

Application of Auto-Coding for Rapid and Efficient Motor Control Development

James Walters, Cahya Harianto, Edward Kelly, and Tanto Sugiarto
Delphi Automotive

ABSTRACT

In hybrid and electric vehicles, the control of the electric motor is a critical component of vehicle functions such as motoring, generating, engine-starting and braking. The efficient and accurate control of motor torque is performed by the motor controller. It is a complex system incorporating sensor sampling, data processing, controls, diagnostics, and 3-phase Pulse Width Modulation (PWM) generation which are executed in sub-100 uSec periods. Due to the fast execution rates, care must be taken in the software coding phase to ensure the algorithms will not exceed the target processor's throughput capability.

Production motor control development often still follows the path of customer requirements, component requirements, simulation, hand-code, and verification test due to the concern for processor throughput. In the case of vehicle system controls, typically executed no faster than 5-10 mSec periods, auto-coding tools are used for algorithm development as well as testing. The advantages of auto-coding to greatly speed the development process by linking the tools for simulation, code generation and testing early in the development process as well as to more easily investigate performance issues late in the process are well known. It is not uncommon, however, to lose coding efficiency with this approach. While the loss of efficiency may be tolerable for slow periods, it is not acceptable at faster periods used in motor controls as it will preclude the algorithms from executing or drive unnecessarily expensive solutions.

This paper will present an auto-coding process applied to motor controls, including full implementation on a production permanent magnet motor drive. Best practices for implementing requirements into models that generate efficient code will be highlighted. An overview of the issues associated with model-based documentation will also be covered. The use of test vectors at the component, model and hardware-in-the-loop (HIL) level will be presented to show the benefits derived from using a formalized process and the natural linkage to a SPICE® compliant process. A timing study performed during dynamometer testing detailing the differences between the original hand-code and the model-based code will be presented.

CITATION: Walters, J., Harianto, C., Kelly, E., and Sugiarto, T., "Application of Auto-Coding for Rapid and Efficient Motor Control Development," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 7(2):2014, doi:10.4271/2014-01-0305.

INTRODUCTION

A general product development cycle contains several basic steps including: the formation of customer requirements, analysis of requirements to form the high level system design, development of software and hardware requirements, software and hardware design and ultimately verification testing at both the component and system level [1]. Automotive manufacturers often require their suppliers to follow a software development process such as the Automotive Software Process Improvement and Capability Determination (SPICE®) process [2]. The Automotive SPICE® process, shown in [Figure 1](#), summarizes these steps.

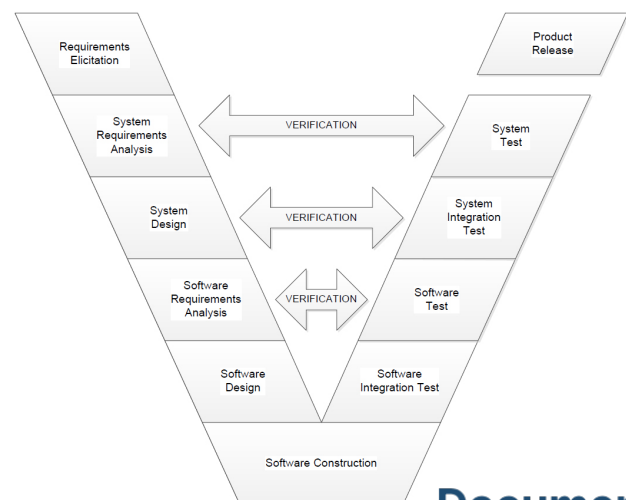


Figure 1. Automotive SPICE® Process

In these steps, there are numerous simulations performed to verify the system goals will be met by the design. From the capturing of requirements to the creation of software, it is desired to directly use the simulations that have been performed so as not to repeat the effort and potentially introduce errors. In addition, it is desired that the simulation model accurately reflects the implementation as it will greatly aid in any subsequent problem resolution.

The Automotive SPICE® Process recognizes that engineering development will need to incorporate the contributions from a variety of engineers including systems, function developers, software, and test engineers. What is needed is a tool and methodology that can directly support the translation of the system and component designs, often simulated early in the process, efficiently and accurately into an implementation that can be verified and released quickly. Auto-coding is a tool that facilitates development by translating simulations, performed in the System Design step, into the final software implementation [3]. It has also been used in the last decade to implement many functions in the vehicle [4] - [6].

