Application Note

K663 and H663 to M519 Migration Guide

Headquarters, Europa

SpringCard SAS
2, voie la Cardon
Parc Gutenberg
91120 Palaiseau
FRANCE

Phone: +33 (0)1 64 53 20 10

Americas

SpringCard Inc.
185 Alewife Brook Parkway,
ste 210
Cambridge, MA 02138
USA

Email: sales@springcard.com

www.springcard.com

Document Identification

Category	Application notes
Group/Family	SpringCore / M519
Reference & Version	PNA24092-AA
Date	
Diffusion	Public
Keywords	M519, PC/SC, CCID, serial, SDK, K663, H663, EOL

Revision History

Version	Date	Author	QC	Description
AA	09/08/2024	JDA	CFE	Initial release

Table of Contents

1 Introduction

1.1 Overview

SpringCard has announced the gradual phase-out of the K663 and H663 modules and encourages its customers to migrate to the SpringSeed M519.

This document aims to identify and explain all the necessary changes, and to guide developers and integration engineers through the migration process.

1.2 Context and motivations

1.2.1 Phased-out families: ‘K’ and ‘H’

SpringCard launched the first module in the 'K' series, the K531, in 2003. The K632, an evolution of the K531, was later introduced, and in 2012 it was succeeded by the K663. All three modules in the 'K' series share the same pinout and metal cover. They operate over a serial communication line and implement the SpringProx 'Legacy' protocol.

The K663/RDR is a variant of the K663. It features the same hardware but runs a different firmware, enabling it to autonomously fetch data from a contactless card in a "Smart Reader" mode, before transmitting this data to the host using a lightweight protocol ('MK1').

The 'H' family consists of a single module, the H663, which was also launched in 2012. The H663 uses the same metal cover as the 'K' series but with additional pins, as it primarily communicates via USB and includes a smartcard interface (ISO 7816). The H663 typically operates in PC/SC (CCID) mode.

Both the 'K' and 'H' modules are designed for manual soldering using through-hole technology. They all require an external antenna to complete the implementation of the NFC/RFID HF feature; their antenna must be symmetrical and tuned to 50 ohms to ensure optimal performance.

Illustration 1: K663-XXX

SpringCard has also designed ready-to-use variants of these modules, which include an integrated antenna. These variants are:

K663 family:

- K663-XXX and K663/RDR-XXX, with the serial interface options depending on the configuration (RS-TTL, RS-232, or RS-485).

H663 family:

- H663-USB: integrated antenna, no contact smart card.
- TwistyWriter-HSP: remote, balanced antenna; 1 ID-000 SAM slot.
- CrazyWriter-HSP: remote, unbalanced antenna; 1 or 4 ID-000 SAM slots and 1 ID-1 card slot.

1.2.2 New family: SpringSeed 'M'

In 2023, SpringCard launched the SpringSeed **M519**, a single module that is designed to replace both the 'K' and 'H' families.

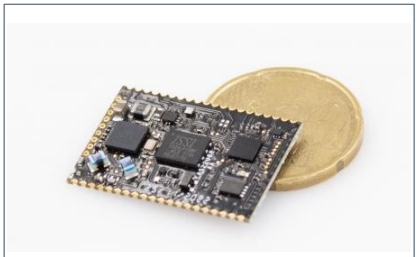

Illustration 2: M519

The continuous evolution in electronics has brought about significant flexibility by enabling the consolidation of multiple features (such as Serial and USB interfaces, Coupler mode and Smart Reader mode) into such a single module. Functions that previously required separate hardware and firmware can now be integrated, simplifying design and reducing the overall system complexity, and cost.

Advances in miniaturisation and technical expertise have made it possible to offer a module without a metal cover, with components mounted on a single side of the PCB. This design allows the module to be placed and soldered like a standard SMD component, which significantly reduces costs for manufacturers who integrate it into their products.

Adhering to state-of-the-art electronics (with the adoption of the latest high-end NFC front-end from NXP, the PN5190) has also resulted in notable improvements in performance. The communication range with contactless cards could be significantly increased without any additional power consumption, or a performance equivalent to earlier devices could still be reached while reducing overall energy requirements. Furthermore, advanced components support new technologies and introduce new functionalities, making a new module more versatile and future-proof.

To achieve all these improvements, the M519 departs from the historical format of the ‘K’ and ‘H’ modules. It also requires a substantially different antenna, tuned to 20 ohms instead of 50 ohms. In some cases, implementing a new communication protocol or at least accommodating new configuration tools is necessary to fully leverage the M519's capabilities.

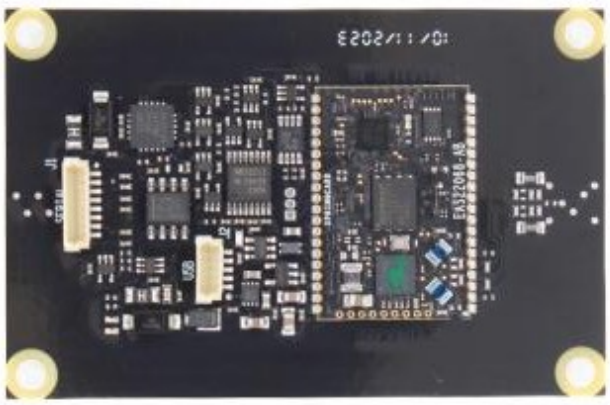

Illustration 3: M519-SUV

A new ready-to-use module with an integrated antenna, the **M519-SUV**, replaces all variants of K663-XXX and H663-USB. The M519-SUV has the same 69x45mm dimensions and the same mounting holes as the products it replaces, and is able to address all the use cases (USB, RS-TTL, RS-232 or RS-485) with a single hardware.

Another ready-to-use module with a remote antenna, the **M519-USS**, replaces the TwistyWriter-HSP.

Important notice: the M519, the M519-SUV and all products based on the M519 must be used through its USB "XOR" (exclusive-or) its Serial interface. It is not possible to use both interfaces at the same time.

