ISSUE: April 2023

Real-Time MCU Programming Is Applied To PMS Motor Control

Introduction to Microcontroller Programming for Power Electronics Control Applications, Mattia Rossi, Nicola Toscani, Marco Mauri and Francesco Castelli Dezza, CRC Press, 2022, 429 pages, hardback, ISBN: 978-0-367-70985-3.

Reviewed by Dennis Feucht, Innovatia Laboratories, Cayo, Belize

The authors of this book are with the Politecnico di Milano, Italy, in the Laboratory of Electrical Drives and Power Electronics. The book is a collaborative effort between the authors of the laboratory and companies: Würth Electronics, Texas Instruments, and Matlab, produced by Mathworks. Consequently, the book offers a concrete orientation around the TI TMS320C2000 µC line and circuit and electric machine simulation in Matlab.

The introductory chapter opens with the place of the µC (or what the book calls the MCU) in a real-time closed-loop control system where the MCU is within the loop. The emphasis throughout the book is on automotive motor control though the principles apply to motor drives more generally and also to electronic power conversion.

Part I begins with chapter 2, "Automatic Code Generation with MATLAB" in revealing the design style of programming the MCU through Matlab and Simulink which generates C/C++ MCU code. The design focuses on modeling everything, including the MCU, as "Model-Based Design". The motor is also emulated in an FPGA with which the MCU interacts to result in real-time "simulation"—actually, motor emulation—and testing.

This approach to design obviously depends on being able to accurately model what is being emulated. For benchtop loads, this seems unnecessary, but if the load is larger (such as a vehicle drive motor of >10 kW), emulating it might be more conducive to control development. A hint emerges of the need to go beyond continuous control to digital control in design—part II of the book.

In the software iteration steps, Simulink outputs C code which is assembled to object code, and via a linker the hex file is downloaded to the MCU. This multi-step procedure seems cumbersome to a real-time Forth programmer, though it is dominant in embedded software development.

In an alternative *umbilical* scheme, a few commands are downloaded to a target MCU to access its address space while the executable control code remains running on a desktop computer in a Forth interpreter-compiler language environment. Forth programs can be compiled incrementally onto a linked list of "named subroutines" (or "words") and when executed on the development computer running Forth causes the I/O to occur in the target system via the umbilical link. No complete recompilation, assembly, loading, and linking are required. The amount of equally well-documented source code in Forth is typically a tenth the number of lines in C.

The authors recognize some of the "drawbacks" in the Simulink-C development scheme caused by language incompatibilities between Simulink and C. For instance, MATLAB supports *polymorphism*—a software feature that allows the same name to be used in different contexts—while C has no provision for it. (In Forth, it is a consequence of programming in a way that allows it.)

For instance, oscilloscope software might have two channels, each with a VOLTS/DIV setting. If the same name, VOLTS/DIV, is used in two different linked lists of words, one for each channel, then the chosen list determines for which channel VOLTS/DIV applies.

TI provides a library of functional blocks for motor drives that reduce the amount of effort required to achieve a working development system, though as a first-rate engineer, you might resort to DIY to try to improve on any of them. Chapter 3 goes through details of the TI development board and chapter 4, the TI development software. Because TI has done so much of the groundwork for motor-drive development, their hardware-software tools provide an attractive jump-start in developing a motor drive for a particular application. There is no need to "reinvent the wheel" if you like the one TI offers.

Part II returns to a more theoretical topic in reviewing the fundaments of closed-loop (feedback) control. The general state-variable formulation of system dynamics—in the time domain with a system of two differential equations, one for the state of the system $x(t)$ and the other for the output $y(t)$—begins the math, with a given solution, commented to show the natural and forced response terms, for both x and y.

For those more familiar with Laplace-transformed *s*-domain formulations, the authors do a nice job of relating the two. While the matrix formulation in state variables is often applied to multiple I/O systems, for motors it can usually be reduced to single-variable I/O and an *s*-domain transfer function describes linear system behavior. The authors demonstrate with a proportional-integral or PI controller example. PI refers to the kind of dynamic compensation in the loop—namely, a quasistatic (0+ Hz) gain added to a single pole at the origin.

The authors have, in my view, a clear and simple understanding of control theory that is well-presented. My only preference (other than symbol refinement) is that the transfer functions be put in normalized instead of canonical form so that they are more easily interpretable. However, the authors are "spot on" in deriving equations from first principles that are then incorporated into a design procedure. The alternative is experimental pole-zero "tuning".

Although final refinement might be made with some tuning, a solid theoretical design foundation moves a project more quickly to completion while knowing performance margins that only extensive testing of prototypes would otherwise obtain. (This might also be necessary in the product development process, but theoretical prediction adds another dimension of insight and certainty to the final product design.)

Discrete-time control is then presented in chapter 5, with the three most common integration methods: backward and forward rectangular (Euler) and trapezoidal (Tustin). Chapter 6 presents a PI controller example. Chapter 7 covers numeric representation, including fixed-point (rational number) and floating-point formats.

In MCU control of physical systems, the additional computing required for floating-point arithmetic is usually not needed because voltages and currents are not input or output through DACs or ADCs with typically any more than 12 to 20 bits, where $2^{20} \approx 10^{20/3.32} \approx 10^6$ or about one million, well within the range of 32-bit fixed-point arithmetic; more commonly, $2^{12} = 4096$—within 16-bit arithmetic. However, divide 16-bit numbers and they require double-precision or a 32-bit range.