Although auto-coding offers considerable advantages for development, many of the implemented designs have been for relatively slow loop control tasks (5-10 mSec or greater period). Ironically, complicated time-critical components which are mostly likely to be simulated are not as commonly auto-coded, thus losing a potential resource saving opportunity. This is due to the concerns that the auto-coding process may have additional overhead in algorithm execution causing throughput issues in the application.

It should be noted that there are Rapid Controls Prototyping (RCP) tools that allow for quick transitions from a model-based / simulation environment to concept testing [7]. RCP does eliminate the need for the manual hand-coding step when transitioning from the simulation model to the implementation, which requires additional effort and is prone to errors; however, this process involves extra hardware. The RCP approach, which can be extremely useful for concept testing, is inherently different though since the goal is concept development and evaluation rather than efficient code generation in a production environment.

This paper focuses on the development of an Automotive SPICE® compliant process using auto-coding techniques for the torque control of a permanent magnet synchronous machine (PMSM). By appropriately defining a process and associated design environment, it will be shown how to directly go from simulation to implementation for a complicated system involving many contributors and to naturally perform both system and software tests in an automated fashion. The process to actually implement a design in this fashion is non-trivial and necessitates a requirements focused approach with the implementation being

evaluated not only for accuracy to requirements, but also for efficient code generation. The benefits of this approach are substantial as the simulation tools can now be used to verify compliance to system requirements and thus allow porting concepts from simulation directly onto hardware. There are also other inherent benefits as the already tested concepts are now easily transferable between different projects and production hardware can now be utilized as a development platform.

Motor Control Background

In a hybrid vehicle, the motor controller is responsible for the efficient and accurate production of torque. This torque may be used to perform a variety of tasks including motoring, generating, speed regulation, and/or dc voltage regulation, depending upon the application. An overview of an electric machine drive system is shown in Figure 2. The input to the system is a command that is typically obtained from a Hybrid Control Module (HCM) via communication lines and the outputs are 3-phase voltages applied to the machine. Typical components of the motor drive system include: the battery, motor controller, sensors for current, temperature and position, inverter and motor.

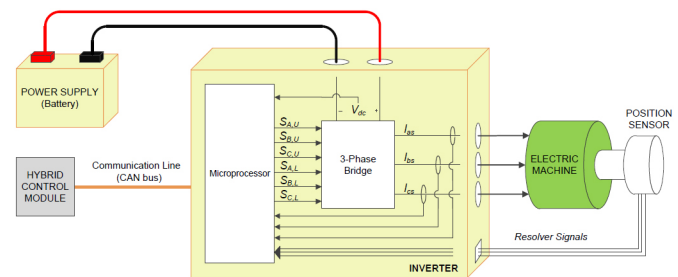


Figure 2. PMSM Drive System

As shown in Figure 3, the main components in the inverter are the 3-phase bridge which receives 6 gate signals from the motor controller and the DC bulk capacitor. The three legs of the inverter are connected to the electric machine. For many applications in hybrid and electric vehicles, the PMSM has emerged as the preferred motor technology, though induction motors are commonly used as well [8, 9].