1.3 Structure of the document

This document explains how to migrate from an 'H' or 'K' family module to the M519 module or its antenna variant, the M519-SUV. It is structured into four main sections:

- A quick reference guide that brings together the various relevant products, highlights their main features, and provides an overview of the migration paths at a glance.
- The hardware changes, split into a specific subsection the bare modules (M519) and another for the modules with antenna (M519-SUV, M519-USS).
- The necessity to configure the M519 for the needs of the end-application, and the impact on the production/integration line or on the end-product itself.
- The software changes, with different subsections dedicated to operation in coupler mode (CCID or Legacy) or as a Smart Reader and an RFID Scanner. In particular, a subsection will be dedicated to the complete migration from the SpringProx Legacy protocol to the CCID (PC/SC) protocol, which, although it may initially seem a bit more complex, actually offers not only a better support of all today's card technologies, but also a better development experience with code portability, a structure more aligned with the OSI model, and the ability to validate each module independently.

1.4 Related Documents

1.4.1 Documents available as PDF

Reference	Title / Description
PFT22217	M519 Data Sheet and Integration Guide
PFT23234	M519-SUV Data Sheet and Integration Guide
PMD23175	M519-SRK Getting Started Guide
PMI23209	M519-SUV Getting Started Guide
PNA23174	Using the M519 in PC/SC Coupler mode over a Serial interface
PNA23207	Using the M519 in PC/SC Coupler mode over a USB interface
PNA23208	Using the M519 in Smart Reader or RFID Scanner mode
PNA23189	Using the M519 in SpringProx Legacy mode

1.4.2 Online Material

Documentation of the SpringCore firmware.

https://docs.springcard.com/books/SpringCore/Welcome

SpringCard Tech Zone, the blog of the R&D Team

https://tech.springcard.com/

1.5 Glossary

Reference	Title / Description
APDU	Application-Protocol Datagram Unit. The name comes from the OSI model to describe end-to-end command/response exchanges, between two applications.
ATR	Answer To Reset. This is the identification message that a smart card sends when starting up.
CCID	Chip Card Interface Device, the USB specification for smart card couplers
PC/SC	Personal Computer/Smart Card
RS-232	A standard serial communication standard, with 2 independent lines (RX/TX) for full-duplex communications. Electrical levels are - 0: +3 to +15V - 1: -15 to -3V
RS-485	A standard serial communication standard, with 1 differential pair allowing half-duplex communications only. Electrical levels are: - 0: $V_A < V_B$ - 1: $V_A > V_B$
RS-TTL	Not a standard, but a shortcut to describe serial communication with 2 independent lines (RX/TX) for full-duplex communications and electrical levels that are TTL or TTL/CMOS: - 0: < 0.8V - 1: > 2V
USB	Universal Serial Bus

2 Quick reference guide

The tables below show the different migration paths. The 'effort' column represents our estimation of the complexity of the work required for both hardware and software.

2.1 Bare modules

Phased-out product	Replacing product	Migration path / Remarks	Effort
K663	**M519** *in CCID (PC/SC) mode*	- Adjust power supply and serial interface if needed - Design new antenna - Port the CCID library to the host - Migrate the software application from SpringProx Legacy to CCID - Qualify the new solution	*****
	M519 *in Legacy mode*	- Adjust power supply and serial interface if needed - Design new antenna - Adjust your implementation of the SpringProx software library - Qualify the new solution *(not recommended)*	****
K663/RDR	**M519** *in Smart Reader mode*	- Adjust power supply and serial interface if needed - Design new antenna - Transpose the Card Processing Templates - Qualify the new solution	***
H663	**M519** *in CCID (PC/SC) mode*	- Adjust smart card interface if needed - Design new antenna - *Potentially:* reconfigure or upgrade CCID driver to support new product ID - Qualify the new solution	***
H663/RDR	**M519** *in RFID Scanner mode*	- Design new antenna - *Potentially:* reconfigure or upgrade HID keyboard driver to support new product ID - Transpose the Card Processing Templates - Qualify the new solution	***

2.2 Modules with antenna, Coupler mode, Serial interface

Phased-out product	Replacing product	Migration path / Remarks	Effort
K663-TTL **K663-XXX** *configured for RS-TTL*	**M519-SUV** *in CCID (PC/SC) mode, interface configured for RS-TTL*	- Port the CCID library to the host - Migrate the software application from SpringProx Legacy to CCID - Qualify the new solution	***
	M519-SUV *in Legacy mode, interface configured for RS-TTL*	- Adjust your implementation of the SpringProx software library - Qualify the new solution *(not recommended)*	*
K663-232 **K663-XXX** *configured for RS-232*	**M519-SUV** *in CCID (PC/SC) mode, interface configured for RS-232*	- Port the CCID library to the host - Migrate the software application from SpringProx Legacy to CCID - Qualify the new solution	**
	M519-SUV *in Legacy mode, interface configured for RS-232*	- Adjust your implementation of the SpringProx software library - Qualify the new solution *(not recommended)*	*
K663-485 **K663-XXX** *configured for RS-485*	**M519-SUV** *in CCID (PC/SC) mode, interface configured for RS-485*	- Port the CCID library to the host - Migrate the software application from SpringProx Legacy to CCID - Qualify the new solution M519 supports SpringProx Legacy over a full-duplex interface only. CCID operation is possible over an RS-485 line, but with more complexity for the CCID library.	*****

2.3 Modules with antenna, Smart Reader mode, Serial interface

Phased-out product	Replacing product	Migration path / Remarks	Effort
K663/RDR-TTL K663/RDR-XXX *configured for RS-TTL*	M519-SUV *in Smart Reader mode, interface configured for RS-TTL*	- Transpose the Card Processing Templates - Qualify the new solution	**
K663/RDR-232 K663/RDR-XXX *configured for RS-232*	M519-SUV *in Smart Reader mode, interface configured for RS-232*	- Transpose the Card Processing Templates - Qualify the new solution	**
K663/RDR-485 K663/RDR-XXX *configured for RS-485*	M519-SUV *in Smart Reader mode, interface configured for RS-232*	- Transpose the Card Processing Templates - Qualify the new solution	**

