Application Note

Using the M519 in SpringProx Legacy mode

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	PNA23189-AA
Date	10/10/2023
Diffusion	Public
Keywords	M519, SpringProx, Legacy, USB, serial, SDK

Revision History

Version	Date	Author	QC	Description
AA	31/08/2023	JDA		Initial release

Table of Contents

1 Introduction

1.1 Overview

This document describes the principle of using the M519 and its derivatives as a coupler, using the legacy protocol of the SpringProx products (K531, K632, K663, CSB4).

The M519 is an advanced dual-interface OEM module, supporting contactless operation (NFC/RFID HF, ISO/IEC 14443 and 15693) through an external antenna, and, through an optional interface board, contact (ISO/IEC 7816) operation.

The M519 supports various operating modes: Smart Reader, RFID Scanner (keyboard wedge), PC/SC Coupler, SpringProx Legacy, and more. Although the PC/SC Coupler mode is both the most versatile and the most open (interoperable) of all modes, developers and integrators who have been using the earlier SpringProx products may find interesting to prefer the SpringProx Legacy mode.

This document explains how to do so; it is associated with the `springcard-springprox-sdk` project that can be downloaded freely from GitHub:

https://github.com/springcard/springcard-springprox-sdk

The document will guide you through the process of setting up and operating the M519 in SpringProx Legacy mode over either the Serial or the USB interface, compiling the SDK on your target system, and start developing your own solution.

Warning: in SpringProx Legacy mode, the contact (ISO/IEC 7816) smart card interface of the M519 is not available. Only the contactless (NFC / RFID HF) interface is supported in this mode.

1.2 Target platforms — a foreword

It is difficult to present the integration of a product such as the M519 within an embedded target in the general case, as each embedded target has its own specific features (bare-metal microcontroller, cooperative micro-kernel, pre-emptive micro-kernel, use of DMA and/or DPC, etc.).

Since this SDK is dedicated to upgrading existing projects, with developers who already have an experience with SpringProx development, we have chosen to focus this SDK onto the PC world (Linux or Windows). Of course the M519 may easily be used together with virtually any host platform.

1.3 Related Documents

1.3.1 Documents available as PDF

Reference	Title / Description
PFT22217	M519 Data Sheet and Integration Guide
PMD23175	M519-SRK Getting Started Guide
PMI23209	M519-SUV Getting Started Guide

1.3.2 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/

1.4 Related Literature

The present document makes use of concepts and vocabulary that are taken from existing standards and specifications. Please refer to the original documents for a complete understanding.

1.4.1 ISO Standards

1.4.1.1 ISO/IEC 7816

ISO/IEC 7816 is an international standard that defines the characteristics and requirements for integrated circuit cards (ICCs) or smart cards, as well as their interfaces with devices. This standard encompasses physical, electrical, communication, and application-specific aspects of smart cards, providing a comprehensive framework for their design and use. It defines two protocols: T=0 (character) and T=1 (block).

Visit iso.org online store if you want to buy this standard.

1.4.1.2 ISO/IEC 14443 and 15693

ISO/IEC 14443 defines the standards for proximity cards used for identification and contactless communication. ISO/IEC 15693, specifies the requirements for vicinity cards, which have a longer communication range when used with a larger antenna. According to these standard the M519 is both a proximity coupling device (PCD) and a vicinity coupling device (VCD).

Both standards detail parameters like physical characteristics, radio frequency power, and signal interface for contactless ICC.

High-end contactless smart cards are operated at APDU-level over a block protocol ("T=CL") exactly like the T=0 and T=1 contact protocols, where low-end RFID ICs use a proprietary command set (this is for instance the case for the NXP Mifare family) or the low-level command set defined in ISO/IEC 15693.

Visit iso.org online store if you want to buy these standards.

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.
HAL	Hardware Abstraction Layer
PCD	Proximity Coupling Device; i.e. a contactless coupler compliant with ISO/IEC 14443
PICC	Proximity Integrated Circuit Card; i.e. a contactless smart card compliant with ISO/IEC 14443
OSI	Open Systems Interconnection: a conceptual framework used to understand and standardize the functions of telecommunication and computing systems
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
TPDU	Transport-Protocol Datagram Unit. The name comes from the OSI model to describe a packet or frame sent by a device to another.
USB	Universal Serial Bus
VCD	Vicinity Coupling Device; i.e. a contactless coupler compliant with ISO/IEC 15693
VICC	Vicinity Integrated Circuit Card; i.e. an RFID chip compliant with ISO/IEC 15693

2 Hardware setup

2.1 Quick-start for the M519-SRK

The M519-SRK “Starter Kit” is a mother board for the M519 and various peripherals (antenna, smart card interfaces). It is designed to let the implementer/developer evaluate the M519 in various configurations easily.

Warning: setting a wrong hardware configuration or applying a wrong power level is likely to damage the M519 or the host. Please refer to [PMD23175] for details.

2.1.1 Using the M519-SRK through the FTDI USB to Serial bridge

In this setup, the M519 is operated through its Serial interface, yet it is connected to the host through an USB cable. Windows and Linux have a driver for the FTDI USB to Serial bridge that is on the board, that exposes the M519-SRK as a (virtual) communication port. This setup is the closest to the CSB4U.

- Disconnect any existing connection (USB or serial), unplug power jack,
- Set jumper JP4 “RF PWR SETUP” to position “EXT”
- Set jumper JP1 to position “JP4”,
- Set jumper JP5 “V_RF_IN” to position “JP1”,
- Plug a DC 12V power supply into jack “12V PWR”,
- Connect the host using an USB cable to the “USB UART” mini-B connector.

NB: in this configuration, you must provide external power to the SRK through its power jack. The schematics does not derivate any power to the M519 from the USB link arriving at the FTDI chip.

