Kinetis L Security Overview

Werner Almesberger
werner@almesberger.net

October 5, 2014

1 Introduction

This document gives an overview of the security architecture of the Freescale Kinetis L series with emphasis on the KL26 chip family. It summarizes information obtained from Freescale product documentation and the results of experiments conducted to resolve the occasional ambiguity. Furthermore, we discuss how the security mechanisms apply to typical usage situations.

This is part of the documentation of the Anelok project, where the microcontroller (MCU) acts as what is commonly called a "secure element" – a part of the system that contains secrets, that applies them for encryption, signing, and similar tasks, and that provides adequate means to keep these secrets from prying eyes.

The specifics of how Kinetis L series microcontrollers are used in the Anelok project are described in [1].

In Freescale's terminology, "security" refers to limiting external access to the inards of the microcontroller. The protection against accidental modification of Flash memory is closely intertwined with these security mechanisms and we therefore describe it as well.

There are some related topics that are beyond the scope of this document:

- A description of the mechanisms to access and configure the elements of the security architecture. Most of this can be found in product documentation provided by Freescale and by ARM, but some details – e.g., the intricacies of using the Serial Wire Debug (SWD) protocol – may merit further attention at a later point in time.

- Mechanisms the chip provides that support the implementation of a secure element and that are not purely defensive in nature. This would include random number generation or hardware encryption modules. While most of the cryptographic features available in the class of MCUs we are concerned with would be of little value to the currently planned uses of Anelok, proper random number generation is of critical importance and shall be discussed in a separate document.

- Resilience of the chip against physical tampering and the exploitation of side channels. A detailed analysis of these properties is beyond the project's current means but the issue should be revisited at a later time.

2 Kinetis L Flash security

The security architecture of Kinetis L series chips is mainly centered around their Flash memory. The following operations are governed by the security settings:

- debug access to the CPU core,
- access to internal memories, such as RAM and Flash, and
- the erasing of the entire[1] Flash memory, which includes a reset of the protection settings.[2]

The security settings themselves are stored in Flash.

2.1 Basic security scheme

Figure 1 illustrates the basic security architecture of Kinetis L series chips.

