
4.1.4	Power Consumption Calculator View	124
4.2	Project Generation	125
4.2.1	Generate C Project with CubeMX	125
4.2.1.1	Understanding Generated Code	127
4.2.2	Create Eclipse Project	129
4.2.3	Importing Generated Files Into the Eclipse Project Manually	132
4.2.4	Importing Files Generated With CubeMX Into the Eclipse Project Automatically	137
4.3	Understanding Generated Application Code	138
4.3.1	Add Something Useful to the Firmware	143
4.4	Downloading Book Source Code Examples	144
5.	Introduction to Debugging	148
5.1	Getting Started With OpenOCD	148
5.1.1	Launching OpenOCD	149
5.1.1.1	Launching OpenOCD on Windows	150
5.1.1.2	Launching OpenOCD on Linux and MacOS X.	151
5.1.2	Connecting to the OpenOCD Telnet Console	153
5.1.3	Configuring Eclipse	154
5.1.4	Debugging in Eclipse	161

CONTENTS

5.2	ARM Semihosting	166
5.2.1	Enable Semihosting on a New Project	166
5.2.1.1	Using Semihosting With C Standard Library	169
5.2.2	Enable Semihosting on an Existing Project	172
5.2.3	Semihosting Drawbacks	173
5.2.4	Understanding How Semihosting Works	173
II Diving into the HAL		178
6.	GPIO Management	179
6.1	STM32 Peripherals Mapping and HAL <i>Handlers</i>	179
6.2	GPIOs Configuration	184
6.2.1	GPIO Mode	186
6.2.2	GPIO Alternate Function	188
6.2.3	Understanding GPIO Speed	189
6.3	Driving a GPIO	193
6.4	De-initialize a GPIO	193
7.	Interrupts Management	195
7.1	NVIC Controller	195
7.1.1	Vector Table in STM32	196
7.2	Enabling Interrupts	200
7.2.1	External Lines and NVIC	200
7.2.2	Enabling Interrupts With CubeMX	204
7.3	Interrupt Lifecycle	206
7.4	Interrupt Priority Levels	210
7.4.1	Cortex-M0/0+	210
7.4.2	Cortex-M3/4/7	215
7.4.3	Setting Interrupt Priority in CubeMX	221
7.5	Interrupt Re-Entrancy	222
7.6	Mask All Interrupts at Once or an a Priority Basis	223
8.	Universal Asynchronous Serial Communications	227
8.1	Introduction to UARTs and USARTs	227
8.2	UART Initialization	231
8.2.1	UART Configuration Using CubeMX	238
8.3	UART Communication in <i>Polling Mode</i>	239
8.3.1	Installing a Serial Console in Windows	243
8.3.2	Installing a Serial Console in Linux and MacOS X	246
8.4	UART Communication in <i>Interrupt Mode</i>	247
8.4.1	UART Related Interrupts	248
8.5	Error Management	255

CONTENTS

8.6	I/O Retargeting	257
9.	DMA Management	261
9.1	Introduction to DMA	261
9.1.1	The Need of a DMA and the Role of the Internal Buses	262
9.1.2	The DMA Controller	265
9.1.2.1	The DMA Implementation in F0/F1/F3/L1 MCUs	266
9.1.2.2	The DMA Implementation in F2/F4/F7 MCUs	270
9.1.2.3	The DMA Implementation in L0/L4 MCUs	273
9.2	HAL_DMA Module	274
9.2.1	DMA_HandleTypeDef in F0/F1/F3/L0/L1/L4 HALs	274
9.2.2	DMA_HandleTypeDef in F2/F4/F7 HALs	277
9.2.3	DMA_HandleTypeDef in L0/L4 HALs	280
9.2.4	How to Perform Transfers in Polling Mode	280
9.2.5	How to Perform Transfers in Interrupt Mode	283
9.2.6	How to Perform <i>Peripheral-To-Peripheral</i> Transfers	285
9.2.7	Using the HAL_UART Module With DMA Mode Transfers	286
9.2.8	Miscellaneous Functions From HAL_DMA and HAL_DMA_Ext Modules	289
9.3	Using CubeMX to Configure DMA Requests	290
9.4	Correct Memory Allocation of DMA Buffers	291
9.5	A Case Study: The DMA <i>Memory-To-Memory</i> Transfer Performance Analysis	292
10.	Clock Tree	297
10.1	Clock Distribution	297
10.1.1	Overview of the STM32 Clock Tree	299
10.1.1.1	The Multispeed Internal RC Oscillator in STM32L Families	303
10.1.2	Configuring Clock Tree Using CubeMX	304
10.1.3	Clock Source Options in Nucleo Boards	306
10.1.3.1	OSC Clock Supply	306
10.1.3.2	OSC 32kHz Clock Supply	307
10.2	Overview of the HAL_RCC Module	308
10.2.1	Compute the Clock Frequency at Run-Time	310
10.2.2	Enabling the <i>Master Clock Output</i>	311
10.2.3	Enabling the <i>Clock Security System</i>	311
10.3	HSI Calibration	312
11.	Timers	314
11.1	Introduction to Timers	314
11.1.1	Timer Categories in an STM32 MCU	315
11.1.2	Effective Availability of Timers in the STM32 Portfolio	317
11.2	Basic Timers	319
11.2.1	Using Timers in <i>Interrupt Mode</i>	322
11.2.1.1	Time Base Generation in <i>Advanced Timers</i>	325

