
Contents

Preface	xi
Chapter 1. Aims, Context and a Guiding Thread Example	1
1.1. Aims of this book	1
1.2. What this book contains... and what it does not	2
1.3. About the code examples	3
1.4. Choosing your μ controller: the STM32 family	3
1.4.1. Criteria when choosing a μ controller	4
1.4.2. STM32 family	5
1.4.3. STM32F10x line	6
1.5. Guiding thread example	8
1.5.1. Context: photovoltaic self-consumption installation	8
1.5.2. System: optimization of electric heating in a photovoltaic self-consumption household	10
1.5.3. Summary of the peripherals seen in the guiding thread example	13
Chapter 2. General Programming Principles	15
2.1. Pointers and registers	15
2.1.1. “Simple” variables	15
2.1.2. Pointer to a variable	16
2.1.3. Using pointers to pass arguments to functions	18
2.1.4. When to use a pointer?	20
2.1.5. Function pointers	21
2.2. C language structure and usage in the STM32	22
2.2.1. Structured type organization	22
2.2.2. Accessing registers via a structure	24
2.3. Case of nested structures	25

2.4. Masks usage	27
2.4.1. General considerations	28
2.4.2. Masks for setting bits to 1	29
2.4.3. Masks for setting bits to 0	30
2.4.4. Bit inversion	31
2.4.5. Extension and generalization to bit field assignment	31
2.5. Code structuring and the concept of drivers	33
2.6. Interrupt routines	36
2.6.1. Handler routines concepts	36
2.6.2. Interrupt routine implementation	37
2.6.3. Redirection via dynamic programming	39
Chapter 3. General STM32F10x Hardware Considerations	43
3.1. Global STM32 architecture	43
3.2. Cortex-M3 components	44
3.3. Clock trees and RCC units	47
3.3.1. Generation of primary clocks	47
3.3.2. Generation of secondary clocks	49
3.3.3. RCC unit programming	51
3.3.4. RCC unit configuration coding example	55
3.4. <i>Watchdogs</i>	57
3.4.1. Independent <i>watchdogs</i>	58
3.4.2. Windowed <i>watchdogs</i>	59
Chapter 4. Binary Inputs/Outputs (I/O): Parallel Ports	61
4.1. Concept of I/O ports	61
4.2. Electronic aspect	62
4.2.1. Global architecture	63
4.2.2. Input configuration	65
4.2.3. Output configuration	66
4.3. STM32 GPIOs	67
4.3.1. Configuration registers	69
4.3.2. Access registers	70
4.4. Example	73
4.4.1. Proposed solution	75
4.4.2. Comments	80
4.5. Unaddressed points regarding GPIOs	81
Chapter 5. Interrupt and DMA Management	83
5.1. General exceptions and particular interrupts	83
5.2. <i>Resets</i> and interrupt vector tables	84
5.2.1. Stack pointers and program counters	84

5.2.2. IVT contents	85
5.3. Principle of redirection and the role of the NVIC	88
5.3.1. NVIC	88
5.3.2. Conditions for an interrupt to be accepted	90
5.3.3. Priority management	92
5.3.4. Redirection process steps	96
5.4. Case of external interrupt inputs	96
5.4.1. AFIO and EXTI peripherals	97
5.4.2. NVIC level	98
5.4.3. GPIO and AFIO levels	99
5.4.4. EXTI level	102
5.5. Guiding thread example: treating an EXTI exception	103
5.5.1. Problem to solve	103
5.5.2. Proposed solution	103
5.6. Direct memory access	105
5.6.1. Generalities	105
5.6.2. DMA on the STM32F10x	106
5.7. Guiding thread example: case of DMA programming	111
5.7.1. UART configuration	112
5.7.2. DMA configuration	112
5.7.3. Program structure	112
Chapter 6. Timers	115
6.1. General information on counters or Timers	115
6.1.1. Basic principle	116
6.1.2. Timer units in the STM32F10x	116
6.1.3. Different types of Timers	117
6.2. Operating modes	118
6.2.1. General structure of a Timer	119
6.2.2. <i>Core</i> block: overflow, events and frequency	120
6.2.3. <i>Capture/Compare</i> block	125
6.2.4. <i>Trigger</i> controller block and encoder interface	129
6.2.5. <i>Master</i> mode	139
6.3. An additional Timer: the SysTick	141
6.4. Guiding thread example	143
6.4.1. Case study: a control knob	143
6.4.2. Code extract	143
Chapter 7. Pulse-Width Modulation (PWM)	145
7.1. Generalities on PWM-type signals	145
7.1.1. Definition	145
7.1.2. Use in information transmission	147
7.1.3. Use in digital-to-analog conversion	148