2.4 Modules with antenna, Coupler mode, USB interface

Phased-out product	Replacing product	Migration path / Remarks	Effort
H663-USB	M519-SUV *in CCID (PC/SC) mode*	- *Potentially:* reconfigure or upgrade CCID driver to support new product ID - Qualify the new solution	*
TwistyWriter-HSP	M519-USS *in CCID (PC/SC) mode*	- *Potentially:* reconfigure or upgrade CCID driver to support new product ID - Qualify the new solution	*
CrazyWriter-HSP	—	Contact us to evaluate alternative solutions	

2.5 Modules with antenna, RFID Scanner mode, USB interface

Phased-out product	Replacing product	Migration path / Remarks	Effort
H663/RDR-USB	M519-SUV *in RFID Scanner mode*	- *Potentially:* reconfigure or upgrade HID keyboard driver to support new product ID - Transpose the Card Processing Templates - Qualify the new solution	*

3 Hardware migration – Bare modules

Due to the changes in dimensions and the significant alterations in characteristics, adopting the X module should be considered a new project, which notably requires the design of an entirely new antenna.

Refer to chapters 7, 8 and 9 of [PFT22127] for all the details.

SpringCard has a strong experience in designing products based on the M519 and also in designing NFC/RFID HF antennas. Don’t hesitate to contact us should you need the assistance of an expert in designing or qualifying your own product.

4 Hardware migration – Modules with antenna

4.1 Serial products

4.1.1 K663-XXX in RS-TTL or K663-TTL to M519-SUV

The M519-SUV has the same external dimensions (69x45mm) and the same mounting holes as its predecessors. It is also thinner. Therefore, it can be easily integrated into existing mechanical setups.

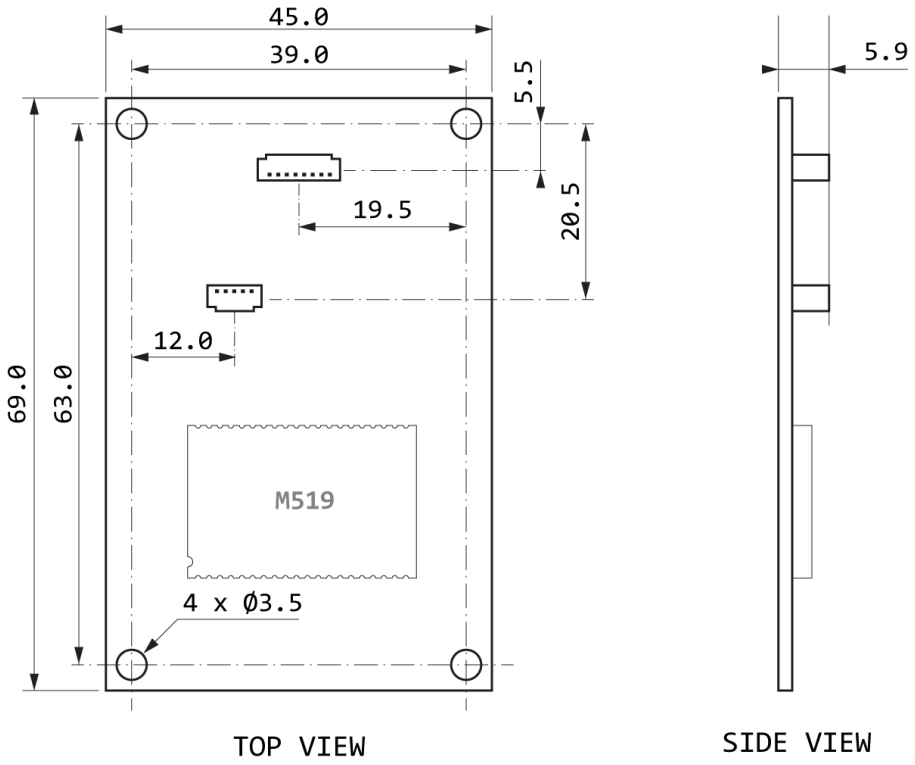

Illustration 4: Mechanical specifications, M519-SUV

The M519-SUV has the same 8-position JST BM08B-SRSS-TB connector as its predecessors for its Serial interface and power supply, with the same pinout and electrically compatible levels (once it has been properly configured). Therefore, it can be easily integrated into existing hardware setups.

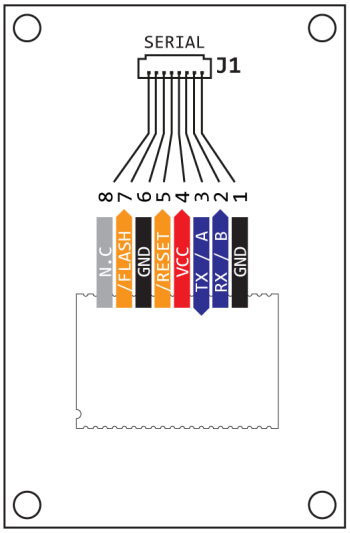

Illustration 5: Pinout of J1, M519-SUV

4.1.2 K663-XXX in RS-232 or K663-232 to M519-SUV

Same as above.

4.1.3 K663-XXX in RS-485 or K663-485 to M519-SUV

Same as above.

4.2 USB products

4.2.1 H663-USB to M519-SUV

The M519-SUV has the same external dimensions (69x45mm) and the same mounting holes as its predecessor. It is also thinner. Therefore, it can be easily integrated into existing mechanical setups *(cf. illustration 4 page 17)*.

The M519-SUV has the same 5-position JST SM05B-SRSS-TB connector as its predecessor for its USB interface. Therefore, it can be easily integrated into existing hardware setups.