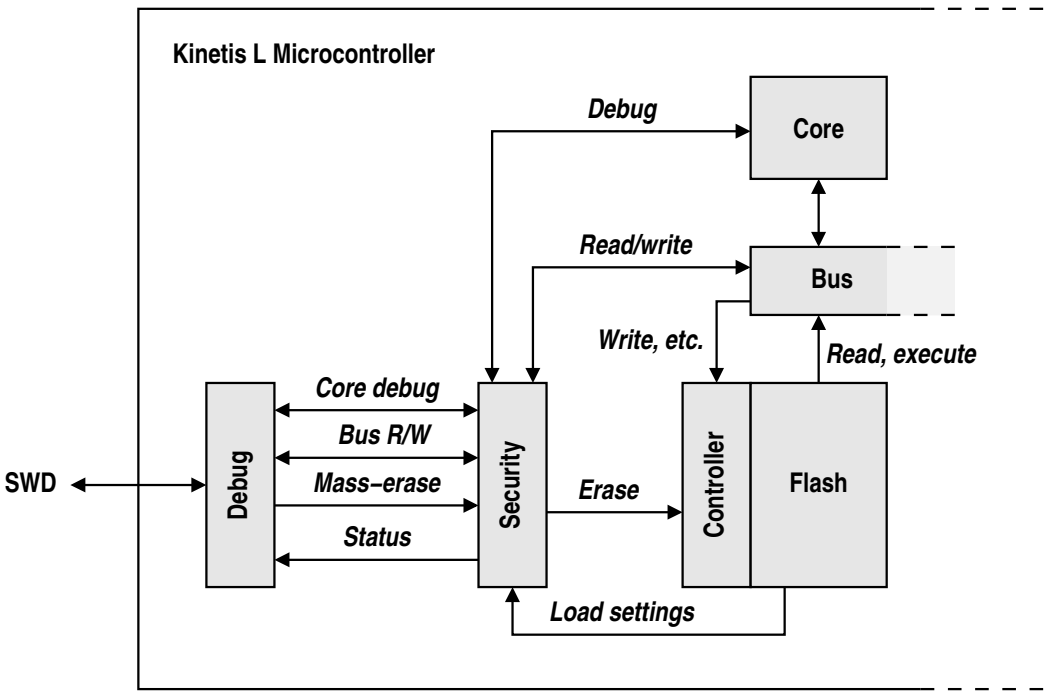

Figure 1: High-level view of the security mechanisms inside Kinetis L series microcontrollers.

[1] With the exception of the one-time programmable bytes. Anelok does not currently use this memory area. Further details can be found in sections 27.3.2, 27.4.10.7, and 27.4.10.8 of [2].

[2] A reset of the security settings may have to be followed by a chip reset in order to take effect.

On power-on reset, the security settings are loaded from Flash into the part of the debug subsystem that enforces these security settings.

When accessing the microcontroller via the Serial Wire Debug protocol (SWD), debugging, bus read and write (which includes accessing RAM, Flash, and registers – including those of the Flash controller), and mass erase operations are only allowed if the security settings agree.

Code that runs on the CPU core is not constrained by the security settings and can read data or execute code from the Flash, and – write protections permitting – write to it, or erase it as it pleases.

Figure 2 takes us one layer deeper and shows how the security configuration is implemented.

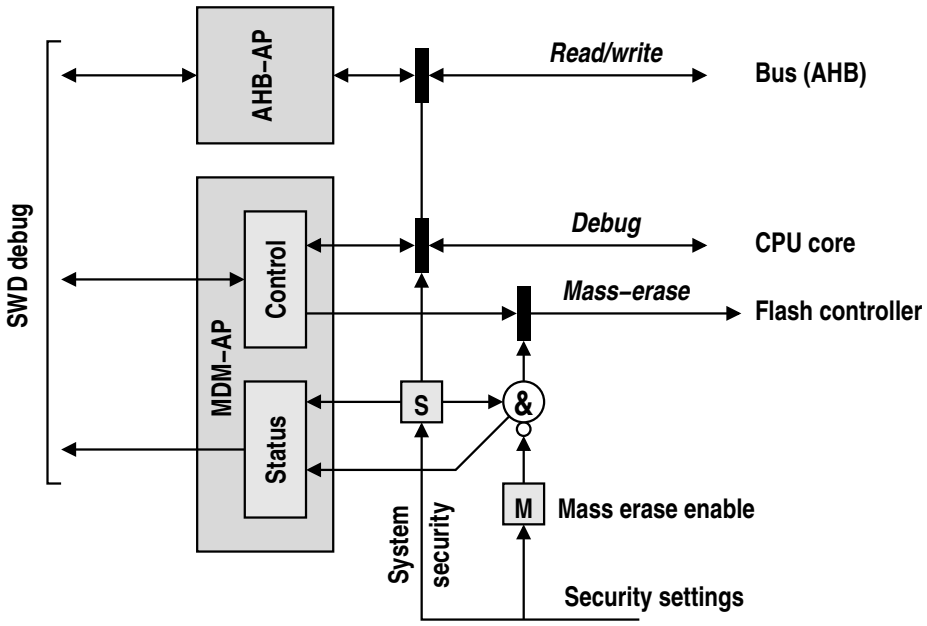

Figure 2: Conceptual drawing of how the "System security" and "Mass erase enable" settings affect the rest of the system. "S" and "M" represent the register bits that hold the setting currently in effect. These bits are initialized on reset according to the values in the "Flash configuration field."

The debug port connects to so-called "access ports" (AP). AHB-AP gives access to the AHB bus while MDM-AP gives access to a number of debug and reset control functions, allows to determine the security status, and provides a direct access to the Flash mass-erase operation, without going through the register interface of the Flash controller.