7.2. Implementation of PWM in an STM32F10x μ controller	154
7.2.1. Basic operation	154
7.2.2. Configuration and options	156
7.3. Guiding thread example	160
7.3.1. First possible solution	160
7.3.2. Second adopted solution	160
7.3.3. Interrupt configuration	160
7.3.4. The PWM configuration itself	161
7.3.5. The GPIO configuration	161
7.3.6. Code excerpt	161
Chapter 8. Analog to Digital Converter	165
8.1. Generalities and operating principles	165
8.1.1. Role and definitions	165
8.1.2. Successive-approximation ADC	168
8.1.3. Variable voltage measurement	169
8.1.4. Multi-channel ADC	172
8.1.5. Overview of the ADC <i>timings</i>	173
8.2. ADC in the STM32F10x	174
8.2.1. Simplified architecture of the ADC	174
8.2.2. ADC functional modes	181
8.2.3. DMA with ADC	185
8.3. Application: the power analyzer	186
8.3.1. Determining the sampling duration T_s	186
8.3.2. Code excerpt for configuration	187
Chapter 9. Some Communication Buses	191
9.1. Introduction	191
9.2. UART units	193
9.2.1. Generalities	193
9.2.2. Reading/writing techniques	195
9.2.3. UART configuration on the STM32F10x	197
9.3. SPI units	203
9.3.1. Generalities	203
9.3.2. Shift register-based read/write techniques	205
9.3.3. SPI configuration on the STM32F10x	209
9.4. I ² C bus	213
9.4.1. Generalities	213
9.4.2. Electrical topology of the I ² C bus	215
9.4.3. Principle and protocol	216
9.4.4. Reading/writing techniques	220
9.4.5. I ² C configuration on the STM32F10x	224

9.5. Guiding thread example	233
9.5.1. UART reception in interrupt mode	233
9.5.2. Temperature measurement via the I ² C bus and DMA	234
Chapter 10. STM32 Power Management	241
10.1. Introduction	241
10.2. Domains of the μ controller	242
10.2.1. 1.8V Domain	242
10.2.2. V_{DD} Domain	244
10.2.3. Backup Domain	244
10.2.4. V_{DDA} Domain	245
10.3. Reset on STM32	245
10.3.1. System Reset	245
10.3.2. Power Reset	247
10.3.3. Backup Domain Reset	247
10.3.4. Reset flags (RCC_CSR) management	247
10.3.5. STM32F10x reset summary	248
10.4. RTC registers	249
10.4.1. Secure access to the Backup Domain	250
10.4.2. RCC_BDCR registers	250
10.4.3. RTC register structure	251
10.4.4. Reading/writing RTC registers	252
10.5. Low Power modes	254
10.5.1. Event and interrupt	254
10.5.2. Key bits for Low Power modes	255
10.5.3. Sleep mode	258
10.5.4. Stop mode	259
10.5.5. Standby mode	260
10.6. Software architecture and using Low Power modes	261
10.6.1. Classical software architecture	261
10.6.2. Sleeping in main(), WFI instruction	262
10.6.3. Automatic sleep upon interrupt exit	262
10.6.4. Sleeping in main(), WFE instruction	264
10.6.5. Standby mode on RTC alarm	264
10.7. Guiding thread example for Low Power mode	264
10.7.1. Problem to solve	264
10.7.2. Proposed solution	265
Appendix	271
References	283
Index	285