

Towards an Open Source Framework for Small Engine Controls Development

Paolo Gai, Francesco Esposito, and Riccardo Schiavi
Evidence Srl

Marco Di Natale
Scuola Superiore S. Anna

Claudio Diglio, Michele Pagano, Carlo Camicia, and Luca Carmignani
Piaggio & C SpA

ABSTRACT

The paper describes the components of an envisioned open source framework that supports several stages in the model-based development of two- and three-wheelers software controls. The proposed solution supports the runtime execution on an OSEK-compatible [8] real-time operating system for multicore platforms. The framework consists of a modeling and simulation tool (including hierarchical state machines) and a code generator for the development of the functional model of controls and the definition of their task implementation; an OSEK/AUTOSAR operating system and device driver stack; OS and I/O configuration tools. The platform has been released open-source under an industry-friendly license. Our framework is currently in use for the development of innovative two-three wheelers control systems at Piaggio. In this paper we describe the experience matured in the application development, the benefits and current limitations of the approach. In particular, the result of this study has been a demonstrator platform which includes a BLDC motor control written using the proposed framework.

CITATION: Gai, P., Esposito, F., Schiavi, R., Di Natale, M. et al., "Towards an Open Source Framework for Small Engine Controls Development," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 8(1):2015, doi:10.4271/2014-32-0070.

INTRODUCTION

The design and development of the electronic and control software architecture for the next generation of two wheelers is one of the major challenges in the development of small engine products.

The need for optimal fuel consumption and emission control, paired with strict safety requirements is pushing manufacturers to reconsider their electronic architecture, adding complex functionality with the risk of incurring significant costs for the integration of new systems and the possible redefinition of the electronics architecture.

These challenges have already been tackled twenty years ago by the automotive manufacturers, generating a set of standards in an attempt to improve productivity and quality by standardization, interoperability and competition on functional content. The result has been the development of distributed software architectures based on the CAN bus and the OSEK/VDX [8] standard, and more recently the AUTOSAR [4] infrastructure. A set of automotive standards are today

available and can be used as an inspiration as they share common goals such as portability, reusability, and limited footprint. Unfortunately, the inherent complexity and license cost of complete commercial bundles makes the adoption of these automotive standards unlikely for low-cost engines.

Moreover, the new challenge in the field is the application of functional safety concepts (becoming a requirement for automotive systems, after the ISO26262 norm publication [11]), which are soon expected to become the standard also for motor-motive systems. Functional safety includes several measures and best practices to assist the maker in providing the highest achievable safety level. These measures have an impact on the design of new products, on their bench and road tests, as well as on their assembly, service and product lifecycles.

All these changes will have a non-negligible short-term impact on costs. Knowing that the concepts of functional safety apply to all the stages of the software and hardware development,

Piaggio & C. planned for a significant investment in the development of a platform to guarantee adequate safety levels for its vehicles.

The first step has been the development and assessment of an electronic platform based on new hardware and software components developed in accordance with functional safety requirements.

This paper presents the results of this initial development effort, aimed at building low-cost two and three-wheelers software applications. The proposed solution is composed of a set of open-source target software modules, supported by a set of design tools for the simulation and the efficient code generation of control laws and state machines. The idea is to enable manufacturers to add their own functional and (system-level) controls know-how easily, adapting the platform to various kinds of applications.

Being open-source is a major advance in the development of software for small engines: it guarantees the possibility to develop applications while retaining control of the intellectual property and avoiding the disclosure of proprietary code (thanks to the GPL2+Linking Exception license [3]), but still enables manufacturers to share code not contributing to product differentiation. Moreover, it allows to build products without recurring expenses.

The resulting software platform has been ported to the Freescale MPC5643L Leopard MCU, and is composed by a Bootloader, the ERIKA Enterprise RTOS, Peripheral Drivers and High-level communication protocols aimed at diagnostic and calibration support;

Special attention has been dedicated to the testing of the RTOS and of other part of the system, including MISRAC: 2004 compliance [5], and black box regression tests including the MODISTARC and OSEK/VDX certification tests.

Finally, the platform supports the integration with model-based development tools like Scicos and E4Coder [6]: a set of tools that can be used to model and simulate hybrid control algorithms, and generate code for embedded microcontrollers, in a similar fashion as the widely used Matlab and Simulink [7].

We believe that such an open platform has the potential to create a common low-cost execution platform that will help manufacturers in creating innovative control applications.

The following sections describe the choice of the open-source licensing model, and will provide a high-level view of the proposed platform. Afterwards, the microcontroller, bootloader, RTOS, Drivers and Model based Design tools are described in more details. Finally the current Piaggio demonstrator, composed by a BLDC motor control, is described in detail, including experiments and benchmarks.

Open-source Licences and Automotive software

The choice of using open-source software for the components of the platform has been driven by an analysis of automotive market trends and the evolution of software components development.

Automotive software architectures are becoming increasingly complex. The control architecture is transitioning from the federated networked infrastructure of the 90s to the integrated architecture concept, enabled by the AUTOSAR [4] standard. Market trends for infotainment systems are pushing for subsystem integration on multicore ECUs, with a widespread adoption of embedded Linux platforms.

In the AUTOSAR view, complexity is managed by cooperation at the level of the basic software and differentiation (or competition) on the application-level functional content. Today, ECU requirements are common across several car manufacturers, with a small percentage of notable exceptions that provide for product differentiation [9].