CONTENTS

11.2.2	Using Timers in <i>Polling Mode</i>	325
11.2.3	Using Timers in <i>DMA Mode</i>	326
11.2.4	Stopping a Timer	328
11.2.5	Using CubeMX to Configure a <i>Basic Timer</i>	328
11.3	General Purpose Timers	329
11.3.1	Time Base Generator With External Clock Sources	329
11.3.1.1	External Clock Mode 2	331
11.3.1.2	External Clock Mode 1	335
11.3.1.3	Using CubeMX to Configure the Source Clock of a <i>General Purpose</i> Timer	340
11.3.2	Master/Slave Synchronization Modes	341
11.3.2.1	Enable Trigger-Related Interrupts	346
11.3.2.2	Using CubeMX to Configure the Master/Slave Synchronization	346
11.3.3	Generate Timer-Related Events by Software	347
11.3.4	Counting Modes	349
11.3.5	Input Capture Mode	350
11.3.5.1	Using CubeMX to Configure the Input Capture Mode	357
11.3.6	Output Compare Mode	358
11.3.6.1	Using CubeMX to Configure the Output Compare Mode	363
11.3.7	Pulse-Width Generation	363
11.3.7.1	Generating a Sinusoidal Wave Using PWM	367
11.3.7.2	Using CubeMX to Configure the PWM Mode	372
11.3.8	One Pulse Mode	373
11.3.8.1	Using CubeMX to Configure the OPM Mode	375
11.3.9	Encoder Mode	376
11.3.9.1	Using CubeMX to Configure the <i>Encoder Mode</i>	381
11.3.10	Other Features Available in <i>General Purpose</i> and <i>Advanced</i> Timers	382
11.3.10.1	<i>Hall Sensor</i> Mode	382
11.3.10.2	Combined Three-Phase PWM Mode and Other Motor-Control Related Features	383
11.3.10.3	Break Input and Locking of Timer Registers	383
11.3.10.4	Preloading of Auto-Reload Register	383
11.3.11	Debugging and Timers	384
11.4	SysTick Timer	385
11.4.1	Use Another Timer as System Timebase Source	386
11.5	A Case Study: How to Precisely Measure Microseconds With STM32 MCUs	387
12.	Analog-To-Digital Conversion	393
12.1	Introduction to SAR ADC	393
12.2	HAL_ADC Module	398
12.2.1	Conversion Modes	401
12.2.1.1	Single-Channel, Single Conversion Mode	401
12.2.1.2	Scan Single Conversion Mode	401