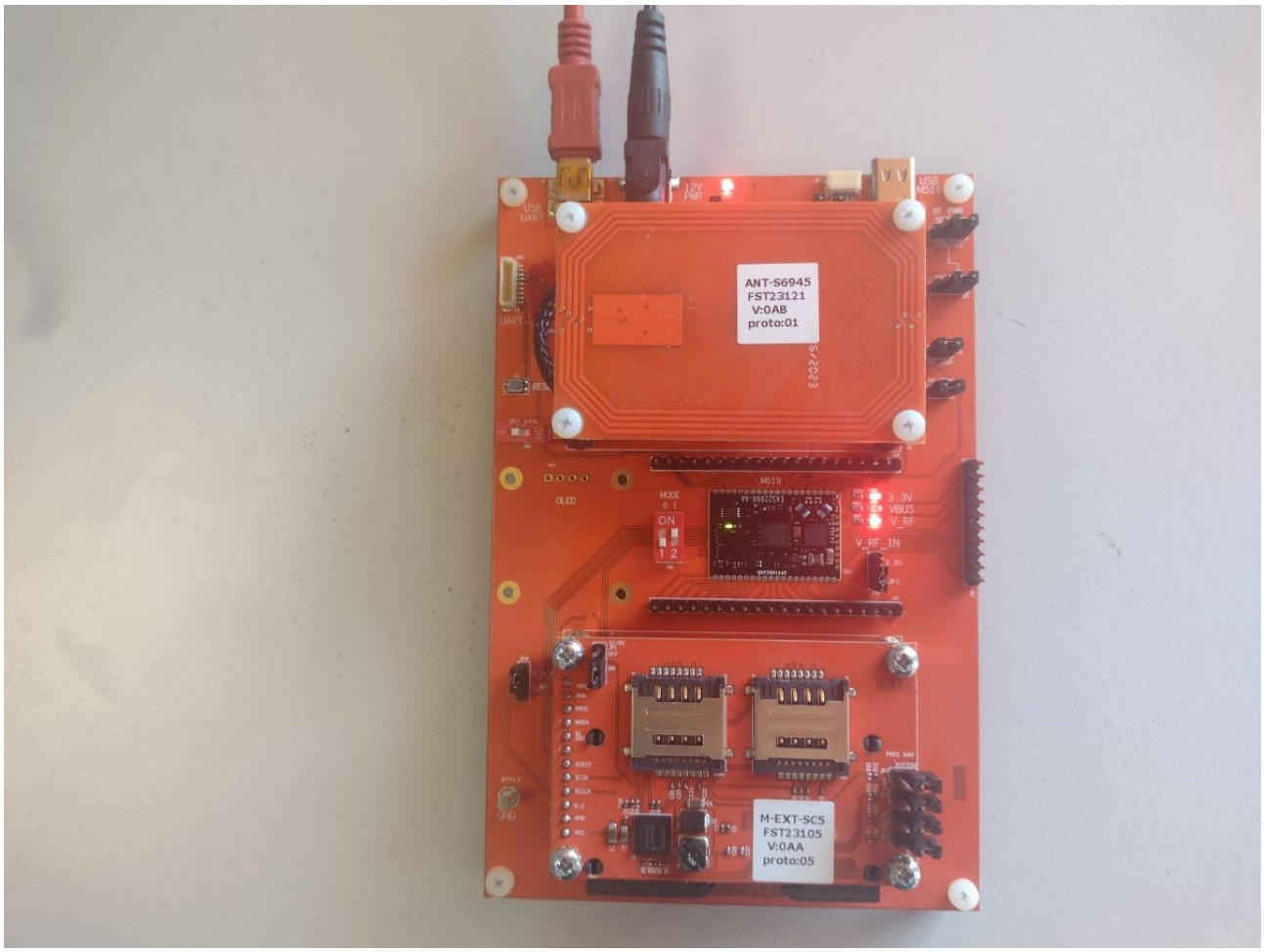

Illustration 1: M519-SRK connected to the PC though its FTDI USB to Serial bridge

2.1.2 Using the M519-SRK through the JST 8-pin connector

In this setup, the M519 is operated through its Serial interface at TTL level. This setup is the closest to the K531-TTL, K632-TTL or K663-TTL.

- Disconnect any existing connection (USB or serial), unplug power jack,
- Set jumper **JP4** “RF PWR SETUP” to position “EXT”
- Set jumper **JP1** to position “JP4”
- Set jumper **JP5** “V_RF_IN” to position “JP1”,
- Connect the host UART (3V level) and the power supply (5V DC) to the JST PH 8-pin connector “UART”.

NB: in this configuration, you may provide external power to the SRK through its power jack if the power source at the JST connector is too weak.