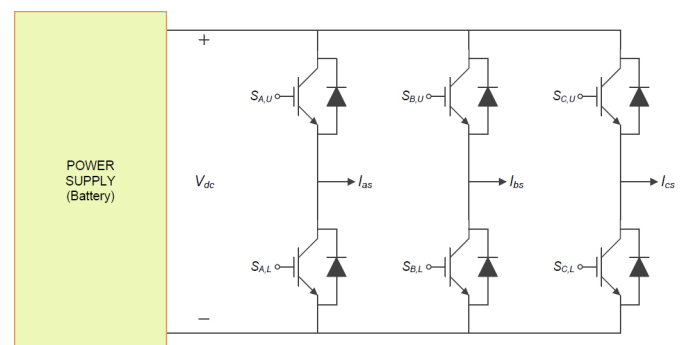


Figure 3. 3-Phase Bridge

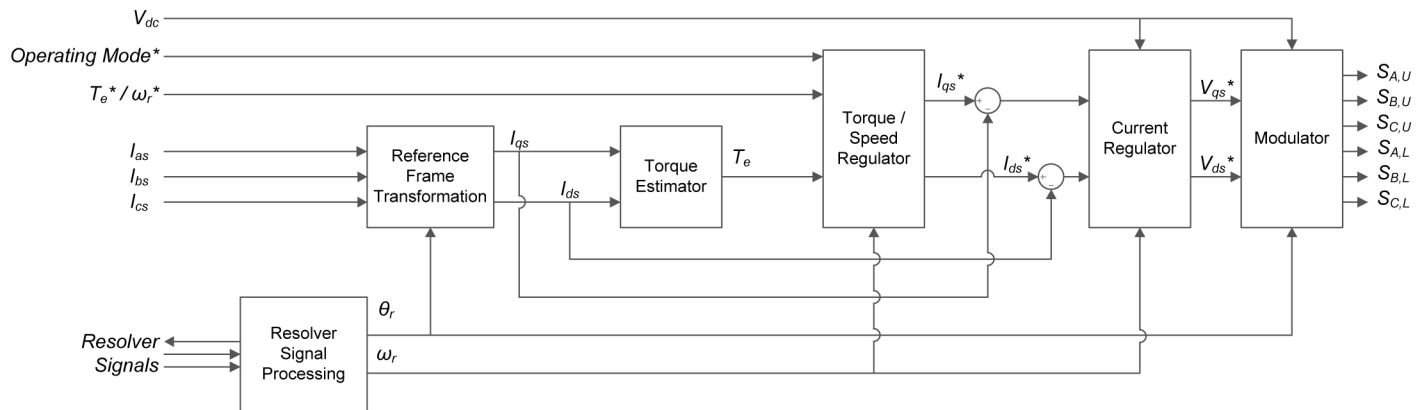


Figure 4. Torque Control Overview

From a vehicle driveline perspective, torque response is needed within a 30-50 mSec range in order to satisfy performance requirements for cranking, torque assist, generating, and braking. In order to create torque that is impervious to voltage or speed disturbances, it is necessary to control currents in the machine at a much faster rate, however. Vector control techniques are commonly employed to perform this task [10]. For a variety of reasons including motor time constants, the maximum fundamental frequency in the machine (1000-1400 Hz) and limiting DC capacitor voltage ripple, it is necessary to run current control loop rates of approximately 100 uSec. The sampling of the inputs, current and position, calculation of current commands, control of current, and the associated transformation pairs for vector control as well as the center-based PWM generation must all be performed at this rate of 100 uSec.

The regulation process can be described in Figure 4. Therein, the commands to / from the controller are denoted with the starred quantities (e.g., T_e^* / ω_r^* denote the commanded torque / speed).

In addition to the motor control functions, the protection of the inverter has to be considered. The diagnostic protection needs to respond quickly to critical fault conditions (e.g., shorted devices, loss of current control, etc.) and will require functionality in the 100 uSec loop. Finally, sufficient throughput must be available to ensure that slower loops (1, 5 and 10 mSec) have adequate time to execute.

As can be observed, the torque control of an electric machine is inherently a challenging process. Not only does the control process require fast data sampling, the execution of the current control to generate the torque on the machine requires fast and precise timing. Although auto-coding the PMSM control algorithm can provide great benefits (e.g., faster and more efficient development process and quick transitions from simulation to final product implementation), care must be taken in this approach so that the resulting code is efficient and does not consume excessive throughput on the microprocessor.

AUTO-CODING TOOL SELECTION

Conceptually, a simulation model represents the programming language and an auto-coder is effectively the compiler. The goal is to select a toolset that enables the transfer of the 'coding' responsibility from the software engineer to the actual function developers. This step is fundamentally important as the function developers possess the domain knowledge that allows for a direct implementation to the requirements. Furthermore, it is also important that the toolset satisfies the initial goal of performing the pertinent simulations used to design the system. It is also necessary for the toolset to quickly and accurately generate code that executes efficiently. Finally, it is desired, as part of the SPICE® process that the toolset allows for seamless verification testing of the implementation. Testing should be performed at each stage, from software integration test continuing through the complete system test. It is desired that test vectors, known inputs with expected outputs, are used to verify each major block as well as the overall motor control strategy shown in Figure 4.

While there are a variety of options, the MATLAB® toolset was chosen for this work. First, the system and motor control teams are already familiar with this tool from their design simulation work. Next, the toolset can be used from one end of the development process to the other as the model can be conceived, simulated, auto-coded and verified all within this environment.

AUTO-CODING PROCESS FOR PMSM CONTROL

Per the SPICE® process, product development starts from the point of the high level requirements being captured continuing to system requirements analysis which should contain a high level simulation phase intended to define the system. In SPICE®, there are also many steps involved with the selection of tools and methods that should occur, which fall outside of the intent of this paper. Focusing instead on the development of requirements, reviews are held to ensure that the proposed system design will meet the overall requirements and to establish traceability.