CONTENTS

12.2.1.3	Single-Channel, Continuous Conversion Mode	402
12.2.1.4	Scan Continuous Conversion Mode	402
12.2.1.5	Injected Conversion Mode	403
12.2.1.6	Dual Modes	403
12.2.2	Channel Selection	404
12.2.3	ADC Resolution and Conversion Speed	405
12.2.4	A/D Conversions in Polling Mode	405
12.2.5	A/D Conversions in Interrupt Mode	409
12.2.6	A/D Conversions in DMA Mode	410
12.2.6.1	Convert Multiple Times the Same Channel in DMA Mode	414
12.2.6.2	Multiple and not Continuous Conversions in DMA Mode	414
12.2.6.3	Continuous Conversions in DMA Mode	414
12.2.7	Errors Management	414
12.2.8	Timer-Driven Conversions	415
12.2.9	Conversions Driven by External Events	418
12.2.10	ADC Calibration	419
12.3	Using CubeMX to Configure ADC Peripheral	419
13.	Digital-To-Analog Conversion	422
13.1	Introduction to the DAC Peripheral	422
13.2	HAL_DAC Module	425
13.2.1	Driving the DAC Manually	426
13.2.2	Driving the DAC in DMA Mode Using a Timer	428
13.2.3	Triangular Wave Generation	432
13.2.4	Noise Wave Generation	433
14.	I²C	435
14.1	Introduction to the I ² C specification	435
14.1.1	The I ² C Protocol	437
14.1.1.1	START and STOP Condition	438
14.1.1.2	Byte Format	438
14.1.1.3	Address Frame	438
14.1.1.4	Acknowledge (ACK) and Not Acknowledge (NACK)	439
14.1.1.5	Data Frames	440
14.1.1.6	Combined Transactions	440
14.1.1.7	Clock Stretching	441
14.1.2	Availability of I ² C Peripherals in STM32 MCUs	442
14.2	HAL_I2C Module	443
14.2.1	Using the I ² C Peripheral in <i>Master Mode</i>	446
14.2.1.1	I/O MEM Operations	454
14.2.1.2	Combined Transactions	456
14.2.1.3	A Note About the Clock Configuration in STM32F0/L0/L4 families	458
14.2.2	Using the I ² C Peripheral in <i>Slave Mode</i>	458

CONTENTS

14.3	Using CubeMX to Configure the I ² C Peripheral	464
15.	SPI	466
15.1	Introduction to the SPI Specification	466
15.1.1	Clock Polarity and Phase	469
15.1.2	Slave Select Signal Management	470
15.1.3	SPI <i>TI Mode</i>	471
15.1.4	Availability of SPI Peripherals in STM32 MCUs	472
15.2	HAL_SPI Module	473
15.2.1	Exchanging Messages Using SPI Peripheral	475
15.2.2	Maximum Transmission Frequency Reachable using the CubeHAL	477
15.3	Using CubeMX to Configure SPI Peripheral	477
16.	Cyclic Redundancy Check	478
16.1	Introduction to CRC Computing	478
16.1.1	CRC Calculation in STM32F1/F2/F4/L1 MCUs	481
16.1.2	CRC Peripheral in STM32F0/F3/F7/L0/L4 MCUs	483
16.2	HAL_CRC Module	484
17.	IWDG and WWDG Timers	488
17.1	The <i>Independent Watchdog</i> Timer	488
17.1.1	Using the CubeHAL to Program IWDG Timer	489
17.2	The <i>System Window Watchdog</i> Timer	490
17.2.1	Using the CubeHAL to Program WWDG Timer	492
17.3	Detecting a System Reset Caused by a Watchdog Timer	494
17.4	Freezing Watchdog Timers During a Debug Session	494
17.5	Selecting the Right Watchdog Timer for Your Application	494
18.	Real-Time Clock	496
18.1	Introduction to the RTC Peripheral	496
18.2	HAL_RTC Module	498
18.2.1	Setting and Retrieving the Current Date/Time	499
18.2.1.1	Correct Way to Read Date/Time Values	501
18.2.2	Configuring Alarms	503
18.2.3	Periodic Wakeup Unit	505
18.2.4	Timestamp Generation and Tamper Detection	506
18.2.5	RTC Calibration	507
18.2.5.1	RTC Coarse Calibration	507
18.2.5.2	RTC Smooth Calibration	508
18.2.5.3	Reference Clock Detection	509
18.3	Using the Backup SRAM	510