We believe that the small engine market will experience the same kind of evolution. In this view, open source software and an open platform represents the most radical form of collaboration in the development and exchange of basic software solutions. For this to happen, a crucial choice was the selection of the most appropriate licensing terms. In particular, the licensing term must allow the static linking of proprietary code in order to preserve the confidentiality of the application part (providing the real value in the differentiation of the product), while retaining the capability of cooperation and sharing on the open-source part. For this reason, the selected license is the GPLv2 with Linking Exception [3] for the code to be embedded in the target ECUs, and the EPL license [10] for the Eclipse plugins of the development environment.

PLATFORM DESCRIPTION

The project aims at the creation of a framework that support all the development stages for the design and implementation of controls for modern hybrid combustion small engines. The developed systems should include the powertrain controller for the endothermic motor as well as for the electrical motor hosted in two and three-wheelers.

Platform requirements include the capability of providing automotive-grade performance, without jeopardizing future requirements on safety, and access to the platform at an affordable cost for the tight 2-3 wheelers budgets.

The cornerstone of the run-time execution platform architecture is the open-source ERIKA Enterprise OSEK/VDX RTOS. The development framework is based on the open source Eclipse Modeling Framework, on which the RT-Druid open plugin provides for component customization and configuration. Complementing the runtime components and the configuration tools, we provided for an open source flow for model-based

development, based on an extension of the Scicos modeling tools [26] with a state machine modeler and efficient code generators [27].

The overall view of the runtime architecture is represented in Figure 1.

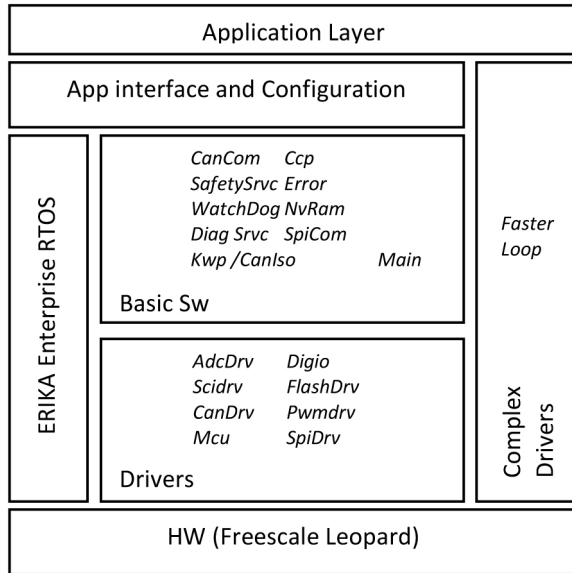


Figure 1. Components of the runtime architecture.

The main components are:

- The Application layer, which contains the functional part of the application;
- The Application Interface/Configuration Layer, which contains all the configurations and initial setup of the underlying Basic Software and Drivers;
- The Basic Software Layer, which provides application services to the main application, including the communication services (CAN, CCP [31], KWP2000 [32], ISO-TP [33]);
- The Drivers Layer, which abstracts the underlying hardware peripherals (ADC, PWM, ...);
- The ERIKA Enterprise OSEK/VDX RTOS, offering operating system services;
- The Complex Drivers Layer, which contains a set of software components that for their complexity and criticalness cannot be described by a standard layer (in particular, the ADC/PWM coupling implemented with the CTU of the Leopard MCU);
- The Freescale Leopard MPC5643L MCU.

In addition to these layers, the Platform offers:

- A Bootloader, enabling CAN-based reprogramming, reconfiguration, boot, and system diagnostics.

The following sections will analyze in detail the Leopard Microcontroller, the Bootloader, the RTOS, and provide information on the Drivers and the Basic Software. The Complex drivers and Application Layer will be discussed in the context of the test application Section.

THE LEOPARD MICROCONTROLLER

The platform target for the hybrid engine project is the Freescale MPC5643L Leopard MCU. The microcontroller was selected because it may support safety-critical applications up to ASIL D level, offering an easier path toward future safety qualification in the motor-motive market.

The Leopard MCU offers two PowerPC CPUs, which can be configured to run in Lockstep (as safety measure), or in multicore mode (to boost performance), and offers fast ADC with PWM integration which are ideal for implementing electric machine control strategies.

The Flash memory of the MCU is 1Mb, organized on three address spaces:

1. Low address space (256 KB);
2. Mid address space (256 KB);
3. High address space (512 KB).

The MPC5643L flash memory is managed as follow (see Figure 2):

- Block L0 (0x0 - 0x00004000) hosts the boot key of the microcontroller. The MPC5643L has several boot sectors, which can be configured to run in VLE or BOOK E mode.
- Block L1, L2 and L3 (0x00004000 - 0x00020000) are dedicated to Non-Volatile-RAM (NVRAM) data.
- Block L4 and L5 (0x00020000 - 0x00040000) contain the boot code. The content of this flash area is explained in the section dedicated to the description of the Bootloader.
- Block M0 and M1 (0x00040000 - 0x00080000) contain the calibrations data.
- Block H0 and H1 (0x00080000 - 0x000C0000) contain the application code.

FLASH_BASE address offset	Use	Block	Size (KB)	Partition
0x0	Low Address Space BOOT KEY	L0	16	1
0x0000_4000	NVRAM	L1	48	2
0x0001_0000		L2	48	
0x0001_C000		L3	16	
0x0002_0000	BOOT	L4	64	2
0x0003_0000		L5	64	
0x0004_0000	Mid Address Space CALIBRATIONS	M0	128	3
0x0006_0000	High Address Space APPLICAZIONE	M1	128	4
0x0008_0000		H0	256	
0x000C_0000		H1	256	
0x0010_0000 - 0x00EF_FFFF	Reserved			

Figure 2. Platform Flash Memory Map.