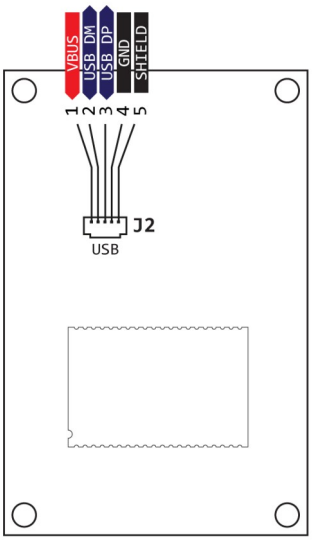

Illustration 6: Pinout of J2, M519-SUV

4.2.2 TwistyWriter-HSP to M519-USS

The main board of the M519-USS has the same external dimensions (69x45mm) and the same mounting holes as the main board of its predecessor. It is also thinner. Therefore, it can be easily integrated into existing mechanical setups.

The same applies to the antenna board.

The M519-USS has the same 5-position JST SM05B-SRSS-TB connector as its predecessor for its USB interface. Therefore, it can be easily integrated into existing hardware setups.

4.3 Qualification

Please remember that even though the new OEM product you are integrating is CE/FCC certified, due to the substantial modification of the end product, it is necessary to re-qualify it for RF performance compliance.

5 Configuring the M519 to meet application requirements

5.1 Understanding the M519 configuration process

5.1.1 M519 configuration registers

The configuration of the M519 is stored in its non-volatile memory, which is structured in registers. Each configuration register has a 2-byte address, ranging from $_H$0200 to $_H$02FF. The complete list of registers and their functions is documented online at

https://docs.springcard.com/books/SpringCore/Non_volatile_memory/Configuration

5.1.2 M519 Card Processing Templates registers

In both CCID (PC/SC) and SpringProx Legacy modes, the M519 operates as a 'transparent' coupler, meaning the host application has full control over the transaction with the card. Conversely, in Smart Reader and RFID Scanner modes, the M519 must execute a transaction with the card independently to retrieve the small amount of data (the token) required by the application, before transmitting this token to the host.

The transaction performed by the M519 when used as a reader is governed by 1 to 4 Card Processing Templates. Their configurations are stored in registers $_H$0310 to $_H$034F. Documentation for the reader's Template engine is available online at

https://docs.springcard.com/books/SpringCore/Smart_Reader_Operation

5.1.3 SpringCard Companion

SpringCard Companion is the new all-in-one solution for managing, configuring, and updating the latest generation of SpringCard 'SpringCore' devices, including the M519. SpringCard Companion is a hybrid application that combines a clean, efficient, and always up-to-date web front-end with a lightweight local software component, the Companion

Service, which serves as the gateway between the cloud and the devices connected to your computer or local network.

Using SpringCard Companion is the preferred method for creating and validating a configuration for the M519. Developers and those responsible for integrating the M519 into an end-product are strongly encouraged to familiarise themselves with SpringCard Companion, which guides them through the process of defining their configurations as well as setting up Templates for the readers.

Once the configuration has been defined, a single click is sufficient to trigger its transmission to the M519, provided this OEM product is directly connected to the PC running SpringCard Companion. However, for the industrialisation of the integration process, another method should be used to automatically transmit the configuration to the OEM product without a manual intervention from an operator. To facilitate this, SpringCard Companion allows the configuration to be exported as a JSON file, that can feed an unattended configuration tool like SpringCardConfig.exe (see below).

SpringCard Companion is also able to load a new firmware in the device. The unattended counterpart for this function is SpringCardFlash.exe (see below).

5.2 Writing the configuration into the M519 OEM products during the manufacturing process

In most situations, the factory manufacturing the end product will receive SpringCard OEM products with a default configuration, which is not the configuration required for the end application. Therefore, the OEM product must be configured either before or during its integration into the end product. Additionally, it is often necessary to upgrade the OEM product's firmware to the latest version during integration or even in the field. As a result, the end-product and its production process must be adapted to include the commissioning of the SpringCard OEM product. This section describes the key steps to achieve this.

There are two exceptions to the need to adjust the production system: firstly, integrators with very low volumes can perform the configuration and updates manually using the

basic tools provided by SpringCard before proceeding with the integration; conversely, integrators with very large volumes can request the creation of a specific part number with their own configuration preloaded at the SpringCard factory, along with a locked firmware version.

5.2.1 Using the SpringCore Tools during the manufacturing process

To transmit a configuration file (in JSON format) to the M519 without using SpringCard Companion, SpringCard recommends using SpringCoreConfig.exe, a command-line utility. This utility can be easily invoked within a script or from third-party software to automate the configuration of the M519 during the production phase. SpringCoreConfig.exe can also be used to write to the device's registers individually, without relying on a global JSON configuration file.

The documentation of SpringCoreConfig.exe is available online at

https://docs.springcard.com/books/Tools/SpringCore/SpringCoreConfig

The SpringCard TechZone blog contains an article on how to use SpringCoreConfig.exe:

https://tech.springcard.com/2021/writing-a-configuration-with-springcoreconfig-exe/

Another useful utility from the SpringCore Tools suite is SpringCardFlash.exe, which is designed to load new firmware onto a device. Its documentation is available online at

https://docs.springcard.com/books/Tools/SpringCore/SpringCoreFlash

The SpringCard TechZone blog also contains an article on how to use SpringCoreFlash.exe:

https://tech.springcard.com/2021/flashing-a-new-firmware-with-springcoreflash-exe

An important point must be considered: communicating with the M519 via USB using SpringCoreConfig.exe and is SpringCoreFlash.exe is straightforward and always possible. However, communicating with the M519 using SpringCoreConfig.exe and SpringCoreFlash.exe via a serial connection is not always feasible.

In the case of the M519-SUV, the electrical level of the product's serial interface is defined by configuration; thus, until the product has the correct configuration, its serial interface might not have the expected level for the interface that links it to the host.

Another situation arises when the product is configured for CCID (PC/SC) operation over the serial interface. In this case, the SpringCore Direct protocol, which would allow for reconfiguration, is unavailable because it cannot be multiplexed with the CCID protocol.