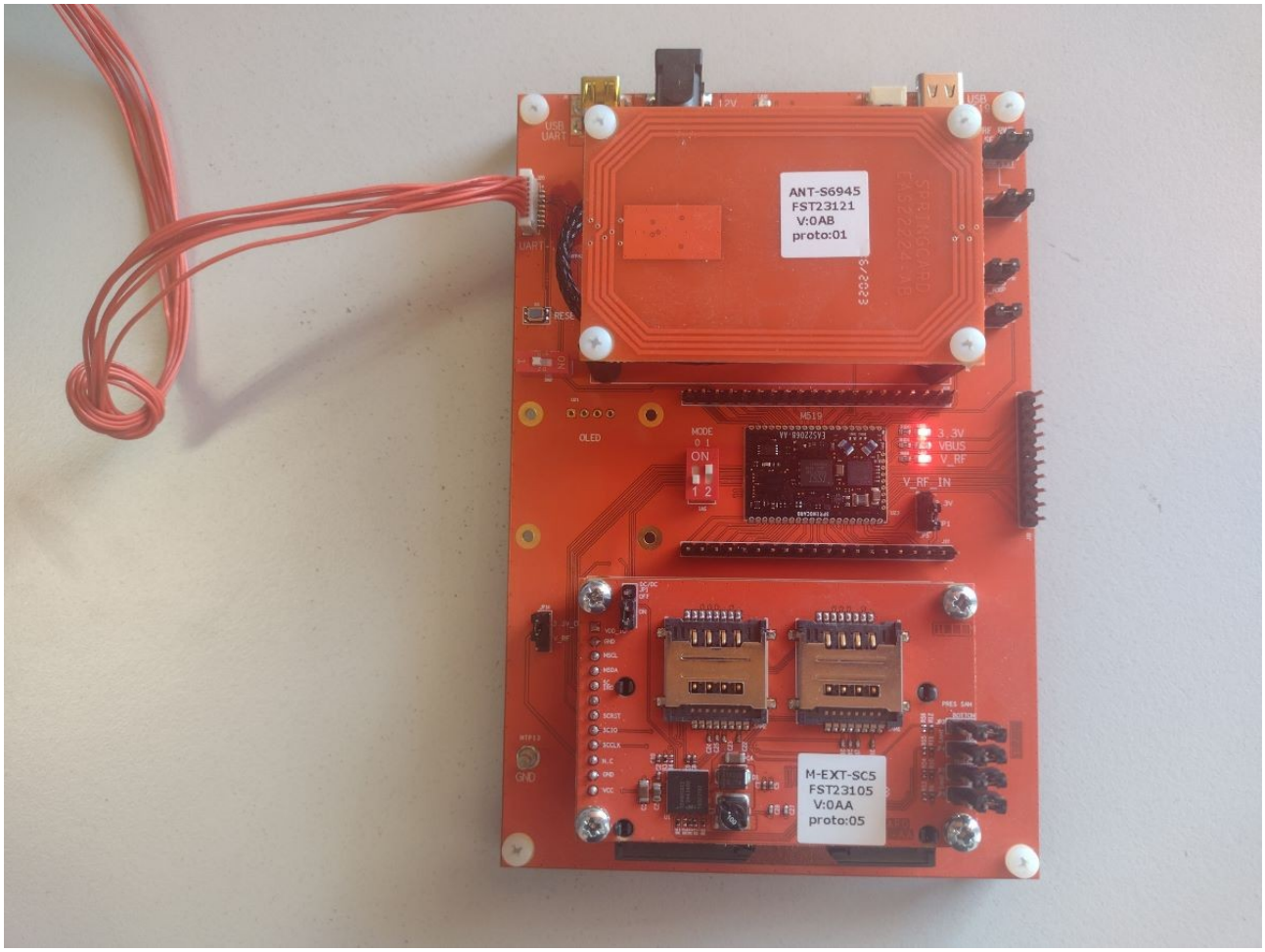

Illustration 2: M519-SRK connected and powered through its JST 8 connector

2.1.3 Using the M519-SRK through the USB interface

In this setup, the M519 is operated through its USB interface and exposes a standard CDC ACM profile (Communication Device Class – Abstract Control Model). Windows and Linux have a driver for this profile, that exposes the M519-SRK as a (virtual) communication port.

- Disconnect any existing connection (USB or serial), unplug power jack,
- Set jumper JP4 “RF PWR SETUP” to position “USB”
- Set jumper JP1 to position “JP4”,
- Set jumper JP5 “V_RF_IN” to position “JP1”,
- Connect the host using an USB cable to the “USB M519” type C connector.

2.2 Quick-start for the M519-SUV

The M519-SUV “Serial & USB Versatile antenna” is a 69x45 antenna that holds the M519. It is designed to be integrated in virtually any equipment and either an USB or a Serial interface. The electrical level of the Serial interface (“RS-TTL”, RS-232 or RS-485) depends on the variant.

Warning: setting a wrong hardware configuration or applying a wrong power level is likely to damage the M519 or the host. Please refer to [PMD23175] for details.

2.2.1 Using the M519-SUV through the Serial JST 8-pin connector

TBD

2.2.2 Using the M519-SUV through the USB JST 5-pin connector

TBD

2.3 Other platforms

This paragraph exposes only the requirements regarding the serial interface itself, the power supply, and the optional handshaking and configuration signals. Of course, an antenna and its matching circuits are necessary for contactless operation, and contact interfaces are necessary to use contact smart cards.

Please refer to [PFT22217] for details.

2.3.1 Serial interface

Using the M519 in PC/SC Coupler over a serial link is possible using only the 2 communication lines (RX, TX) of the serial interface.

On the M519 module itself, the serial interface operates as 0 / 3.3.V. Depending on the hardware behind, your system may communicate with the M519 by the mean of an RS-232, RS-422 or “RS-TTL” (0 / 5V) interface. *RS-485 can not be used with the SpringProx Legacy binary protocol since the protocol requires a full-duplex medium.*

When using the M519-SUV, always check the actual configuration of the antenna, since this single hardware provides the 3 interface levels. Do not connect the USB interface together with the serial interfaces since USB takes priority.

When using the M519-SRK, you may connect to the M519 using either

- the USB to Serial bridge “USB UART” (USB mini-B connector, FTDI chipset),
- the JST PH 8 connector “UART” that supports either 0 / 3.3V or 0 / 5V levels,
- the header that is directly connected to the module’s pin (0 / 3.3V only in this case).

Connect only one interface at once. Do not connect the main USB interface (USB type C connector, label “USB M519”) together with the serial interfaces since USB takes priority.

Always report to the detailed documentation of the actual hardware involved to avoid damaging the M519 or the host system.

2.3.2 Power supply

The serial interface is available only when the M519 is powered at 3.3V (5V power is associated with USB).

Unless you are using a complete product that derivates a 3.3V rail from over power source, respect the following setup:

- Connect both VIN_3V3 (pin 40) and VIN_RF (pin 39) to a 3.3V power supply that can deliver at least 500mA,
- Connect both GND signals (pins 37 and 20) to the ground of the power supply.
- Leave VBUS (pin 38) unconnected.

2.3.3 Handshaking signals

- /WAKEUP (pin 16) and /SUSPEND (pin 17)

The /WAKEUP and /SUSPEND signals of the M519 are not used in SpringProx Legacy mode. Leave this signals unconnected.

- /RESET (pin 19)

This signal may be driven by the host to reset the M519. It could be left unconnected as well.

Start-up of the M519 may take up to 500ms (more if a firmware upgrade takes place). The host knows that the M519 is ready when it sends the “M519” string on its Serial line.

2.3.4 Optional configuration signals

MODE0 and MODE1 shall be HIGH level or left unconnected to use the M519 in SpringProx Legacy mode.

3 Configuring the M519 for SpringProx Legacy operation

M519 OEM module:

The M519 is delivered with a default configuration that enables only the SpringCard Direct protocol. The SpringProx Legacy protocol must be explicitly selected before using the M519 in this mode.

There are some exceptions to the fact that the M519 is delivered with only the SpringCard direct protocol enabled. For instance, sample products may have been pre-configured by SpringCard FAE / Support team for the ease of a particular demonstration. Also custom order codes may be placed for large batches of pre-configured modules.

Anyway, generally speaking, the integrator shall always plan for and execute a configuration step for the M519 module while assembling the final system.

To configure the M519 for SpringProx Legacy operation, value $_H$01 shall be written in register $_H$02C0 in the non-volatile memory of the module.

Products derived from the M519:

Products like the M519-SUV are generally delivered with a specific factory configuration. Consult SpringCard Sales team to select the appropriate order code to get M519-SUV pre-configured for SpringProx Legacy operation. Pay attention that writing a new configuration, as explain below, will overwrite the factory configuration.

3.1 Configuring the M519's non-volatile memory

The simplest procedure uses a terminal software and human interaction with the M519. It should work whatever the current configuration of the M519 is, since the shell ("human console") is available in all modes.

- Connect to the M519 over a serial interface. Refer to next paragraph "Testing the connection" for the procedure and parameters,
- Enter command `cfgC0` to read the current value of register $_H$02C0,

- Enter command `cfgC0=01` to activate the SpringProx Legacy protocol,
- Enter command `reset` to activate the new configuration.

If the M519 is in its default configuration (SpringCard Direct protocol), the procedure may be completely automated using SpringCoreConfig command line tool, like this:

```
SpringCoreConfig --serial=COM5 --write 02C0=01
```

(Replace COM5 with the actual communication port on your production/test system).

The drawback is that this procedure with fail if the device has already been configured (it works only when the Direct protocol is selected).

3.2 Testing the connection

3.2.1 Linux

- Download and install a terminal software for serial communication.