If the system is "secure", access to the bus through AHB-AP is blocked and debugging operations are also disabled. If mass-erase is not enabled *and* the system is "secure", then mass-erase through the debug port is disabled.

2.2 Backdoors

There are two additional twists to the above: one can establish a backdoor access and there is also a largely undocumented method for Freescale factory access. Both have the ability to remove the "secure" status of the MCU without having to change the Flash content for it.

The backdoor consists of a secret bit string – chosen when programming the MCU – stored in Flash and a Flash command to verify backdoor access. If the key provided with the verify command is identical to the one stored in Flash, the Flash controller clears the "secure" flag.

This requires the cooperation of the code running on the MCU. It is *not* a way to gain control over firmware gone bad.

The second backdoor is called "Freescale factory access" and is intended for bypassing security when chips are returned to Freescale for problem analysis.

Figure 3 zooms in on the "secure" bit of figure 2 and shows the structure of the backdoor mechanisms. Note that it is not possible to determine from SWD whether Freescale factory access is enabled.

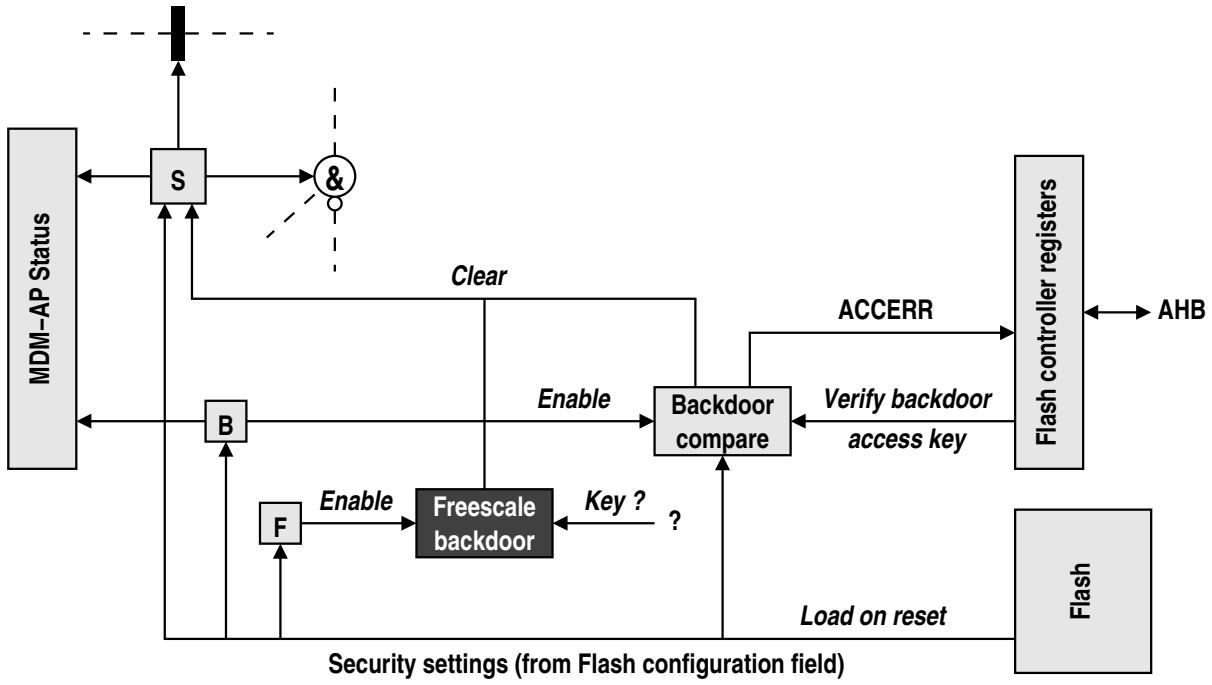

Figure 3: Conceptual drawing of the backdoor mechanisms. "S" is the "secure" flag, "B" is set when the backdoor is enabled, and "F" is set when Freescale factory access is allowed. Freescale factory is largely undocumented, hence the almost black box.