An interesting aspect of rational arithmetic is that of scaling fixed-point numbers by using multiple unit-point formats. The authors conclude that the best representations are in signed or unsigned fixed-point formats "wherever it is possible". I agree.

Part III, chapters 8-14, of the book plunges back into the details of peripheral I/O interfacing and the accompanying software driver programming. This includes serial, DAC/ADC, PWM, and digital bit (GPIO) interfaces. Various methods for generating PWM waveforms are presented; on-time can be phased as leading-edge, trailing-edge, centered in the switching period, or given an arbitrary phase offset. Another consideration is the synchronization of DAC outputs and ADC inputs. Chapter 14 covers rotary position encoders and speed computation.

Part IV, chapters 15-19, begins with open-loop control of a "Permanent Magnet DC Motor"—a small, low-cost commodity PM brush motor and a linear model of it. Throughout the book, photographs of test setup variations are shown, with MCU board, a motor-drive power electronics board, and an adjacent controlled brush motor coupled by shaft to a load motor or shaft encoder, all mounted on the development board with rubber mounting feet at the corners.

Half- and full-bridge power drivers appear in the modeling and involve PWM, always presented concretely in the context of Simulink and the TI C2000 MPU. The circuit of low-side current sensing is given a chapter (16) as is (chapter 17) the current control of an RL load. The general progression is from power conversion loads, where nothing moves (corresponding to a zero-speed or stalled-rotor motor), to a range of electric-machine loads. Load parameter variation tests control robustness.

Chapter 18 ventures farther, to voltage control of an RLC load in the form of a buck power-transfer circuit. Some attention is paid to how supply voltage ripple can affect behavior. As for PWM ripple, the authors choose state-space average voltage-mode control modeling to derive the incremental system behavior around an operating-point.

When linearized, a transfer function can be constructed and a control compensator designed. This is possible when the sampling rate is made sufficiently higher than the loop unity-gain crossover frequency ω_c (or ω_T for analog circuits and transistors). What complicate motor loops are mechanical load variations and departures from a given operating-point, defined by the speed and torque of the load. Mechanical loads are characteristically nonlinear and the full range of control behavior is tested or derived over the full range of anticipated mechanical dynamics.

The pace of excitement heightens with chapter 19, section 19.5.1, when what is popularly called *sensorless control* of motors appears as model reference adaptive control. Advanced control methods have abounded since at least the 1960s. Not only is there continuous and digital classical feedback control, beyond these are predictive, adaptive, optimal, nonlinear, and neural control.

Feedback control with MCUs in the loop is digital control. In this case, the induced voltage in the motor winding, which varies proportionally with motor speed, is sensed and the motor model used to estimate its speed. The voltage at the winding terminals is not the induced voltage as such but has added to it the voltage drop across the winding resistance. This drop is the motor resistance from the motor model times terminal current, which is controlled as the output.

Instead of feeding back the quantity to be controlled, observable variables are fed back to the motor model so that it can apply model parameters to these quantities and more accurately estimate the output that will produce the commanded speed. An equation for output current is based on model parameters, adjusted or adapted by the measured voltage.

The book then goes on to describe how the motor model electrical parameters, winding resistance and inductance, mechanical parameter of rotor inertia, and what relates the electrical and mechanical sides of the model, what I call the (average) electromechanical flux constant, λ_{me}, are measured. All of page 354 is filled with the block diagram of the system simulation.

A simple *dynamometer*, an active mechanical load, is constructed by connecting two of the motors through their shafts. The load motor is voltage-driven for speed control and the braking motor—essentially the dyno—is current-driven to control the torque applied to the control-loop load motor. Then by varying the brake current, speed is tested over the torque range of the load motor. (A deluxe Helical shaft coupler is used, the kind with a spiral cut.)

Finally, part V, chapters 20-21, like part IV, is about real-time control, but addresses load emulation. Two evaluation schemes are given, named processor-in-loop (PIL) testing and external-mode execution. Both connect Simulink in different ways to the MCU loop, to compare the running system with the Simulink model.

The last chapter, 21, turns to automotive applications as "Electric Propulsion Case Studies". It gives two cases, a tramway vehicle in Milan, Italy and an electric racing car with three-phase, Y-configured permanent-magnet synchronous (PMS) motors. Two appendices follow: A on the basics of the C language, and B on the electronic MCU and motor-drive boards and the motor kits used in the book. A bibliography is followed by a token index.

What recommendation rating does this book receive? Overall, it is a worthwhile exposition of all major aspects of µC-based motor controller design. Instead of remaining purely theoretical, it combines well both theory and experiment, both desk and bench, by choosing a particular simulator (Simulink) and µC (TI TMS320C2000) to present a concrete development of actual electronics and software. (Some Simulink and C code are also given.)

English is not the native language in Italy, yet the authors are fluent in it, with only occasional hints of "Italish". The theory is not complete but the essential equations and clear explanation of them are given. I would recommend this book to anyone who wants to put together a first motor control system with a preference for the TI µCs and a desire to use MATLAB Simulink. The book is broader in coverage than these two particular products but for anyone interested in them in particular, the book hits the bull's eye for applicability.

About The Author

Dennis Feucht has been involved in power electronics for 40 years, designing motor-drives and power converters. He has an instrument background from Tektronix, where he designed test and measurement equipment and did research in Tek Labs. He has lately been working on projects in theoretical magnetics and power converter research.

To read Dennis' reviews of other texts on power supply design, magnetics design and related topics, see How2Power's Power Electronics Book Reviews.