Erratum

Chapter 3

Page 38: since the Texas Instruments™ products mentioned in the **Remark** are out on the market now, this paragraph should be modified as follows:

Remark: all the examples proposed in this book refer to the hardware presented in this chapter. It must be said that Texas Instruments™ started the production of new LaunchPads™ F280025C and F280049C and new BoosterPacks BOOSTXL-DRV8320RS and BOOSTXL-DRV8323RS, which are available along with the LaunchPad™ F28069M and BOOSTXL-DRV8301 on the market. Even if a new hardware will imply differences in the settings of the programming environment as well as some modifications in the pinout of the adopted boards, the proposed design approach/workflow of closed-loop control strategies for power electronic applications is still valid. This holds also for the LaunchPad™ F28379D. The reader is referenced to [7] for checking the compatibility between LaunchPads™ and BoosterPacks and [31] to verify the Embedded Coder® support of the adopted MCU board from Simulink® blockset.

Chapter 8

Page 101: it is important to note that, starting from newer releases of MATLAB$^{®}$ (later than 2021a), the generation of the virtual COM port is automatically performed at the connection of the board to the host PC via USB cable. However, it is of paramount importance to check the correct generation of the COM port in the **Device Manager** window.

Chapter 13

Page 184: it is worth mentioning another setting that can be edited after **CMPA initial value**.

- **Reload for Compare A Register (SHDWAMODE)**: this option allows to specify the time at which the value of the control signal is uploaded and whether a shadow register should be adopted or not during this task. Some considerations are reported in the following chapters on the introduced delay when the drop-down menu is set on **Counter equals to zero**.

The same setting is available for **CMPB** as well.

Chapter 15

Page 245, equation (15.23) has an error in the intermediate step. The correct formula should be

$$v_{\mathrm{a,RMS}} = \sqrt{\frac{1}{T_s}\left(\int_0^{t_1} V_{\mathrm{DC}}^2 + \int_{t_1}^{T_s} (-V_{\mathrm{DC}})^2\right)\mathrm{d}t} = \sqrt{\frac{V_{\mathrm{DC}}^2 t_1}{T_s} + \frac{V_{\mathrm{DC}}^2\,(T_s - t_1)}{T_s}} = V_{\mathrm{DC}}$$

Chapter 17

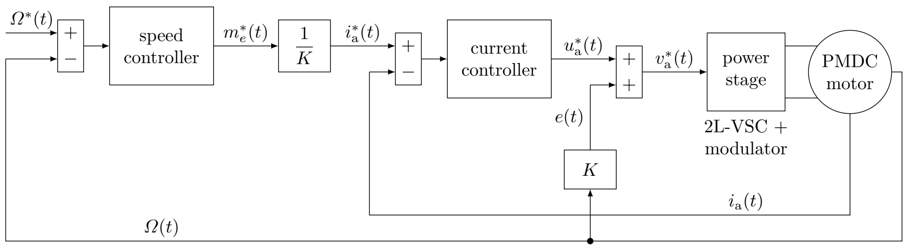
Page 268, Figure 17.1: there is a typo in the caption. The word bard *should* be **board**.

Chapter 18

Page 294, Figure 18.1: there is a typo in the caption. The word bard *should* be **board**.

Chapter 19

- **Page 330, figure 19.3**: the sign of the back emf compensation is wrong. It should appear with a + sign. A revised version of this scheme is reported here in the following:



- **Page 362, figure 19.22**: for the sake of clarity, it must be specified that the two back-emf generators in the equivalent circuits of M and B must have an upwards polarity to avoid the persistent polarization of the free-wheeling diode. Additionally, it must be noted that the sign of the torque in **figure 19.22 (b)** is coherent with the sign of $i_b(t)$ in the same picture.

- **Page 363, equation (19.26)** reports a sign of $m_{e,b}(t)$ which is not consistent with **equation (19.25)**, but it refers to **figure 19.22 (b)**. **Equation (19.26)** should be rewritten as follows:

$$J_{eq}\frac{d\Omega(t)}{dt} = m_{e,a}(t) + m_{e,b}(t) - \beta_{eq}\Omega(t)$$

  in order to be consistent with **equations (19.24) and (19.25)**

- **Page 368, figure 19.24**: the main body of this section does not underline the presence of the back-emf compensation term for B and its importance. The reader is invited to run this scheme with and without the compensation term and observe the resulting behaviour of the system.

Chapter 20

- **Page 380**: it is important to note that the execution profile of the firmware can be achieved for the whole script as well in the Simulink® environment. To this aim, the user should go in the **Modeling** bar, click on the **Model Settings** icon, look for the **Code Generation** – **Verification** pane and thick **Measure task execution time**. Then, select **Detailed (all function call sites)** in the **Measure function execution times** drop-down menu. After that, choose **All data** in the **Save options** drop-down menu. In this way, it is possible to check the execution time of the code running onboard the microcontroller. Indeed, once the firmware is executed on hardware (click on the **Build, Deploy & Start** icon), some commands can be entered in the MATLAB® command window. They are:

  ```
  codertarget.profile.getData('Simulink_file_name')
  executionProfile.report
  executionProfile.timeline
  ```

  The first line is required to create a variable in the MATLAB® workspace that contains all the information associated to the execution profiling. The second and third lines allows to visualize a table containing the execution time of the code and a timeline which visually describes how the code is executed inside each step time. More information on this topic can be found in [1]. Another new reference [2] is reported here below for the sake of completeness

- **Page 382**: it is important to note that, starting from newer releases of MATLAB® (later than 2020a), the available COM port are automatically recognized and loaded in the corresponding drop-down menu in the **Hardware Implementation**, **External Mode** pane. However, it is of paramount importance to check that the correct COM port is selected properly and, if necessary, refresh this field.

- **Page 382**: newer releases of Simulink® have a slightly different icon for starting external mode execution with respect to the one reported in the book. It is Monitor & Tune ▾ .

- **Page 382**: it must be noted that newer releases of MATLAB® significantly improved the stability of external mode execution, allowing runs that may last many hours.

Appendix B

Page 422: contacts of ePEBB^s are updated and information regarding this start-up is added.

Newer versions of these hardware kits would be distributed by ePEBB^s Srl. In case of interest, please contact:

alessandro.grittini@epebbs.com
mattia.rossi@epebbs.com
nicola.toscani@epebbs.com

ePEBB^s Srl is a start-up founded by researchers of Politecnico di Milano which develops high-end hardware and prototyping equipment for power electronic systems, drives, and smart grid. The name ePEBB^s stands for Efficient Power Electronics Building Blocks, representing the aim of the company products which is to accelerate the implementation of laboratory-scale power converters and facilitate the derivation of high-quality experimental results (i.e., signal integrity) and IoT connectivity (e.g., predictive maintenance). For further information, please visit

`https://www.linkedin.com/company/epebbs/`

Bibliography

[1] MathWorks. Embedded Coder® Support Package for Texas Instruments™ C2000™ Processors – Users' Guide. Technical Report texasinstrumentsc2000_ug, MathWorks, 2021. Available at `https://it.mathworks.com/help/pdf_doc/supportpkg/texasinstrumentsc2000/texasinstrumentsc2000_ug.pdf`.

[2] MathWorks. Embedded Coder® Support Package for Texas Instruments™ C2000™ Processors – Reference. Technical Report texasinstrumentsc2000_ref, MathWorks, 2021. Available at `https://it.mathworks.com/help/pdf_doc/supportpkg/texasinstrumentsc2000/texasinstrumentsc2000_ref.pdf`.