3 Kinetis L Flash change protection

The Flash can be protected against writing and erasing. This is accomplished through the FPROT registers in the Flash controller. Just like FSEC, these registers are loaded from the Flash configuration field at reset time.

Figure 4 shows how the FPROT register interacts with the rest of the system.

The volatile protection bits can be changed after reset by code running on the MCU or – if allowed – from the debug interface, but only towards stricter protection.

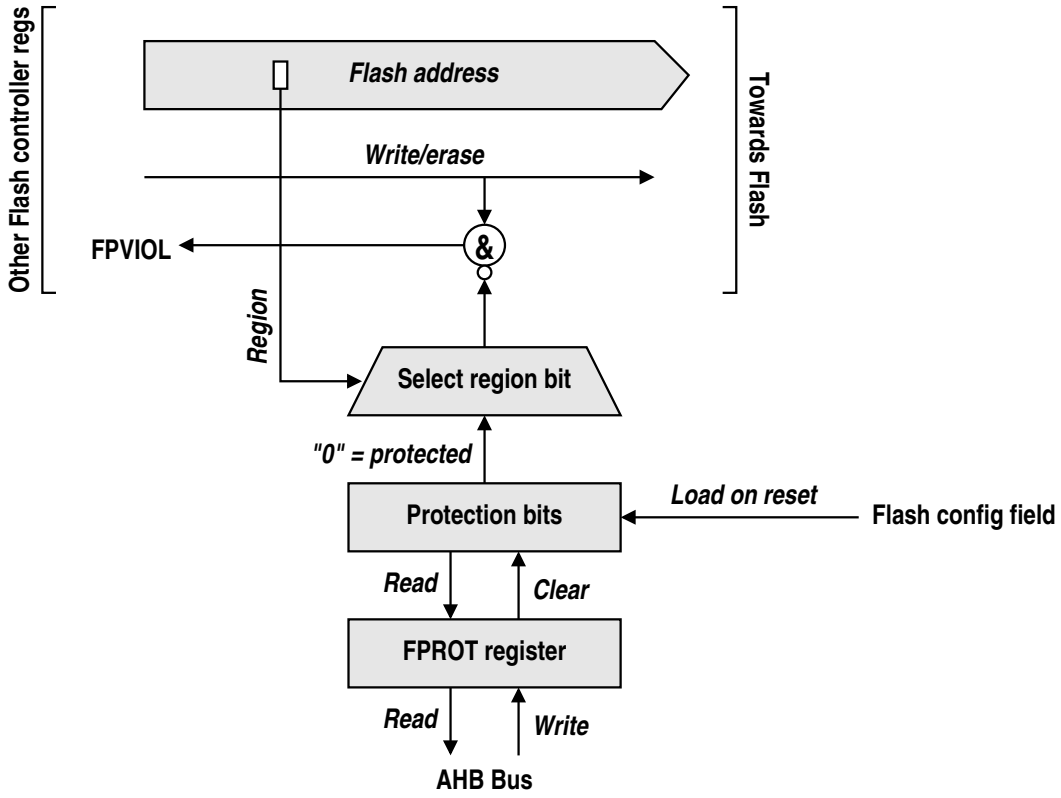

Figure 4: @@@@

There are up to 32 protection bits. Each protects a Flash region of $\frac{1}{32}$ the total Flash size or 1 kB, whichever is larger. E.g., in a device with 128 kB of Flash, there are 32 regions of 4 kB each.

A Flash mass-erase sets the protection bits to one and thus allows writing and erasing of the Flash. @@@ need reset ?

4 Practical considerations

The following

4.1 Reading from Flash

4.2 Writing to Flash

References

[1] *Anelok Security Architecture*, TBD.

[2] Freescale. *KL26 Sub-Family Reference Manual*, Document Number: KL26P121M48SF4RM, Rev. 3.2, October 2013.