For the snapshots, we are using *Minicom* (on Ubuntu/Debian, use `sudo apt-get install minicom` to install it).

- Open a the serial port the M519 is connected to. Configuration is:
  - Baudrate: 38400bps,
  - Format: 8 data bits, 1 stop bit, no parity, no flow control.

The serial port shown in the examples is /dev/ttyS5. Adjust to our actual setup.

- Reset the M519,
- Wait at least 250ms to let the bootloader execute,
- Send *<CR><LF>*,
- The M519 echoes the new line sequence and replies

  `SpringCard M519 v1.26 Legacy Serial`*<CR><LF>*
  `>` *<CR><LF>*

Congratulation! This sequence shows that the M519 is correctly configured and connected to your Linux computer. Don't forget to disconnect the terminal software from the serial port before trying to connect to the M519 from another software.

3.2.2 Windows

- Download and install a terminal software for serial communication.

For the snapshots, we are using *Hterm*, a free software available at

https://www.der-hammer.info/pages/terminal.html

- Open a the serial port the M519 is connected to. Configuration is:
  - Baudrate: 38400bps,
  - Format: 8 data bits, 1 stop bit, no parity, no flow control.

The serial port shown in the examples is COM5. Adjust to our actual setup.

- Adjust the configuration of the terminal software so that
  - Terminal sends <*CR*><*LF*> when you hit the <*RETURN*> key,
  - Terminal use <*CR*><*LF*> as new line separator.
- Reset the M519,
- Wait at least 250ms to let the bootloader execute,
- Send <*CR*><*LF*>,
- The M519 echoes the new line sequence and replies

```
SpringCard M519 v1.26 Legacy Serial<CR><LF>
> <CR><LF>
```

"v1.26" is the version number of the firmware. Newer firmwares will show another version number.

"PC/SC" is the operating mode. If your M519 answers with another operating mode, go back to previous paragraph "Configuring the M519 for PC/SC Coupler operation".

“Serial” is the primary host interface. If your M519 answers “USB”, not “Serial”, go back to step “Connecting the M519 to your target”.

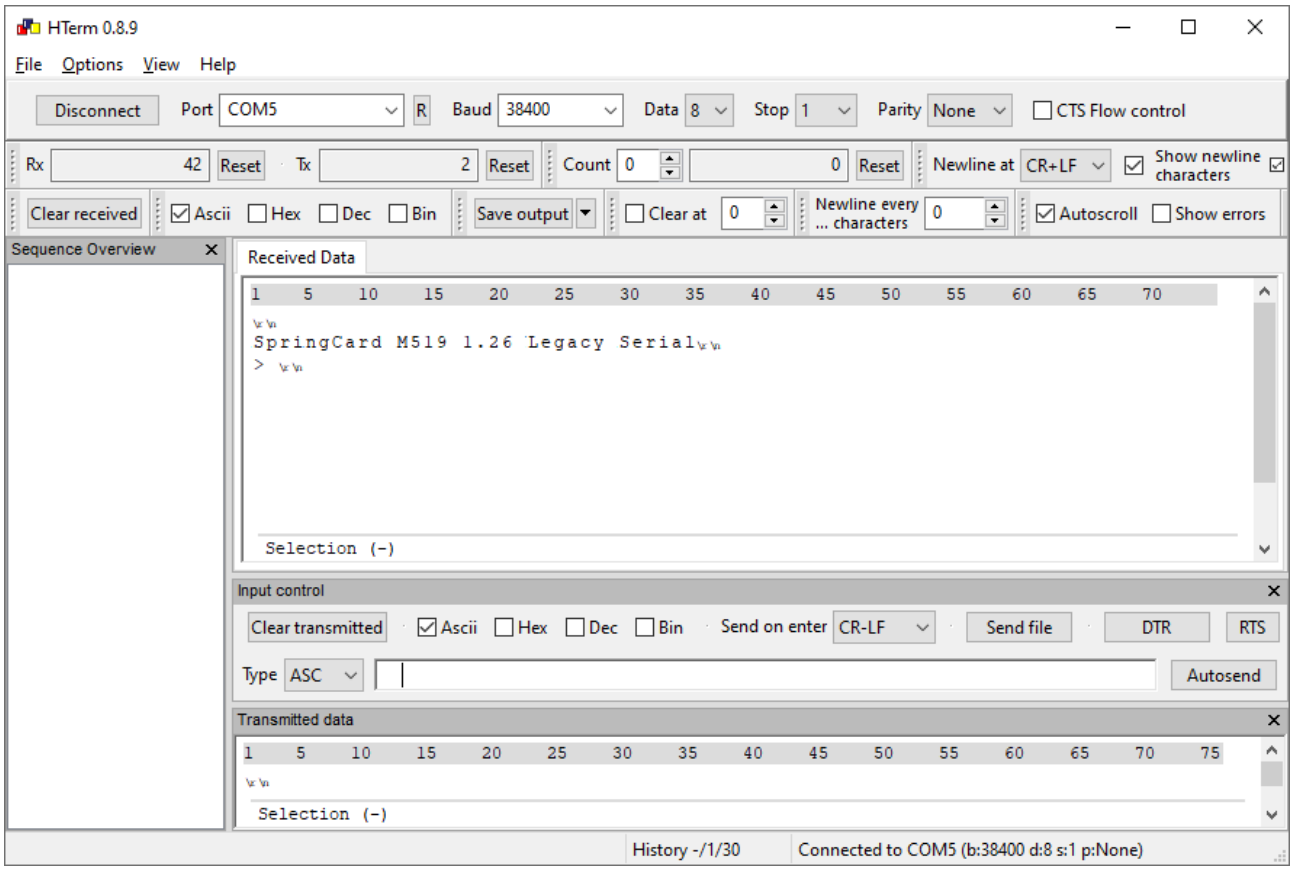

Illustration 3: retrieving the prompt of the M519 with HTerm

Congratulation! This sequence shows that the M519 is correctly configured and connected to your Windows computer. Don’t forget to disconnect the terminal software from the serial port before trying to connect to the M519 from another software.

3.3 Reset message

Toggle the /RESET signal (or press and release the reset button on the M519-SRK) to reset the M519.

When the firmware has finished its startup sequence, the M519 sends its startup string “M519”.