Therefore, even if the end-product utilises the product's serial interface, it is still advisable to also provide access to the USB interface of the OEM product, allowing the production system to run SpringCoreConfig.exe and SpringCoreFlash.exe easily, regardless of the previous configuration.

5.2.2 Writing the configuration directly from USB host software

If the end-product functions primarily as a USB host (such as a Windows, Linux, or MacOS computer, or an embedded system like a Raspberry Pi or equivalent running Linux) it is often much simpler to delegate the configuration of its M519-based device to the finished product itself, rather than arranging for pre-configuration of the device by another party during the production process.

SpringCoreConfig.exe is written in .NET and can run on Windows as well as Linux and MacOS platforms, meaning the integrator does not need to develop their own configuration software.

The precautions to take are:

1. Ensure that the main application is not launched until the product is properly configured.
2. Ensure that SpringCoreConfig.exe has direct access to the hardware via the WinUSB or LibUSB interfaces (on Unix systems, this typically requires running the execution with root privileges).
3. Use the return code of SpringCoreConfig.exe (0 = OK, non-0 = failure) to notify the production system and its operator if a product has not been correctly configured.

Alternatively, the integrator may develop a custom software on top of WinUSB or LibUSB, using the SpringCore Direct protocol to send configuration commands to the device. The documentation of the protocol is available at

https://docs.springcard.com/books/SpringCore/Host_Protocols/Direct_Protocol

5.2.3 Writing the configuration directly from Serial host software

5.2.3.1 M519 bare module

Communicating with the M519 over a serial interface is straightforward, but only once you know which protocol the device is running. The only truly robust approach for a production system is to force the protocol to be used.

The M519 has two pins, MODE0 and MODE1, which allow its configuration to be temporarily overloaded, ignoring the values that have been entered in the registers.

In normal use, both MODE0 and MODE1 pins are left floating or set to 1, but, is both MODE0 and MODE1 are tied to 0 on startup, the M519 enters a fail-safe mode, and activate the SpringCore Direct protocol at default bitrate (38400bps).

The production bench or the hardware of the end-product must be adapted to drive MODE0 and MODE1 to activate this fail-safe mode.

Once the device is running in fail-safe mode, the production software or the application in the end-product's host processor may write the new configuration using the SpringCore Direct protocol over the Serial interface.

The documentation of the SpringCore Direct protocol over the Serial interface is available at

https://docs.springcard.com/books/SpringCore/Host_Interfaces/Serial/SpringCore_Direct

The commands to write a configuration in the M519 are documented at

https://docs.springcard.com/books/SpringCore/Host_Protocols/Direct_Protocol/CONTROL_class/Configuration

5.2.3.2 M519-SUV module on antenna

The electrical level of the serial interface of the M519-SUV is defined by configuration; thus, until the product has the correct configuration, its serial interface might not have the expected level for the interface that links it to the host. Also, if the product is configured

for CCID (PC/SC) operation, the SpringCore Direct protocol, which would allow for reconfiguration, is unavailable because it cannot be multiplexed with the CCID protocol. That's why SpringCard offers different order codes for the M519-SUV, each corresponding to a basic configuration.

Please refer to [PFT23234] for the list of available order codes, and select the version of the M519-SUV with the configuration that already matches your application's requirements for the electrical level of the serial interface (register $_{H}$029F) and for the operating mode and protocol (register $_{H}$02C0).

Important: when writing a new configuration during production or in the field, ensure that you always rewrite these two registers with their initial values, as altering them could permanently prevent the host from communicating with the device.

Hint: when the M519-SUV is starting-up, its two LEDs light for a few seconds, to let the user know how the serial interface is configured:

- LED 1 ON and LED 2 ON → RS-TTL
- LED 1 ON and LED 2 OFF → RS-232
- LED 1 OFF and LED 2 ON → RS-485

Both LEDs OFF denotes an invalid configuration.

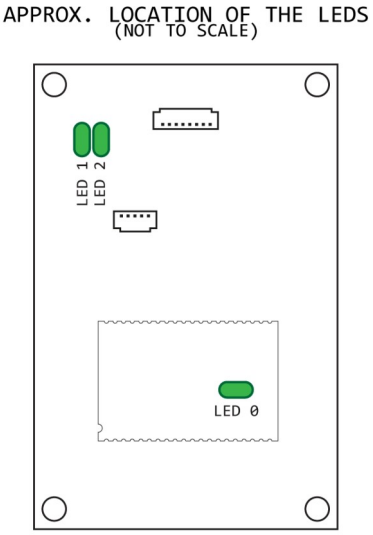

Illustration 7: Location of the LEDs, M519-SUV

5.3 Reference configurations

5.3.1 Summary of the registers involved

Phased-out product	Configuration to be set in M519-based product	Registers and details
K663 **K663-TTL** **K663-232** **K663-485** **K663-XXX** **H663** **H663-USB** **TwistyWriter-HSP** **CrazyWriter-HSP**	CCID (PC/SC) mode	- Set register $_H$02C0 to $_H$02 to select CCID (PC/SC) mode - *For M519-SUV only:* select the electrical level of the serial interface (TTL, 232, 485) in register $_H$029F
K663 K663-TTL K663-232 K663-485 K663-XXX	SpringProx Legacy mode *(not recommended)*	- Set register $_H$02C0 to $_H$01 to select SpringProx Legacy mode - *For M519-SUV only:* select the electrical level of the serial interface (TTL, 232, 485) in register $_H$029F
K663/RDR **K663/RDR-TTL** **K663/RDR-232** **K663/RDR-485** **K663/RDR-XXX**	Smart Reader mode, serial interface, MK1 protocol	- Set register $_H$02C0 to $_H$04 to select Smart Reader mode - *For M519-SUV only:* select the electrical level of the serial interface (TTL, 232, 485) in register $_H$029F - Set register $_H$02A0 to $_H$0103 to have the same behaviour as K663/RDR - Define the Templates in registers $_H$03xx
H663/RDR **H663/RDR-USB**	RFID Scanner mode	- Set register $_H$02C0 to $_H$03 to select RFID Scanner mode - Define keyboard layout in register $_H$02A6, the prefix in register $_H$02A8 and the suffix in register $_H$02A9 - Define the Templates in registers $_H$03xx

5.3.2 Operating mode

Set the operating mode in register $_H$02C0.

