Errata

I regret that the following errors have come to my notice. The corrections are listed with the names of those who reported them, to whom I am very grateful. Please send a note to me at jdavies@elec.gla.ac.uk if you find others.

Widespread corrections

1. **Symbols for 'kilo'**. When I wrote the book I was careful to use a low-case 'k' for 'decimal kilo', standing for $10^3 = 1000$, and an upper-case 'K' for 'binary Kilo', $2^{10} = 1024$. This is explained on page 6. Unfortunately the publisher changed all kHz to KHz after I had returned the proofs, destroying the distinction. A sentence on page 313 has become meaningless and should read: "Note the capital K because this is a binary Kilo, meaning $2^{10} = 1024$, rather than the decimal kilo of 1000 as in kHz." Other occurrences of KHz should be changed to kHz on the following pages, and probably many others that I have not yet noticed: 449, 451, 459, 461, 491, 492, 508, 517, 536, 537, 545, 546, 547 (in the code, which appals me), 550, 556, 557, 558. (2009 February 01)

2. **Sinc function**. The publisher changed the function $\mathrm{sinc}(x)$ to $\mathrm{sin\,c}(x)$ in numerous places, notably on pages 441–445. (2009 February 01)

3. **Names of bit fields for input/output ports**. IAR have changed the names for the bit fields for input/output ports in version 5.40 of Embedded Workbench (I am not sure when the change occurred). Each bit field had a unique name when I wrote the book. For example, bit 0 of `P1IN` was called `P1IN_0` and bit 0 of `P1OUT` was `P1OUT_0`. Thus `P1OUT.P1OUT_0 = 1` would set bit 0 of `P1OUT`. Now the names of the fields have been abbreviated to `P0`, `P1` and so on, which are common to all registers. The new usage

is `P1OUT.P0 = 1` to set bit 0 of `P1OUT`. The complete reference to the field remains unique and the new names are more concise but the change affects a huge number of programs throughout the book. Fortunately the bit fields for the `TACTL` register, which is used as an example to introduce bit fields on page 49, have not been changed. (Andi Antonius, 2012 February 02)

Individual corrections

1. Page 12, description of EPROM. I wrote that EPROM stands for electrically programmable read-only memory, which is incorrect: It should be *erasable* programmable read-only memory. The description that follows is poorly worded as well and would have been better as 'it can be programmed electrically but not erased *electrically*'. (Andrew Beaulieu, 2010 May 24)

2. Page 28, figure 2.4 for the memory map. The third address up should be 0x0010, not 0x0100. (Andrew Bovill, 2010 January 22)

3. Pages 71–73, `ledson.c` and `ledson.s43` programs. In the C program I write to P2DIR before P2OUT but in the assembly language it is done in the opposite sequence. The reason is that I used the 'obvious' method in the first program, `ledson.c`, and the better approach the second time in `ledson.s43`. This is explained on page 75 but it would probably have been better to do it correctly from the start. (Ramakrishna, 2008 October 27)

4. Page 80, last sentence but one before section 4.4. This should read: The format of the header file is different and so is the directive to include it, it does not permit absolute assembly at all (no `ORG` directives!) and executable code is introduced with `.text` instead of `RSEG CODE`. The publisher incorrectly moved the period before `.text`. (2009 March 10)

5. Page 82, figure 4.5(a). The text in the two rectangular boxes on the left should be interchanged. The top box should contain 'turn LED on, clear P2OUT.3' and the bottom should contain 'turn LED off, set P2OUT.3'. I should include a reminder in the caption that the LED is active low; the button is active low as well but that is typical. (Richard Lyons, 2011 June 13)

6. Page 88, section 4.4.4, bit fields in listing 4.8. IAR have changed the names of bit

fields for input/output ports and the listing is no longer correct. See *Widespread corrections* above. The definitions for the LED and button should be changed to

```
#define LED1    P2OUT_bit.P3
#define B1      P2IN_bit.P1
```

and the line to set the direction of P2.3 should be

```
    P2DIR_bit.P3 = 1;           // Set pin with LED1 to output
```

Similar changes are required for almost all subsequent programs; I've flagged this one because it is the first to use bit fields rather than masks. (Andi Antonius, 2012 February 02)

7. Page 131, section on 'Constant Generator and Emulated Instructions'. This section might have been clearer if I had listed all six constants available using SR/CG1, CG2 and the four addressing modes for a source. A table is provided in the section 'Constant Generator Registers CG1 and CG2' of the chapter 'RISC 16-bit CPU' of the Family User's Guides.

