Issue 3/2026

The Engineering City Journal

Published by the Engineering Association *Yerevan, February 16, 2026*

The purpose of the Journal is to disseminate knowledge, experience, and best practice in engineering & manufacturing.
The articles are written by practicing professionals in the Engineering City (EC) in Yerevan. They cover issues relevant to their projects, operations, and organizations.
The articles address the challenges of the time. As such, the Journal is also a chronicle of the evolution of the EC, its companies, and its individuals.
It is the aim of the Journal to advance the engineering & manufacturing practice, to enrich the engineering & manufacturing community, and to contribute to the worldwide discourse in engineering & manufacturing.

Table of Contents

DEVELOPER'S INSIGHT 1
. Why Register-Level Drivers Still Matter in 2026 1
ART & ENGINEERING 2
. Poetry as Engineering Preparation - An Unexpected Foundation of Thinking 2
ARMENOLOGY 3
. Stories and Artifacts about the Venerable Legacy of 20th-Century Armenian Industrialists & Entrepreneurs 3

Developer's Insight

. Why Register-Level Drivers Still Matter in 2026

by Hripsime Harutyunyan, YEA Engineering

Introduction: The Illusion of "Done"

In 2026, embedded engineers have access to more abstraction layers than ever before. Hardware Abstraction Layers (HALs), middleware stacks, auto-generated code, and ready-made drivers promise faster development and portability across platforms. In many projects, they deliver exactly that.

Yet in real products — not demos or evaluation boards, but systems expected to operate reliably every day — engineers still find themselves opening datasheets, studying register maps, and manipulating individual bits by hand.

This article explains why register-level drivers still matter, where abstractions fall short, and why understanding what happens below the API boundary remains a fundamental engineering skill.

The Promise of High-Level Drivers

Modern vendor libraries and third-party frameworks solve real problems: faster system bring-up, reduced boilerplate code, easier onboarding of new engineers, and nominal portability across hardware families.

For many peripherals — GPIO, basic timers, simple UART communication — high-level drivers are often sufficient. They allow teams to focus on application logic rather than low-level details.

However, this promise rests on a critical assumption: that the abstraction accurately reflects hardware behavior under all relevant conditions.

Where Abstractions Begin to Leak

The need for register-level control rarely appears at the beginning of a project. It emerges later — when the system meets real-world constraints.

Typical triggers include timing-critical interfaces (SPI, CAN-FD, high-speed ADCs), shared buses, non-default operating modes, power-up and reset sequences, and performance limitations.

Debugging Reality: Registers Don't Lie

When something fails, the first real question is always the same: What is the hardware actually doing? High-level APIs rarely answer this question. Registers do.

Registers allow engineers to verify peripheral states, clocking, FIFO behavior, and interrupt conditions.

Determinism, Timing, and Control

Abstractions are designed for generality. Hardware is not. Register-level control provides deterministic timing and precise sequencing.

Long-Term Maintainability: The Quiet Advantage

Register-level code often ages better than high-level drivers because datasheets outlive software frameworks.

Register-Level Does Not Mean Unstructured

Good register-level drivers are clean, readable, and modular.

The Engineer's Mindset

Register-level work builds a deeper understanding of hardware and improves engineering quality.

Conclusion: Still Relevant, Still Necessary

Register-level drivers are not nostalgia. They are engineering responsibility.

Art & Engineering

. Poetry as Engineering Preparation - An Unexpected Foundation of Thinking

by Hripsime Harutyunyan, YEA Engineering

Many of us learned poetry in childhood. It is usually seen as part of cultural upbringing, something distant from professional life. Yet, with time, it becomes clear that this experience may have played a deeper role — especially for those who later became engineers.

This short article reflects on how memorizing poetry in childhood can shape cognitive skills that are directly relevant to engineering thinking.