III	Advanced topics	511
19.	Power Management	512
19.1	Power Management in Cortex-M Based MCUs	512
19.2	How Cortex-M MCUs Handle <i>Run</i> and <i>Sleep</i> Modes	513
19.2.1	Entering/exiting sleep modes	516
19.2.1.1	Sleep-On-Exit	518
19.2.2	<i>Sleep</i> Modes in Cortex-M Based MCUs	519
19.3	Power Management in STM32F Microcontrollers	519
19.3.1	Power Sources	520
19.3.2	Power Modes	521
19.3.2.1	Run Mode	521
19.3.2.1.1	Dynamic Voltage Scaling in STM32F4/F7 MCUs	522
19.3.2.1.2	Over/Under-Drive Mode in STM32F4/F7 MCUs	523
19.3.2.2	Sleep Mode	523
19.3.2.3	Stop Mode	524
19.3.2.4	Standby Mode	525
19.3.2.5	Low-Power Modes Example	525
19.3.3	An Important Warning for STM32F1 Microcontrollers	530
19.4	Power Management in STM32L Microcontrollers	531
19.4.1	Power Sources	531
19.4.2	Power Modes	533
19.4.2.1	Run Modes	533
19.4.2.2	Sleep Modes	535
19.4.2.2.1	Batch Acquisition Mode	536
19.4.2.3	Stop Modes	536
19.4.2.4	Standby Modes	537
19.4.2.5	Shutdown Mode	538
19.4.3	Power Modes Transitions	539
19.4.4	Low-Power Peripherals	539
19.4.4.1	LPUART	539
19.4.4.2	LPTIM	540
19.5	Power Supply Supervisors	540
19.6	Debugging in Low-Power Modes	541
19.7	Using the CubeMX Power Consumption Calculator	541
19.8	A Case Study: Using Watchdog Timers With Low-Power Modes	543
20.	Memory layout	544
20.1	The STM32 Memory Layout Model	544
20.1.1	Understanding Compilation and Linking Processes	546
20.2	The Really Minimal STM32 Application	549
20.2.1	ELF Binary File Inspection	553
20.2.2	.data and .bss Sections Initialization	555

CONTENTS

20.2.2.1	A Word About the COMMON Section	562
20.2.3	.rodata Section	563
20.2.4	Stack and Heap Regions	565
20.2.5	Checking the Size of Heap and Stack at Compile-Time	568
20.2.6	Differences With the Tool-Chain Script Files	569
20.3	How to Use the CCM Memory	571
20.3.1	Relocating the <i>vector table</i> in CCM Memory	574
20.4	How to Use the MPU in Cortex-M0+/3/4/7 Based STM32 MCUs	577
20.4.1	Programming the MPU With the CubeHAL	581
21.	Flash Memory Management	585
21.1	Introduction to STM32 Flash Memory	585
21.2	The HAL_FLASH Module	589
21.2.1	Flash Memory Unlocking	589
21.2.2	Flash Memory Erasing	589
21.2.3	Flash Memory Programming	591
21.2.4	Flash Read Access During Programming and Erasing	592
21.3	Option Bytes	592
21.3.1	Flash Memory Read Protection	594
21.4	Optional OTP and True-EEPROM Memories	596
21.5	Flash Read Latency and the ART™ Accelerator	597
21.5.1	The Role of the TCM Memories in STM32F7 MCUs	600
21.5.1.1	How to Access Flash Memory Through the TCM Interface	606
21.5.1.2	Using CubeMX to Configure Flash Memory Interface	607
22.	Bootling Process	609
22.1	The Cortex-M Unified Memory Layout and the Bootling Process	609
22.1.1	Software <i>Physical Remap</i>	610
22.1.2	Vector Table Relocation	611
22.1.3	Running the Firmware From SRAM Using the GNU ARM Eclipse Toolchain	613
22.2	Integrated Bootloader	614
22.2.1	Starting the Bootloader From the On-Board Firmware	616
22.2.2	The Bootling Sequence in the GNU ARM Eclipse Tool-chain	617
22.3	Developing a Custom Bootloader	620
22.3.1	<i>Vector Table</i> Relocation in STM32F0 Microcontrollers	631
22.3.2	How to Use the <code>flasher.py</code> Tool	634
23.	Running FreeRTOS	637
23.1	Understanding the Concepts Underlying an RTOS	638
23.2	Introduction to FreeRTOS and CMSIS-RTOS Wrapper	644
23.2.1	The FreeRTOS Source Tree	645
23.2.1.1	How to Import FreeRTOS Manually	646
23.2.1.2	How to Import FreeRTOS Using CubeMX and CubeMXImporter	647

