Errata of

Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C

Third Edition

2nd Printing (June 2018)

ISBN-10: 0982692668

Yifeng Zhu

Correction Date: November 9, 2020

Thank you all for providing me feedbacks and corrections!

Chapter 1. See a Program Running

- Page 22, bullet list #2, the binary code for memory address 0x08000162 is 0x680A, not 0x680.
- Page 22, "This instruction loads the value of variable *a* into register r1." It should be `r2`.

Chapter 2. Data Representation

Chapter 3. ARM Instruction Set Architecture

Chapter 4. Arithmetic and Logic

Chapter 5. Load and Store

Chapter 6. Branch and Conditional Execution

Chapter 7. Structured Programming

Pg. 145, first paragraph

"Variables *i*, *maxLocation*, and *maxValue* are local variables and are stored in r2, r0, and r1, respectively." should be

"Variables *i*, *maxLocation*, and *maxValue* are local variables and are stored in r2, r1, and r0, respectively."

Chapter 7. Structured Programming

Chapter 8. Subroutines

- Page 191,

```
LDR r5, [#sp, #20] ; r5 = mem[sp + 20] = 5
LDR r6, [#sp, #24] ; r6 = mem[sp + 24] = 6
```

should be

```
LDR r5, [sp, #20] ; r5 = mem[sp + 20] = 5
LDR r6, [sp, #24] ; r6 = mem[sp + 24] = 6
```

Chapter 9. 64-bit Data Processing

Chapter 10. Mixing C and Assembly

Chapter 11. Interrupt

Chapter 12. Fixed-point and Floating-point Arithmetic

Chapter 13. Instruction Encoding and Decoding

Chapter 14. Generic-purpose I/O

- On Page 355, the demo code given in the middle

```
ORR r1, r1, #(1<<6)          ; Set bit 6
```

should be

```
ORR r1, r1, #(1<<2)          ; Set bit 2
```

- On Page 363, Example 14-6,

Incorrect code	Correct code

`void TIM4_IRQHandler(void) {`
`  ...`
`  if((GPIOA->IDR & 0x1) == 0x1){  // check input on pin PA.0`
`  counter++;  // button is pressed`
`  if (counter >= 4) {`
`  pressed = 1;  // set the flag`
`  counter = 0;  // reset counter`
`  } else { // button is not pressed`
`  counter = 0;  // reset counter`
`  }`
`  }`
`}`
	`void TIM4_IRQHandler(void) {`
	`  ...`
	`  if((GPIOA->IDR & 0x1) == 0x1){  // check input on pin PA.0`
	`  counter++;  // button is pressed`
	`  if (counter >= 4) {`
	`  pressed = 1;  // set the flag`
	`  counter = 0;  // reset counter`
	`  }`
	`  } else { // button is not pressed`
	`  counter = 0;  // reset counter`
	`  }`
	`}`

Chapter 15. General-purpose Timers

- Page 383, in the code given in Example 15-3, "// Enable ~~TIM4~~ TIM1 interrupt in NVIC"
- Page 379, at the bottom, removing "driving the timer is 2.097 MHz."
- Page 396, "The difference between two consecutive transitions measures an elapsed time span, as shown in Figure ~~14-19~~ **15-19**."

Chapter 16. Stepper Motor Control

Chapter 17. Liquid-crystal Display (LCD)

- Page 440, caption of Table 17-2, "encoding of five letters (A-Z)" should be "encoding of five letters (A-E)".
- Page 442, Table 17-3 should be:

Segments	**1G**	**1B**	**1M**	**1E**	
LCD_RAM[0]	Bit 3	Bit 22	Bit 23	Bit 4	C[0]
Segments	**1F**	**1A**	**1C**	**1D**	
LCD_RAM[2]	Bit 3	Bit 22	Bit 23	Bit 4	C[1]
Segments	**1Q**	**1K**	**1Colon**	**1P**	
LCD_RAM[4]	Bit 3	Bit 22	Bit 23	Bit 4	C[2]
Segments	**1H**	**1J**	**1DP**	**1N**	
LCD_RAM[6]	Bit 3	Bit 22	Bit 23	Bit 4	C[3]

- Page 442, the code immediately after Table 17-3 is correct but its commends should follow the above corrected Table 17-3.

Chapter 18. Real-time Clock (RTC)

Chapter 19. Direct Memory Access (DMA)

Chapter 20. Analog-to-Digital Converter

- Page 265, Example 11-13, "`EXTI->FTSR &= ~EXTI_FTSR_RT3;`" should be `EXTI->FTSR &= ~EXTI_FTSR_FT3;`

Chapter 21. Digital-to-Analog Converter

- Page 519, ~~Example 11-7~~ Example 21-7 gives a simplified C implementation.
- Page 522,~~ Example 21-9~~ Example 21-10 shows the amplitude-modulating signal based on the ADSR envelope. ~~Figure 20-12~~ Example 21-11 presents the final modulated sinusoidal wave signal used to drive a speaker or headphones.

Chapter 22. Serial Communication Protocols

- Page 529, "0xE1, the bit stream 1000~~10~~111 (read from left to right)"
- Page 531, "The hex equivalent of ~~1667~~ 16667 is 0x411B."
- Page 550, last sentence, "As shown in ~~Table 24-4~~ Table 22-4 and ~~Table 24-5~~ Table 22-5"
- Page 576, in Example 22-27, Send data to an SPI slave
  1. `SPIx->DR = txBuffer[i];`
     `should be: *((volatile uint8_t*)&SPIx->DR) = txBuffer[i];`
  2. `rxBuffer[i] = SPIx->DR;`
     `should be: rxBuffer[i] = *((volatile uint8_t*)&SPIx->DR);`

- Page 577, in Example 22-28, Receive data from an SPI slave
  1. `SPIx->DR = 0xFF;  // A dummy byte`
     `should be: should be: *((volatile uint8_t*)&SPIx->DR) = 0xFF`
  2. `rxBuffer[i] = SPIx->DR;`
     `should be: rxBuffer[i] = *((volatile uint8_t*)&SPIx->DR);`

Chapter 23. Multitasking

- Page 405 and 406,
  run the ~~pseudo~~ instruction "`CPSID I`"
  the ~~pseudo~~ instruction "`CPSIE I`"

Chapter 24. Digital Signal Processing