Structure Before Formulas

Poetry is, above all, structure. Rhythm, sequence, internal logic, and pauses form a system where every element matters. One misplaced word breaks the entire construction. A child learns to sense when something is "wrong" even before formal logic appears.

Engineering relies on similar principles: ordered sequences, state transitions, timing, and cause–effect relationships. In this sense, poetry becomes an early exercise in structural thinking.

Memory as an Engineering Tool

Engineers constantly work with constraints: parameters, assumptions, interfaces, and operating modes. The ability to hold these elements in mind directly affects the quality of decisions.

Memorizing poetry trains active reconstruction of complex, ordered structures. Later, this skill appears as the ability to mentally hold an entire system rather than treating it as disconnected parts.

Armenian Poetic Tradition and Meaning

Armenian poetry is rich in imagery and internal logic. Works by Hovhannes Tumanyan, Sayat-Nova, and Paruyr Sevak cannot be memorized mechanically. Accurate reproduction requires understanding meaning and internal connections.

This forms the habit of seeking underlying logic rather than repeating surface form — a mindset that naturally transfers to engineering.

Intuition and a Sense of Correctness

Experienced engineers often sense that a solution is flawed before they can formally prove it. Poetry develops a similar sensitivity. A line either sounds right or it does not. This intuition later transfers to technical systems.

Abstraction and Symbolic Thinking

Poetry works through symbols and layered meaning. Engineering requires the same ability: schematics, code, and models are symbolic representations of reality.

Conclusion

The experience of memorizing poetry in childhood is not an incidental episode of the past. It is a form of early intellectual training that develops memory, structural thinking, intuition, and the ability to work with abstractions.

For an engineer, these qualities are no less important than knowledge of tools, technologies, or programming languages. They shape how problems are approached, how systems are understood, and how solutions are evaluated beyond formal correctness.

Perhaps this is why many strong engineers are drawn to music, literature, or other forms of art — not merely as hobbies, but as a natural continuation of the way they think. These domains exercise the same mental skills: structure, balance, intuition, and meaning.

Poetry does not automatically make someone an engineer. But it can help form the mindset of an engineer — long before the word engineering ever appears.

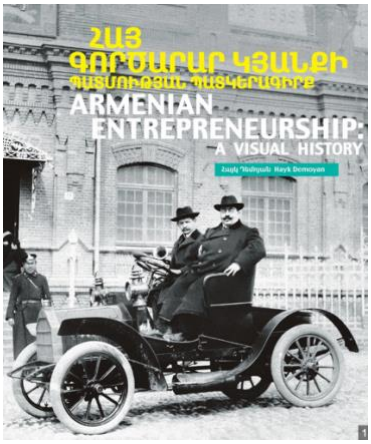
Armenology

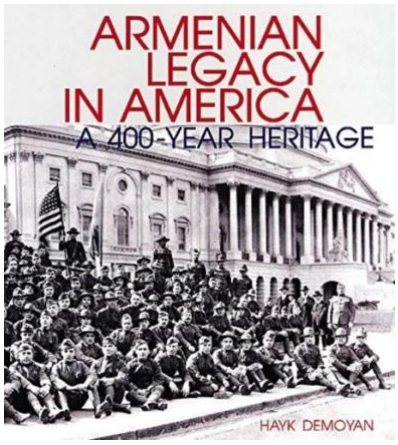
. Stories and Artifacts about the Venerable Legacy of 20th-Century Armenian Industrialists & Entrepreneurs

Պատմություններ և իրեր 20-րդ դարի հայ գործարարների ու արդյունաբերողների պատկառելի ժառանգության մասին

We had the historian Hayk Demoyan give a lecture on the legacy of 20th-century Armenian businessmen and industrialists.

The following books were also presented:











© 2025 The Authors of the articles

All materials contained in *The Engineering City Journal* may be used with reference to the Authors.

For feedback on the *Journal*, its content, and its articles please send to info@engineeringcity.am.

To submit articles, please send to info@engineeringcity.am. Articles can be in any language.