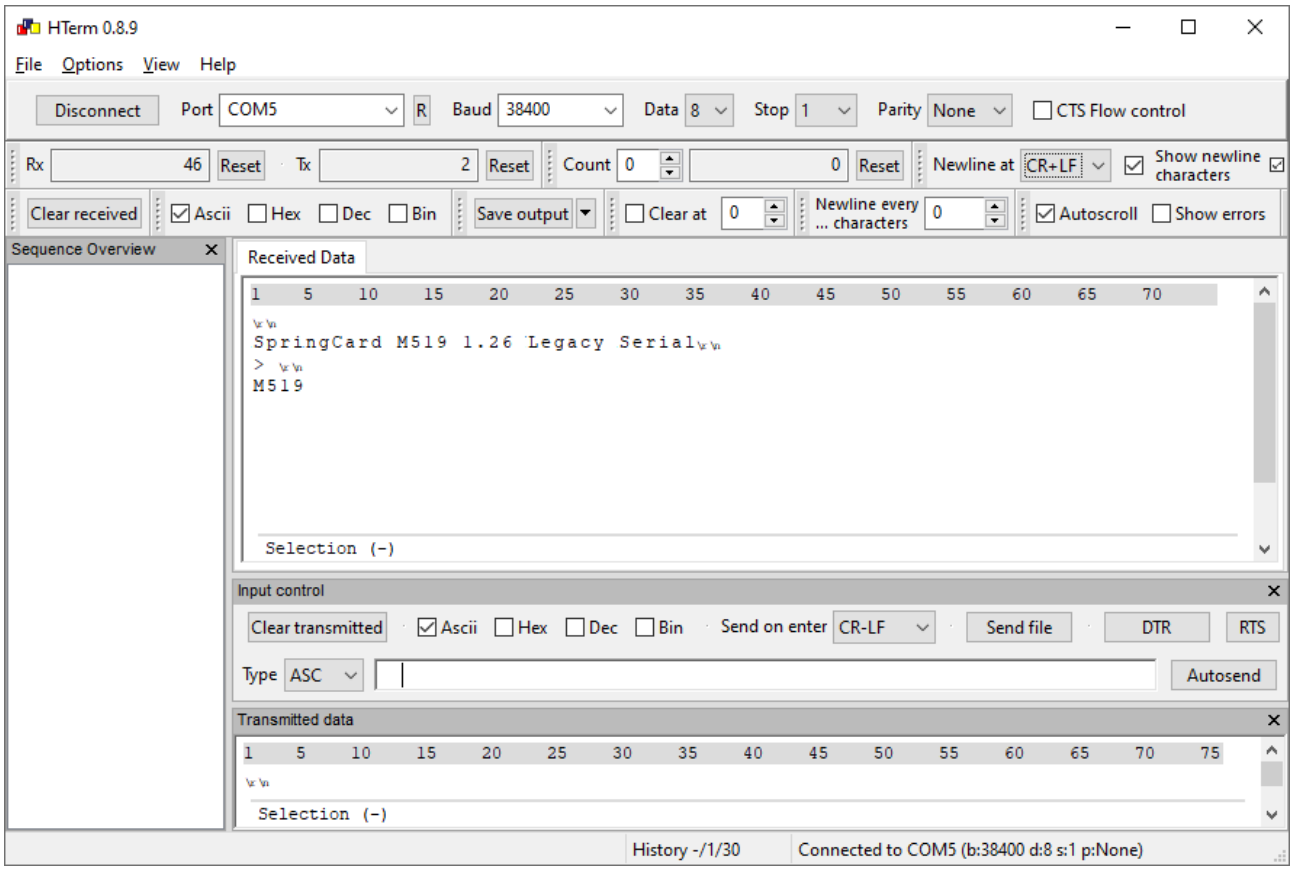

Illustration 4: the startup string

4 Using the M519 with SpringProx Library 1.80

Most existing applications using a SpringProx product (K531, K632, K663 or CSB4) or based on the SpringProx Library (`springprox.dll` on Windows, `springprox.h` header file when developing in the C language).

Prior to the release of the SDK described in chapter 5 , the last version of the Library was 1.80, dated 2014. This version of the Library could be used with the M519, provided that a few precautions are taken, that are described in this chapter.

Using the M519 with a version of the SpringProx Library that is older than 1.80 is not possible.

4.1 Particularities of the SpringProx implementation in M519

4.1.1 “M519” startup string

The Library detects fatal hardware or protocol errors that make the device reset by recognizing its startup string. Supported startup strings are “CSB[?]”, “[?]531”, “[?]632”, “[?]663”, “[?]601”.

Since the M519 sends “M519” as startup string, the Library will not understand that the module resets and will not reconnect automatically and silently in this situation. Should this happened, the host application must be adapted, to implement an explicit disconnect/reconnect sequence every time the Library reports a fatal communication error (fortunately this never happens in stable, validated implementations).

4.1.2 Fake version number

The SpringProx Library has the knowledge of all generations of SpringProx device, and “knows” which features are supported by every version of SpringProx firmware. Supported versions of SpringProx firmware go from 1.40 (2006) to 2.xx (2015 and onwards).

Since M519 is based on the new SpringCore firmware, a new version numbering sequence has been started (at the date of writing, the version of M519 is 1.26).

To ensure that the Library supports the M519 as a latest-generation SpringProx device, the M519 returns “2.99” when its version number is queried over the SpringProx protocol (SPROX_ReaderGetFirmware function), instead of its actual (SpringCore) version number.

The application must be aware of this fact.

4.1.3 Binary and ASCII protocols only

The SpringProx Library supports 3 protocols:

- OSI 3964, a half-duplex protocol used by the CSB3 generation,
- “SpringProx Binary”, a full-duplex protocol developed in 2003,
- “SpringProx Bus”, a half-duplex variation of the later, introduced in 2004 for RS-485 links,
- “SpringProx ASCII”, a “human readable” protocol, introduced in 2004 to help using the devices from a script language.

At the difference with the earlier generations of devices, the M519 does support only the Binary and the ASCII protocols. OSI 3964 and Bus protocols are not implemented.

This has normally no impact on the application, unless the application tries to enforce the protocol. In this case, the call to SPROX_ReaderSetDeviceSettings must be removed.

4.2 Features announced by the M519

The SPROX_ReaderGetFeatures function of the Library returns a DWORD value that tells which features are supported by the device. The corresponding bit mask is documented in `sprox_capas.h`.