8. Page 151, listing 5.2. The publisher removed a semicolon from the second line of the CopyLoop, which makes the comments meaningless. It should read as follows.

```
CopyLoop:
; This ought to be sufficient but the assembler complains...
;   mov.b   @R14+,BeginDest-BeginSource-1(R14)  ; -1 allows for @R14+
; ...so I had to handle the wrapping around myself
    mov.b   @R14+,BeginDest-BeginSource+0xFFFF(R14)
                                ; 0xFFFF allows for increment in @R14+
```

The point is that the assembler does not accept '-1', which must be represented as '+0xFFFF' instead.

I am horrified that the publisher chose to edit my programs without asking. The book was composed using LaTeX, which read the programs from the source files used by EW430. The printed listings therefore ought to match my code exactly. (2009 March 10)

9. Page 191, listing 6.4. This includes the line

```
    mov.w   #CCIE,&TACCTL0          ; Enable interrupts on Compare 0
```

Note that I use the `mov` instruction to write to the complete register, rather than `bis` to set the CCIE flag alone. An important difference is that `mov` clears all the other bits in the register at the same time, including CCIFG. This action avoids an unintended interrupt that would occur if CCIFG had become set before interrupts were enabled. The issue is mentioned on page 197 in the section 'Configure Interrupts Carefully'.

More care is needed if the interrupt flag is not in the same register as its enable bit. This applies to modules such as the watchdog and basic timers, whose bits are in the IFGn and IEn registers. I have not always remembered to clear the flag in IFGn before enabling the interrupt with IEn! (2009 July 21)

10. Page 196, final paragraph of section 6.8.3 on Nonmaskable Interrupts. The first sentence describes the treatment of *maskable* interrupts to emphasize that nonmaskable interrupts are handled differently. This should be clearer. (Tamas Hornos, 2009 July 21)

11. Page 204, caption to listing 6.8. The caption mentions low-power mode 0 but it should be low-power mode 3. (Tamas Hornos, 2009 July 21)

12. Page 210, description of pull resistors. The warning about the pull resistors should be more comprehensive. Suppose that some pins of a port are used for output and others are used for input with pull resistors. After the port has been configured you must write *only* to those bits of PnOUT that correspond to output pins when you want to change the output. Usually it doesn't matter if you write to the bits of PnOUT for input pins as well because it has no effect. However, it can change the pull resistors from pullup to pulldown or *vice versa* if these are enabled. This is easily forgotten! (Andy Palm, 2010 August 10)

13. Page 212, final paragraph, second sentence. This states 'Thus there is no direct electrical connection between the *source* and channel' but should say 'Thus there is no direct electrical connection between the *gate* and channel'. (Lawrence Normie, 2011 July 03)

14. Page 214, before example 7.2. The text says that 'The output is pulled to V_{SS} by the p-MOSFET' but it should be V_{DD} rather than V_{SS}. (Ramakrishna, 2008 October 27)

15. Page 272, section 7.9.5. IAR introduced a new calling convention with version 4.x of the C compiler for EW430. This affects the way in which parameters are passed to a function and the value is returned. Fortunately it does not affect the simple function

described in this section. However, this is a reminder of the hazards of mixing the assembly and C languages, as I warn you in the text. (2009 January 01)

16. Page 308, equation (8.2). The publisher changed the denominator to 32,768 but it should be 32.768. (2009 January 01)

17. Page 308–312, section on 'How should the program be organised?' I give two programs for the reaction timer with contrasting structures: Listing 8.7 has all the action in interrupt service routines, while listing 8.8 has minimal interrupt service routines with most work done in the main routine. The discussion on page 310 between the two listings includes the unfortunate sentence 'I normally prefer to put the actions in ISRs but here the overall strategy is much easier to follow when it is in the main loop.' I should have made it clear that the comment about putting the actions in ISRs was meant to apply only to trivial programs, although of course that includes most if not all of the examples in the book. The general rule, which I set out on page 196, is to *keep interrupt service routines short*. Listing 10.10 on page 565 has a better structure, where the interrupt service routines for the USCI_B wake the main routine and pass a value to show what action is needed. (Ciarán Mac Aonghusa, 2009 January 27)