CONTENTS

23.2.1.3	How to Enable FPU Support in Cortex-M4F and Cortex-M7 Cores	649
23.3	Thread Management	649
23.3.1	Thread States	652
23.3.2	Thread Priorities and Scheduling Policies	653
23.3.3	Voluntary Release of the Control	657
23.3.4	The <i>idle</i> Thread	657
23.4	Memory Allocation and Management	658
23.4.1	Dynamic Memory Allocation Model	659
23.4.1.1	heap_1.c	660
23.4.1.2	heap_2.c	661
23.4.1.3	heap_3.c	661
23.4.1.4	heap_4.c	661
23.4.1.5	heap_5.c	662
23.4.1.6	How to Use malloc() and Related C Functions With FreeRTOS	662
23.4.1.7	FreeRTOS Heap Definition	663
23.4.2	Static Memory Allocation Model	663
23.4.2.1	<i>idle</i> Thread Allocation With Static Memory Allocation Model	664
23.4.3	Memory Pools	664
23.4.4	Stack Overflow Detection	666
23.5	Synchronization Primitives	668
23.5.1	Message Queues	668
23.5.2	Semaphores	672
23.5.3	Thread Signals	675
23.6	Resources Management and Mutual Exclusion	676
23.6.1	Mutexes	676
23.6.1.1	The Priority Inversion Problem	677
23.6.1.2	Recursive Mutexes	678
23.6.2	Critical Sections	679
23.6.3	Interrupt Management With an RTOS	680
23.6.3.1	FreeRTOS API and Interrupt Priorities	681
23.7	Software Timers	682
23.7.1	How FreeRTOS Manages Timers	683
23.8	A Case Study: Low-Power Management With an RTOS	684
23.8.1	The <i>idle</i> Thread Hook	684
23.8.2	The Tickless Mode in FreeRTOS	686
23.8.2.1	A Schema for the <i>tickless</i> Mode	688
23.8.2.2	A Custom <i>tickless</i> Mode Policy	691
23.9	Debugging Features	699
23.9.1	configASSERT() Macro	699
23.9.2	Run-Time Statistics and Thread State Information	700
23.10	Alternatives to FreeRTOS	703
23.10.1	ChibiOS	704

CONTENTS

23.10.2	Contiki OS	704
23.10.3	OpenRTOS	705
24.	Advanced Debugging Techniques	706
24.1	Understanding Cortex-M Fault-Related Exceptions	706
24.1.1	The Cortex-M Exception Entrance Sequence and the ARM Calling Convention	708
24.1.1.1	How the GNU ARM Eclipse Tool-chain Handles Fault-Related Exceptions	713
24.1.1.2	How to Interpret the Content of the LR Register on Exception Entrance	715
24.1.2	Fault Exceptions and Faults Analysis	716
24.1.2.1	<i>Memory Management</i> Exception	716
24.1.2.2	<i>Bus Fault</i> Exception	717
24.1.2.3	<i>Usage Fault</i> Exception	718
24.1.2.4	<i>Hard Fault</i> Exception	719
24.1.2.5	Enabling Optional Fault Handlers	720
24.1.2.6	Fault Analysis in Cortex-M0/0+ Based Processors	720
24.2	Eclipse Advanced Debugging Features	721
24.2.1	Expressions	721
24.2.1.1	Memory Monitors	722
24.2.2	Watchpoints	723
24.2.3	Instruction Stepping Mode	724
24.2.4	Keil Packs and Peripheral Registers View	725
24.2.5	Core Registers View	728
24.3	Debugging Aids From the CubeHAL	729
24.4	External Debuggers	729
24.4.1	Using SEGGER J-Link for ST-LINK Debugger	731
24.4.2	Using the ITM Interface and SWV Tracing	735
24.5	STM Studio	736
24.6	Debugging two Nucleo Boards Simultaneously	738
25.	FAT Filesystem	741
25.1	Introduction to FatFs Library	741
25.1.1	Using CubeMX to Include FatFs Library in Your Projects	744
25.1.1.1	The <i>Generic Disk Interface</i> API	745
25.1.1.2	The Implementation of a Driver to Access SD Cards in SPI Mode	746
25.1.2	Relevant FatFs Structures and Functions	747
25.1.2.1	Mounting a Filesystem	747
25.1.2.2	Opening a File	747
25.1.2.3	Reading From/Writing Into a File	748
25.1.2.4	Creating and Opening a Directory	749
25.1.3	How to Configure the FatFs Library	752

CONTENTS

26. Develop IoT Applications	755
26.1 Solutions Offered by STM to Develop IoT Applications	756
26.2 The W5500 Ethernet Controller	758
26.2.1 How to Use the W5500 Shield and the ioLibrary_Driver Module	762
26.2.1.1 Configuring the SPI Interface	765
26.2.1.2 Configuring the Socket Buffers and the Network Interface	766
26.2.2 Socket APIs	768
26.2.2.1 Handling Sockets in TCP Mode	770
26.2.2.2 Handling Sockets in UDP Mode	770
26.2.3 I/O Retargeting to a TCP/IP Socket	771
26.2.4 Setting up an HTTP Server	773
26.2.4.1 A Web-Based Oscilloscope	776
27. Getting Started With a New Design	789
27.1 Hardware Design	789
27.1.1 PCB Layer Stack-Up	790
27.1.2 MCU Package	791
27.1.3 Decoupling of Power-Supply Pins	792
27.1.4 Clocks	794
27.1.5 Filtering of RESET Pin	795
27.1.6 Debug Port	795
27.1.7 Boot Mode	797
27.1.8 Pay attention to “pin-to-pin” Compatibility...	798
27.1.9 ...And to Selecting the Right Peripherals	799
27.1.10 The Role of CubeMX During the Board Design Stage	799
27.1.11 Board Layout Strategies	802
27.2 Software Design	803
27.2.1 Generating the binary image for production	803