M519 feature bit map is `01076303`. This is understood as follow (LSB to MSB, i.e. right to left):

Bit	Value	Description	Remark
# 0	1	ISO/IEC 14443 supported	
# 1	1	ISO/IEC 15693 supported	
# 2	0	Felica not supported	Actually the M519 does support Felica / NFC-F, but with a different interface than the one expected by SpringProx Library
# 3	0	ISO/IEC 18092 not supported	Actually the M519 does support ISO/IEC 18092 / NFC peer-to-peer, but with a different interface than the one expected by SpringProx Library
# 4	0	No NFC target mode	Actually the M519 does support NFC target mode, but with a different interface than the one expected by SpringProx Library
# 5	0	No contact smart card	Actually the M519 may have a contact smart card interface (ISO/IEC 7816), but it is not made available through the SpringProx Legacy protocol
# 6	0	No 125kHz reader	
# 7	0	No card emulation	Actually the M519 does support NFC card emulation, but with a different interface than the one expected by SpringProx Library
# 8	1	ASCII protocol	
# 9	1	Binary protocol	
# 10	0	No RS-485 interface	
# 11	0	No Bus protocol	
# 12	0	No RS-485 driver	
# 13	1	Separated RX/TX buffers	
# 14	1	115200bps supported	
# 15	0	Buffers are limited to 260B	
# 16	1	USB VCP profile	(virtual comm port, aka CDC ACM profile)

# 17	0	CCID profile	Used in PC/SC Coupler mode
# 18	0	HID keyboard profile	Used in RFID Scanner mode
# 19	0	*RFU*	
# 20	0	Single antenna	
# 21	0	No I^2C	Actually the M519 has an I^2C bus, but with a different interface than the one expected by SpringProx Library
# 22	0	No non-volatile memory	Actually the M519 has a non-volatile memory, but with a different interface than the one expected by SpringProx Library
# 23	0	No mass storage	
# 24	1	Shell (human console)	
# 25	0	*RFU*	
# 26	0	*RFU*	
# 27	0	*RFU*	
# 28	0	*RFU*	
# 29	0	*RFU*	
# 30	0	*RFU*	
# 31	0	*RFU*	

4.3 Particularities of the NFC / RFID HF implementation in the M519

4.3.1 PN5190 fronted

The NFC / RFID HF interface of the M519 is based on an NXP PN5190 chipset, where earlier devices used either a MfRC500 (CSB3), a MfRC531 (K531), a CLRC632 (K632) or a CLRC663 (K663). The new chipset has a totally different API. It totally impossible for the host application to dynamically access the register of the PN5190, as it was partially possible with the other chipsets.

Therefore, the host application shall not use the following functions:

- ReadRIC
- WriteRIC

- SetRCBitMask
- ClearRCBitMask
- ExchangeByteStream
- All functions named Pcd... (PcdSetTmo, PcdReadE2, etc),
- All functions named Mf500... (Mf500PcdConfig, Mf500PiccRequest, etc).

4.3.2 Mifare Classic keys

At the difference with the MfRC500 (CSB3), a MfRC531 (K531), a CLRC632 (K632) and the CLRC663 (K663), the PN5190 that is in the M519 has E2PROM with a secure storage area for the Mifare Classic (CRYPTO1) keys.

This feature is emulated by the SpringCore firmware; the keys that should be written into the E2PROM of the chipset with SPROX_MifLoadKey are actually written into the flash memory of the M519.

Thanks to this emulation, there is no impact on the application, but the system designer must be warned that the Mifare Classic keys are now stored in an unsecure storage area.

4.3.3 ICODE1

At the difference with the K632, the M519 (as well as the K663) does not support NXP the proprietary modulation of early NXP ICODE1 chip (but the ISO/IEC 15693 modulation of today's NXP ICODE-SLI chips is supported).

4.3.4 Timings

Due to the difference both in the NFC / RFID HF interface and in the implementation, the M519 is generally a little faster than a SpringProx device, but may be slower is some situations.

5 Getting started with the SDK

5.1 Downloading the SpringProx Legacy SDK from GitHub

The SDK that accompanies this application note is named `springcard-springprox-sdk` and is hosted on GitHub. Direct link is:

https://github.com/springcard/springcard-springprox-sdk

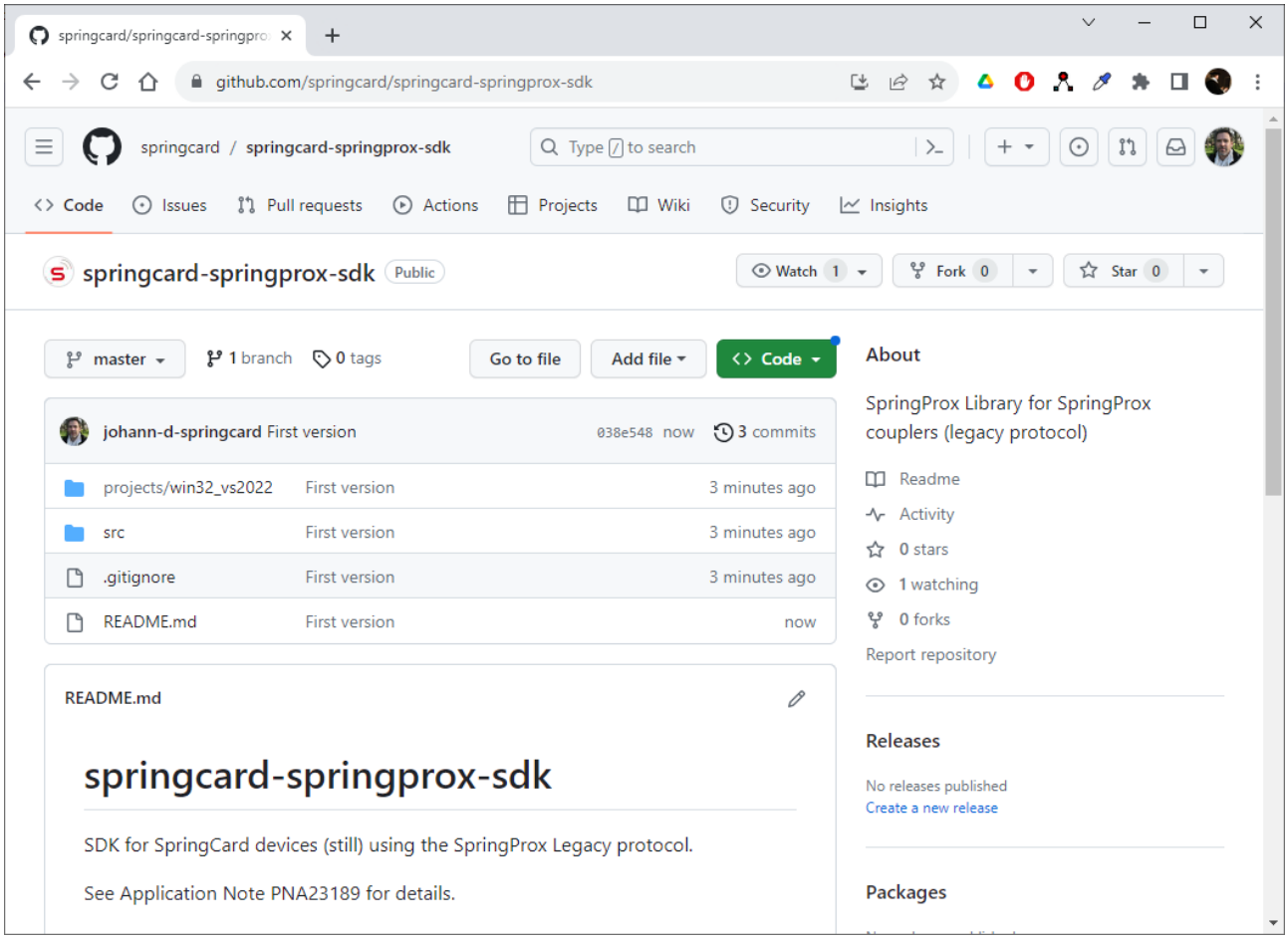

Illustration 5: springcard-springprox-sdk project on GitHub

If you have Git installed, you may clone it directly

```
git clone https://github.com/springcard/springcard-springprox-sdk.git
```

Then

```
cd springcard-springprox-sdk
```

to enter the directory.