18. Page 313, sentence beginning 'Note the capital K…'. This has been rendered meaningless by the publisher. See the 'widespread correction' concerning symbols for kilo. (2009 February 01)

19. Pages 319–320, listing 8.10. Note that the registers are increased by the *full* value of each delay in Continuous mode. This contrasts with Up mode, where the period is TACCR0 + 1 so the register TACCR0 is set to 1 less than the desired delay. (2009 January 01)

20. Page 330, top line. The frequency should be 438.9 Hz rather than 877.7 Hz. I gave the frequency of toggling rather than that of the sound. (2009 January 01)

21. Page 375, equation (9.1). This should appear as follows.

$$R(T) = R_0 \exp\left(\frac{B}{T} - \frac{B}{T_0}\right). \tag{9.1}$$

(2009 February 01)

22. Page 400, figure 9.12(a). The label on the plot says $f = 330\,\text{Hz}$ but it should be 310 Hz, consistent with the caption and text. (2011 February 15)

23. Page 443, sentence beginning 'For a second-order sigma–delta converter'. This could be more clearly set out as 'the averaging is usually spread over $3 \times$ OSR samples'. (2009 February 01)

24. Page 461, listing 9.10. The purpose of the line with the comment 'To reduce bias when dividing' is explained on page 422; I should have provided a cross-reference. (Tamas Hornos, 2009 July 21)

25. Page 466, listing 9.11. I forgot to describe the circuit used with this program. Two input voltages are needed because the SD16_A is used in differential mode. One comes from a potential divider incorporating a potentiometer, as in figure 9.17(a) on page 414; the other comes from a fixed potential divider. I used a home-made demonstration board, which is described fully in another document (demoboards.pdf) that can be downloaded from the web site. (Tamas Hornos, 2009 September 15)

26. Page 489, listing 9.13 for `dacdma2.c`. This has several problems, mostly because IAR have changed some of the definitions in the header file for the device since the book was written.

    - The 'include' file for utility functions should be `LCDutils2.h` (and the corresponding code `LCDutils2.c` should be added to the project); the '2' was missing.
    - The added definition for DMA0SZ should be deleted because it is provided by current versions of the IAR header file for the device.
    - Some of the constants associated with the DMA0CTL register have gained a '0' in their names in recent versions of the header file, which affects both instructions that write to this register. The revised names are `DMA0DSTBYTE`, `DMA0SRCBYTE` and `DMA0CTL_bit.DMA0EN`.

    I have added an updated program `dacdma2a.c` to the web site and checked it with EW4.21.8 (FET_R522, slac050w). (David M Schwartz, 2010 February 14)

27. Page 504, section 10.3. Another bug (USI5) has been reported for the Universal Serial Interface in SPI mode. See the latest *MSP430F20xx Device Erratasheet* (slaz026). The USI5 bug affects masters that are configured with USICKPH = 1 (CPHA = 0, typically SPI mode 0). Fortunately my example of the USI as an SPI master in section 10.5 uses mode 3 and is therefore not upset by this bug. (2009 January 01)

28. Page 544, paragraph beginning 'The USCI has two interrupt vectors'. The first sentence should read 'The USCI_B has two interrupt vectors, both of which are shared with the USCI_A'. (2009 February 01)

Further reading

Here are some useful books that have been published since my book went to press.

[1] Baugh, Tom. *MSP430 State Machine Programming: with the ES2274.* Suwanee, GA: SoftBaugh Inc, 2008. ISBN 0975475928 (2009 January 01)

[2] Edwards, Lewin A. R. W. *Analyzing and Developing Real-Time Code: Texas Instruments MSP430 and ARM9 using Rowley Crossworks.* Boston, MA: Newnes, 2008. ISBN 0750686014 (2009 January 01)

[3] Simone, Lisa K. *If I only changed the software, why is the phone on fire?* Burlington, MA: Newnes, 2007. ISBN 0750682183. This isn't about the MSP430 in particular but describes methods for debugging embedded systems in general, presented as detective stories. Section 10.14.2 of my book, where I measure the load on the CPU imposed by a software UART, should prepare you to solve one of Simone's mysteries! I enjoyed this book greatly. (2011 November 14)