Supported values are:

- $_H$01 for SpringProx Legacy mode *(not recommended)*
- $_H$02 for CCID (PC/SC) mode
- $_H$03 for RFID Scanner mode
- $_H$04 for Smart Reader mode

Other values shall not be used with the M519.

Detailed documentation is available at

https://docs.springcard.com/books/SpringCore/Non_volatile_memory/Configuration/Main_configuration/Profile

5.3.3 Electrical level of the Serial interface (M519-SUV only)

Set the electrical level of the Serial interface in register $_H$029F.

Supported values are:

- $_H$00 for RS-TTL
- $_H$01 for RS-232
- $_H$02 for RS-485

Other values shall not be used with the M519.

Detailed documentation is available at

https://docs.springcard.com/books/SpringCore/Non_volatile_memory/Configuration/Serial/Driver

5.3.4 MK1 protocol and related options (Smart Reader mode)

The M519 in Smart Reader mode supports a variety of protocols, with different behaviours upon startup. Set the register $_H$02A0 to $_H$0103 to have exactly the same behaviour as K663/RDR.

Detailed documentation is available at

https://docs.springcard.com/books/SpringCore/Non_volatile_memory/Configuration/Smart_Reader/Features

5.3.5 Keyboard layout and related options (RFID Scanner mode)

The M519 in Smart Reader mode supports a variety of keyboard layout. The keyboard layout in the device must match the current keyboard layout of the computer; otherwise, the PC cannot correctly display the characters that the reader transmits by simulating keyboard input.

The keyboard layout is defined by register $_H$02A6. You may copy into this register the value from the equivalent register $_H$A0 from the H663/RDR.

Detailed documentation is available at

https://docs.springcard.com/books/SpringCore/Non_volatile_memory/Configuration/Smart_Reader/Keyboard_layout

The reader may transmit a fixed prefix sequence before the actual data. The prefix is defined by register $_H$02A8. You may copy into this register the value from the equivalent register $_H$A2 from the H663/RDR.

Detailed documentation is available at

https://docs.springcard.com/books/SpringCore/Non_volatile_memory/Configuration/Smart_Reader/Prefix

The reader may transmit a fixed suffix sequence after the actual data. The suffix is defined by register $_H$02A9. You may copy into this register the value from the equivalent register $_H$A3 from the H663/RDR.

Detailed documentation is available at

https://docs.springcard.com/books/SpringCore/Non_volatile_memory/Configuration/Smart_Reader/Suffix

5.3.6 Card Processing Templates (Smart Reader and RFID Scanner modes)

The implementation of the Card Processing Templates in the M519 is very similar to that in the K663/RDR and H663/RDR (yet the M519 supports new card technologies and offers more flexibility in some Templates).

The easiest migration path is to copy and paste the configuration of the Templates from the K663/RDR or H663/RDR to the M519. The only difference lies in the register addresses: in the K663/RDR and H663/RDR, the Templates use registers $_H$10 to $_H$4F; on the M519, the Templates use registers $_H$0310 to $_H$034F.

The translation is straightforward: register $_H$10 corresponds to register $_H$0310, register $_H$11 to register $_H$0311, and so on.

6 Software migration — Serial devices

6.1 Coupler: K663 SpringProx Legacy to M519 CCID (PC/SC)

Using a standards-based software stack like PC/SC offers several advantages over relying on a vendor-specific API like the SpringProx API, especially since the latter is a legacy solution, designed nearly 25 years ago, before the ISO/IEC 14443 and 15693 standards were ratified, and when 99.9% of the use cases where based on Mifare Classic cards.

PC/SC is device-agnostic, ensuring compatibility with a wide range of devices from different manufacturers (not just SpringCard's). This reduces dependency on a single vendor, promotes flexibility in hardware choice, and makes the software solution future-proof. Additionally, PC/SC is inherently cross-platform, facilitating easier integration into diverse environments and ensuring that future migrations are seamless.

That's why SpringCard recommends that all clients still using SpringProx Legacy with the K663 take advantage of the migration to the M519 to also transition to CCID (PC/SC).

Migrating from SpringProx Legacy to CCID (PC/SC) over a serial connection is a significant project that may impact the core architecture of the application.

Here are the two important points to understand before embarking on this transition:

- ***Application Control and Communication Model:*** with SpringProx, the coupler acts purely as a slave to the application; the entire operation can generally be synchronous (sending a command, waiting, receiving the response), including card detection. In contrast, with CCID (PC/SC), the coupler manages card polling in the background and can notify the application as soon as a card is detected. This event-driven behaviour is only possible if the underlying host implementation allows the host application to receive messages asynchronously at any time, without being explicitly in a blocking wait state for a response.
- ***APIs and Functionality Exposure:*** with SpringProx, the coupler has limited intelligence and exposes relatively low-level functions specific to each card technology. An application handling Mifare Classic badges will use the low-level

Mifare Classic APIs, an application handling ISO/IEC 15693 tags will use the low-level 15693 APIs, and an application handling contactless smart cards like Desfire will use the T=CL APIs to send APDUs, etc. Conversely, with a CCID (PC/SC) coupler, everything is managed through APDUs, including access to all memory cards like Mifare Classic or ISO/IEC 15693. The coupler incorporates the necessary logic to translate generic APDUs (such as READ BINARY, UPDATE BINARY) into appropriate low-level read or write commands specific to the card. This has the advantage of being universal and portable, but it does require rewriting the application's logic to assemble and send APDUs, instead of simply calling a function with the correct parameters.