5.2 Exploring the SDK files and projects

The organisation of the SDK tree is pretty straightforward:

Directory	Content
/ projects	
/ linux	Projects for Linux (GCC + GNU Make)
/ win32_mingw	Projects for Windows (GCC + GNU Make under MinGW)
/ win32_vs2022	Projects for Windows (Visual Studio 2022)
/ src	
/ common	
/ cardware	
/ calypso	Helper library for Calypso cards *(unmaintained)*
/ desfire	Helper library for NXP Desfire *(unmaintained)*
/ mifplus	Helper library for NXP Mifare Plus *(unmaintained)*
/ mifulc	Helper library for NXP Mifare UltraLight C *(unmaintained)*
/ products	
/ springprox	
/ api	SpringProx Library
/ samples	Source code of the sample applications

Notice for unmaintained card helper libraries: In the past, SpringCard used to provide free libraries to help working with a few cards. Given the major and frequent changes made by card issuers, given that the source code developed by SpringCard was being plundered and used with non-SpringCard products, all development on the card libraries in C language has ceased (some C# counterpart are still being developed for PC/SC

Couplers). Don't hesitate to contact the card issuers to get support and source code to accompany their products. SpringCard may also offer custom development services.

5.3 Building the libraries and the examples

5.3.1 Linux

You must have GCC and GNU make installed on your machine. There are plenty of tutorials explaining how to do so on Linux.

To build the project,

- Enter the directory of the project

```
cd projects/linux
```

- Run `make` to build everything (the SpringProx library, then the helper libraries, then the samples)

```
make
```

When running on the the samples, specify the communication device after the **`-d`** command line flag, e.g.

```
bin/ref_info -d /dev/ttyS5
```

5.3.2 Windows, using MinGW and GCC

You must have MinGW, GCC and GNU make installed on your machine. There are plenty of tutorials explaining how to do so on Windows.

To build the project,

- Enter the directory of the project

```
cd projects/win32_mingw
```

- Run `make` to build everything (the SpringProx library, then the helper libraries, then the samples)

```
make
```

When running the sample, specify the communication device after the `-d` command line flag, e.g.

```
bin/ref_info.exe -d COM5
```

5.3.3 Windows, using Visual Studio

To build this sample, you must have Microsoft Visual Studio 2022 and its C/C++ compiler installed on your machine.

- Open the `projects/win32_vs2022/springprox.sln` solution that contains the project.

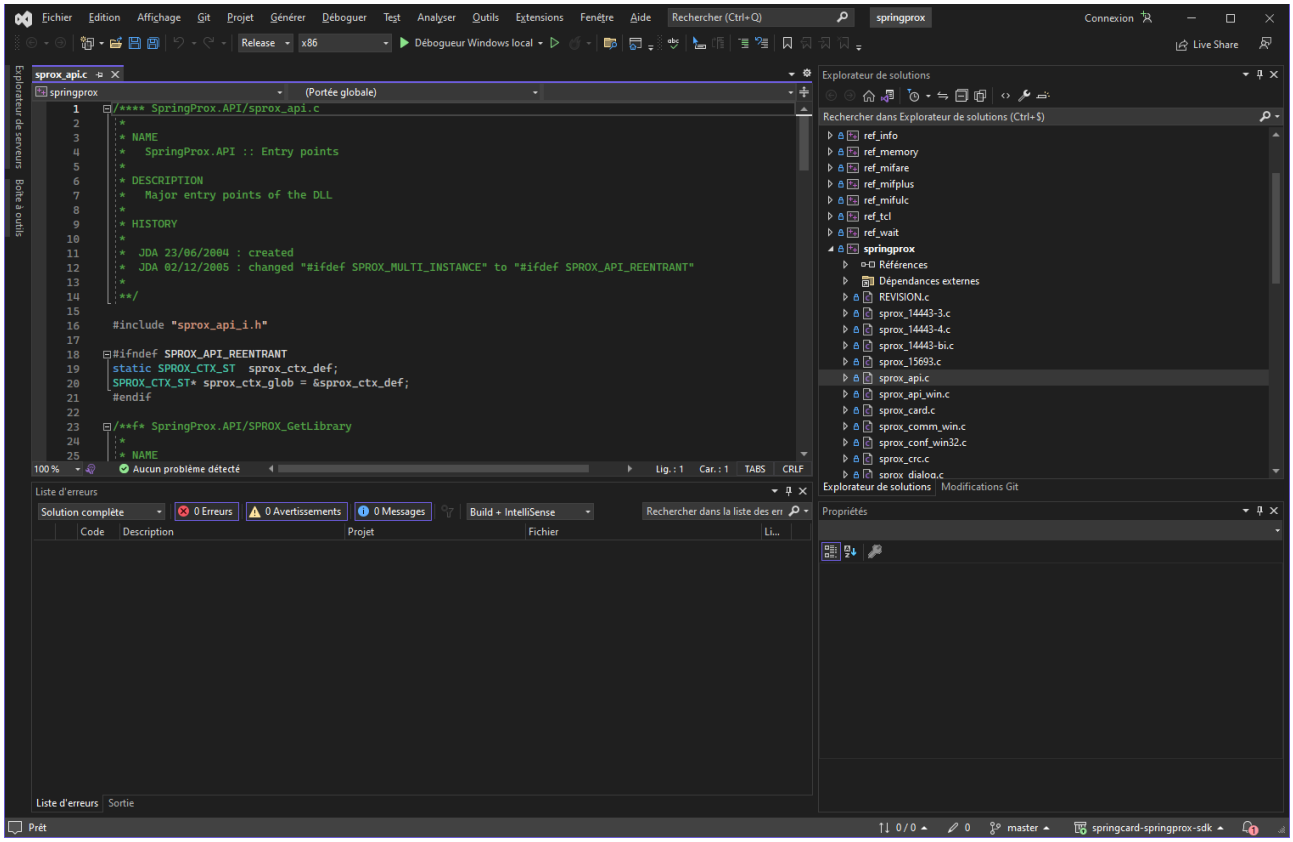

Illustration 6: The springprox solution and its projects in Visual Studio 2022

- Select the target configuration you want to build: Debug or Release, x86 or x64,
- Build the solution.