Appendix **806**

A. Miscellaneous HAL functions and STM32 features	807
Force MCU reset from the firmware	807
STM32 96-bit Unique CPU ID	807

B. Troubleshooting guide	809
GNU ARM Eclipse Installation Issues	809
Eclipse related issue	809
Eclipse cannot locate the compiler	810
Eclipse continuously breaks at every instruction during debug session	811
The step-by-step debugging is really slow	811
The firmware works only under a debug session	812

CONTENTS

STM32 related issue	812
The microcontroller does not boot correctly	812
It is Not Possible to Flash or to Debug the MCU	814
C. Nucleo pin-out	816
Nucleo-F446RE	817
Arduino compatible headers	817
Morpho headers	817
Nucleo-F411RE	818
Arduino compatible headers	818
Morpho headers	818
Nucleo-F410RB	819
Arduino compatible headers	819
Morpho headers	819
Nucleo-F401RE	820
Arduino compatible headers	820
Morpho headers	820
Nucleo-F334R8	821
Arduino compatible headers	821
Morpho headers	821
Nucleo-F303RE	822
Arduino compatible headers	822
Morpho headers	822
Nucleo-F302R8	823
Arduino compatible headers	823
Morpho headers	823
Nucleo-F103RB	824
Arduino compatible headers	824
Morpho headers	824
Nucleo-F091RC	825
Arduino compatible headers	825
Morpho headers	825
Nucleo-F072RB	826
Arduino compatible headers	826
Morpho headers	826
Nucleo-F070RB	827
Arduino compatible headers	827
Morpho headers	827
Nucleo-F030R8	828
Arduino compatible headers	828
Morpho headers	828
Nucleo-L476RG	829
Arduino compatible headers	829

CONTENTS

Morpho headers	829
Nucleo-L152RE	830
Arduino compatible headers	830
Morpho headers	830
Nucleo-L073R8	831
Arduino compatible headers	831
Morpho headers	831
Nucleo-L053R8	832
Arduino compatible headers	832
Morpho headers	832
D. STM32 packages	833
LFBGA	833
LQFP	833
TFBGA	834
TSSOP	834
UFBGA	834
UFQFPN	835
VFQFP	835
WLCSP	835
E. History of this book	837
Release 0.1 - October 2015	837
Release 0.2 - October 28th, 2015	837
Release 0.2.1 - October 31th, 2015	837
Release 0.2.2 - November 1st, 2015	838
Release 0.3 - November 12th, 2015	838
Release 0.4 - December 4th, 2015	838
Release 0.5 - December 19th, 2015	838
Release 0.6 - January 18th, 2016	839
Release 0.6.1 - January 20th, 2016	839
Release 0.6.2 - January 30th, 2016	839
Release 0.7 - February 8th, 2016	839
Release 0.8 - February 18th, 2016	840
Release 0.8.1 - February 23th, 2016	840
Release 0.9 - March 27th, 2016	840
Release 0.9.1 - March 28th, 2016	841
Release 0.10 - April 26th, 2016	841
Release 0.11 - May 27th, 2016	841
Release 0.11.1 - June 3rd, 2016	842
Release 0.11.2 - June 24th, 2016	842
Release 0.12 - July 4th, 2016	842
Release 0.13 - July 18th, 2016	843

CONTENTS

Release 0.14 - August 12th, 2016	843
Release 0.15 - September 13th, 2016	843
Release 0.16 - October 3th, 2016	844
Release 0.17 - October 24th, 2016	844
Release 0.18 - November 15th, 2016	844
Release 0.19 - November 29th, 2016	845
Release 0.20 - December 28th, 2016	845
Release 0.21 - January 29th, 2017	845
Release 0.22 - May 2nd, 2017	845
Release 0.23 - July 20th, 2017	846