Suggested migration roadmap:

1. **Analyse the existing application and update its documentation**, ensuring you have a thorough understanding of the transaction being performed with the card. When the application supports multiple card types, it's also crucial to clearly understand how the card type is identified and the priority between different processing routines. This step is particularly important for older applications that have not undergone rigorous and continuous maintenance since their creation.

2. **Follow [PNA23174]** to implement “CCID over Serial protocol” and to port SpringCard “PC/SC-Like” software library to your host. The SDK that accompanies this application note is available at

   https://github.com/springcard/springcard-ccid-serial

   Make sure the sample application (§ 4.5 in [PNA23174]) is working as expected before proceeding to next step.

3. **Rewrite the existing application** step by step.

   - All calls to card lookup functions (SPROX_Find, SPROX_A_Select, SPROX_B_Select, SPROX_Iso15693_Select, etc.), are replaced at once by waiting for the arrival of a card, which is obtained through function ScardGetStatusChange.
   - If the application must control when the coupler is allowed to poll, and when it shall switch off its RF field (because the application is not ready to

process a card or for power saving consideration), use SCardControl with command $_H$58 $_H$22 $_H$FF to stop the polling, and command $_H$58 $_H$23 $_H$FF to restart the polling. Details can be found at

https://docs.springcard.com/books/SpringCore/Host_Protocols/Direct_Protocol/CONTROL_class/CCID/CCID_GET_SLOT_NAME

- Call SCardConnect to get access to the newly inserted card and retrieve its ATR.
- Call SCardTransmit ($_H$FF $_H$CA $_H$00 $_H$00 $_H$00) to get the UID of the card. Other protocol-level values (like ATQ and SAK) may be queried as well, see

https://docs.springcard.com/books/SpringCore/PCSC_Operation/APDU_Interpreter/Standard_instructions/GET_DATA

- Use the ATR and the protocol-level values (UID, ATQ, SAK) to identify the card technology and branch to the existing processing flow.
- For T=CL cards (ISO/IEC 14443-4)
  - Calls to SPROX_Tcl_Exchange are directly replaced by calls to SCardTransmit, with the same APDU as parameter.
  - Calls to SPROX_TclA_GetAts, SPROX_TclA_Pps and SPROX_TclB_Attrib must be removed, since the coupler manages the activation on its own.
  - Calls to SPROX_Tcl_Deselect are replaced by SCardDisconnect.
- For other card technologies, function calls must be rewritten to assemble the correct APDU (starting with CLA=$_H$FF) before transmitting it with ScardTransmit. For reference, see

https://docs.springcard.com/books/SpringCore/PCSC_Operation/APDU_Interpreter

- Sending low-level commands to the card is also possible using the CL CONTROL APDU.

https://docs.springcard.com/books/SpringCore/PCSC_Operation/APDU_Interpreter/Vendor_instructions/CL_CONTROL

6.2 Coupler: K663 SpringProx Legacy to M519 SpringProx Legacy (not recommended)

For those who insist on continuing to use the M519 in SpringProx Legacy mode, it remains possible, albeit with several limitations and constraints.

Here are the key points to be aware of:

- The last version of the SpringProx Library (source code and DLL for Win32) is 1.80, released in 2014. No new version will be provided.
- The M519 does support the SpringProx 'BINARY' protocol only. The old OSI3964 protocol, the 'ASCII' protocol and the 'BINARY BUS' protocol are not available.
- Even if the hardware features a smartcard (SAM or ID-1) slot, the M519 does not permit to use it in Legacy mode.
- The M519 does not support the NXP ICODE1 chip that is pre-ISO/IEC 15693 (but all versions of ICODE-SLI are supported).
- Due to the change in the NFC interface, all low-level functions that directly access the NFC front-end shall not be used. This include:
  - SPROX_ReaderGetConsts, SPROX_ReaderSetConsts, SPROX_ReaderGetConstsEx, SPROX_ReaderSetConstsEx,
  - SPROX_ReaderGetRc500Id,
  - SPROX_WriteRCRegister.
- Due to the change in the NFC interface, all low-level functions that are inherited from the initial NXP (Philips Semiconductors) SDK for the MfRc500 are not available. This include:
  - All functions with the name starting with Mf500;
  - PcdSetTmo, PcdGetSnr, PcdReadE2, PcdWriteE2, PcdReset, PcdSetIdleMode, PcdClearIdleMode, ExchangeByteStream, ReadRC, WriteRC, SetRCBitMask, ClearRCBitMask.

- Due to the change in the hardware characteristics, all functions that deal with GPIO, non-volatile memory, etc, shall not be used. This include:
  - SPROX_ControlLed, SPROX_ControlBuzzer,
  - SPROX_ControlReadModeIO, SPROX_ControlReadUserIO, SPROX_ControlWriteUserIO,
  - SPROX_ReadStorage, SPROX_WriteStorage.

6.3 Smart Reader: K663/RDR to M519

The implementation of the 'MK1' protocol in the M519 is very similar to that in the K663/RDR. Provided the M519 is correctly configured, the migration is seamless.

7 Software migration — USB devices

7.1 Coupler: H663 CCID (PC/SC) to M519 CCID (PC/SC)

7.1.1 Driver

In CCID (PC/SC) mode, the OEM product conforms to the USB CCID standard (Card Coupling Interface Devices). It must be supported by a compatible driver running on the host. Several drivers are available.

The only requirement is that the driver must be associated with the M519's USB identifiers, where the Product ID is (of course) different from the H663's.

- ~~**H663**: `VendorID = 0x1C34, ProductID = 0x91B1`~~
- **M519 in CCID (PC/SC) mode**: `VendorID = 0x1C34, ProductID = 0x6212`

The procedure to associate the driver to the device depends on the system and on the driver.

7.1.1.1 Windows

SpringCard PC/SC Driver for USB Couplers [SD16055] supports the **M519** since version 22.09. The latest driver setup is available at

https://www.springcard.com/en/download/find/file/sd16055

Alternatively, run Windows Update after having connected the M519 to the computer.

A basic CCID PC/SC driver is included in Windows and is, by default, associated with any device that claims the CCID class. Unfortunately, this driver has several limitations, the most notable being that it supports only a single slot. Therefore, using it with SpringCard devices is not recommended.

7.1.1.2 Linux

PC/SC on Linux is provided by the PCSC-Lite open-source project. The accompanying CCID driver is developed and maintained by Ludovic Rousseau; the project page is available on GitHub at

https://github.com/LudovicRousseau/CCID

The M519 is supported since version 1.5.3 of the driver:

https://ccid.apdu.fr/ccid/shouldwork.html#0x1C340x6212

SpringCard has no affiliation with the independent and volunteer developers involved in these projects. For any specific support needs regarding PC/SC on Linux, please contact them through their website or open a ticket on their GitHub.

7.1.1.3 MacOS

The CCID driver in macOS is a derivative work of Ludovic Rousseau's CCID driver. Unfortunately, Apple does not keep up with the updates of the driver, and macOS 13.0 is still using CCID driver version 1.5.0, which does not yet support the **M519**. We hope Apple will provide an upgrade soon.

7.1.2 PC/SC API

Thanks to the layered model of PC/SC, as long as the coupler is supported by a driver, the developer of the end application will experience minimal changes when migrating from one product to another. In fact, the only noticeable change is likely to be in the name of the reader slot(s) returned by **SCardListReaders**. New reader slots will have 'M519' in the name, instead of 'H663'.

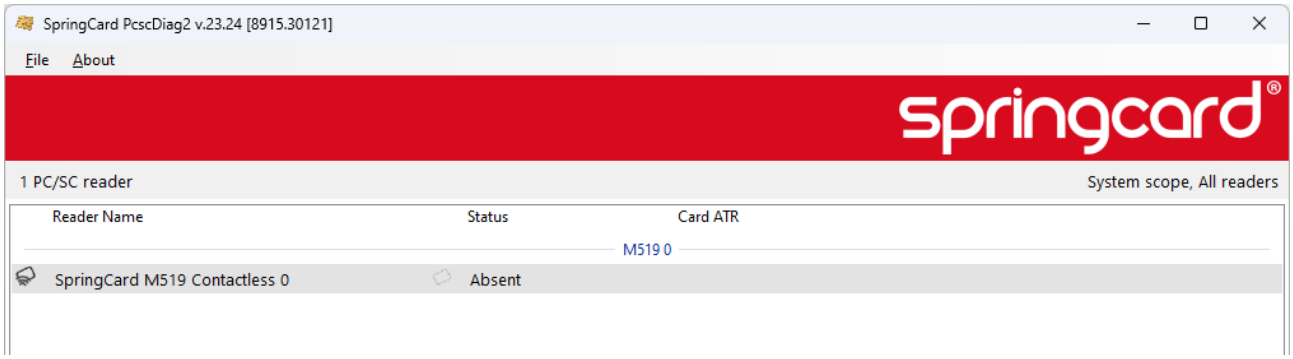

Illustration 8: PcscDiag2 showing the reader name of a M519-SUV in CCID (PC/SC) mode on Windows, using SpringCard driver.

7.2 Coupler: H663/RDR RFID Scanner to M519 RFID Scanner

In RFID Scanner mode, the OEM product conforms to the USB HID (Human Interface Device) standard for keyboards and must be supported by a compatible driver running on the host. All major host operating systems (Windows, Linux, macOS for laptops and desktops, and even Android and iOS for tablets) natively support HID keyboards and thus already include a suitable driver.

There is no need for configuration, as the M519 in RFID Scanner mode exposes the HID keyboard class identifier in its USB descriptor, which is sufficient for the operating system to associate it with the correct driver.

Integrators working with customised, low-end operating systems may still be interested in knowing the product ID:

- ~~**H663/RDR**: `VendorID = 0x1C34, ProductID = 0x9241`~~
- **M519 in RFID Scanner mode**: `VendorID = 0x1C34, ProductID = 0x6213`

Legal Information

DISCLAIMER

This document is provided for informational purposes only and shall not be construed as a commercial offer, a license, an advisory, fiduciary or professional relationship between SPRINGCARD and you. No information provided in this document shall be considered a substitute for your independent investigation. The information provided in the document may be related to products or services that are not available in your country.

This document is provided "as is" and without warranty of any kind to the extent allowed by the applicable law. While SPRINGCARD will use reasonable efforts to provide reliable information, we don't warrant that this document is free of inaccuracies, errors and/or omissions, or that its content is appropriate for your particular use or up to date. SPRINGCARD reserves the right to change the information at any time without notice.

SPRINGCARD doesn't warrant any results derived from the use of the products described in this document. SPRINGCARD will not be liable for any indirect, consequential or incidental damages, including but not limited to lost profits or revenues, business interruption, loss of data arising out of or in connection with the use, inability to use or reliance on any product (either hardware or software) described in this document.

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products may result in personal injury. SPRINGCARD customers using or selling these products for use in such applications do so on their own risk and agree to fully indemnify SPRINGCARD for any damages resulting from such improper use or sale.

INFORMATION ABOUT THE BRAND

SPRINGCARD, the SPRINGCARD logo are registered trademarks of SPRINGCARD SAS. All other brand names, product names, or trademarks belong to their respective holders. Information in this document is subject to change without notice. Reproduction without written permission of SPRINGCARD is forbidden.

COPYRIGHT NOTICE

All information in this document is either public information or is the intellectual property of SPRINGCARD and/or its suppliers or partners.

You are free to view and print this document for your own use only. Those rights granted to you constitute a license and not a transfer of title : you may not remove this copyright notice nor the proprietary notices contained in these documents, and you are not allowed to publish or reproduce this document, either on the web or by any means, without written permission of SPRINGCARD.

EDITOR'S INFORMATION

SPRINGCARD SAS company with a capital of 227 000 €
RCS EVRY B 429 665 482
Parc Gutenberg, 2 voie La Cardon
91120 Palaiseau – FRANCE

CONTACT

For more information and to locate our sales office or distributor in your country or area, please visit **www.springcard.com**