When running the sample, specify the communication device after the `-d` command line flag.

```
ref_info.exe -d COM5
```

5.4 The sample applications at a glance

5.4.1 ref_info

This is a minimal sample to test the communication between the PC and the M519. It shows the features announced by the device (see § 4.2).

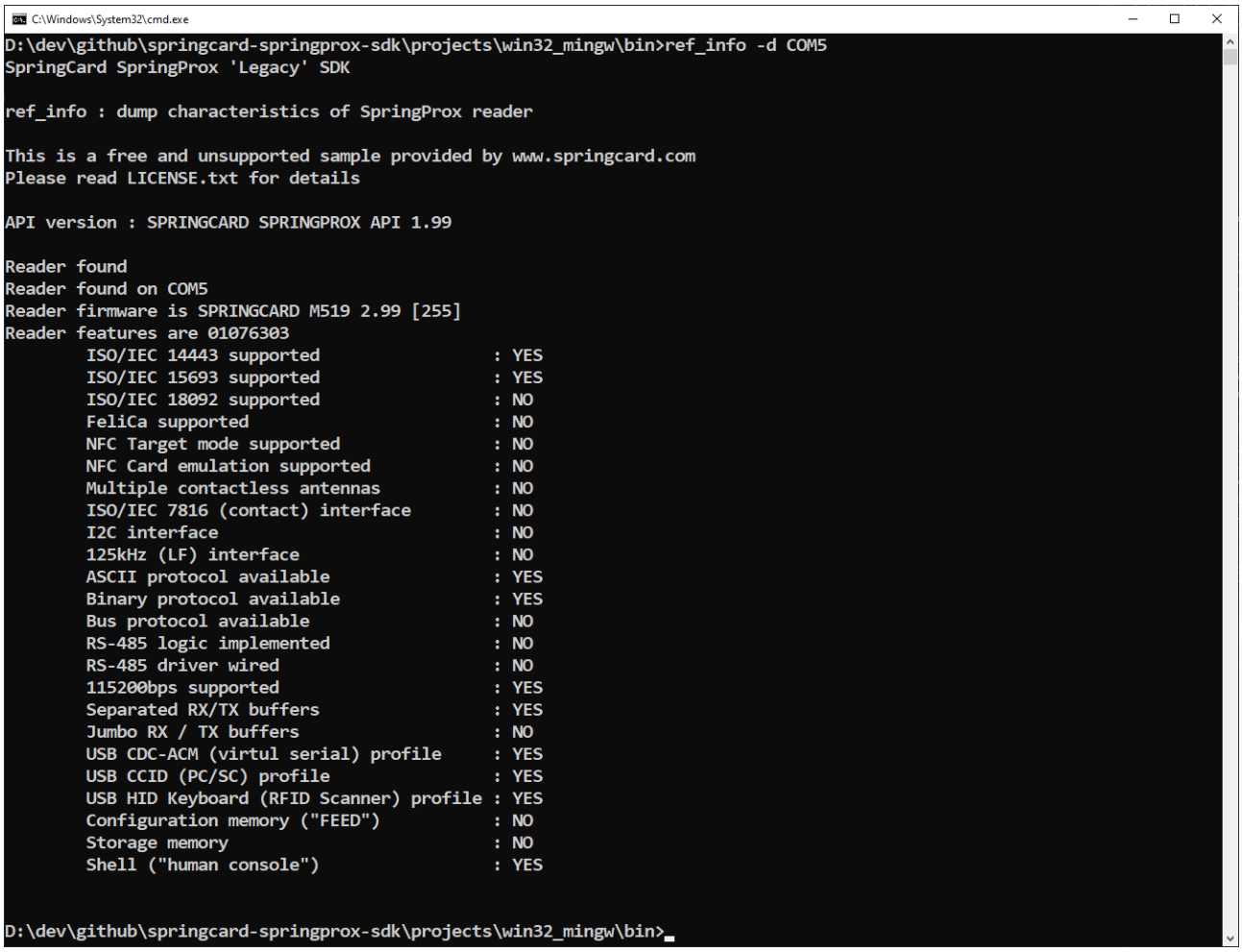

Illustration 7: Output of ref_info

5.4.2 ref_find

This sample shows how to use the SPROX_Find function to look for a contactless card in the RF field

5.4.3 ref_wait

This sample shows how to use the SPROX_Wait function. This is the blocking alternative to SPROX_Find.

5.4.4 ref_tcl

This sample shows how to activate and use the ISO/IEC 14443-4 transport protocol ("T=CL"):

- SPROX_TclA_GetAts, SPROX_TclA_Pps to activate a PICC that is ISO/IEC 14443 type A,
- SPROX_TclB_AttribEx to activate a PICC that is ISO/IEC 14443 type B,
- SPROX_Tcl_Exchange to exchanges APDUs with the card (whatever its type).

5.4.5 ref_memory

This sample shows how to read

- ISO/IEC 153693 cards (also NFC Forum Type 5 Tags),
- NXP Mifare UltraLight (or NTAG) cards (also NFC Forum Type 2 Tags),

5.4.6 ref_mifare

This sample shows how to uses NXP Mifare Classic cards.

Warning: running this sample will overwrite or erase any previous card content.

5.4.7 ref_mifplus

This sample shows how to format, read and write NXP Mifare Plus cards. It is associated with the Mifare Plus helper library (`sprox_mifplus.dll`).

Warning: running this sample will overwrite or erase any previous card content.

5.4.8 ref_mifulc

This sample shows how to get authenticated on NXP Mifare UltraLight C cards *(note the 'C' – this is not Mifare UltraLight)*. It is associated with the Mifare UltraLight C helper library (`sprox_mifulc.dll`).

Warning: running this sample will overwrite or erase any previous card content.

5.4.9 ref_desfire

This sample shows how to format, read and write NXP Desfire EV0 cards. It is associated with the Desfire helper library (`sprox_desfire.dll`).

Warning: running this sample will overwrite or erase any previous card content.

5.4.10 ref_desfire_ev1

This sample shows how to format, read and write NXP Desfire EV1 cards. It is associated with the Desfire helper library (`sprox_desfire.dll`).

Warning: running this sample will overwrite or erase any previous card content.

5.4.11 ref_desfire_benchmark

This sample is used to benchmark the library + M519 setup, together with a NXP Desfire EV1 card. It is associated with the Desfire helper library (`sprox_desfire.dll`).

Warning: running this sample will overwrite or erase any previous card content.

5.4.12 ref_calypso

This sample shows how to read Calypso cards. It is associated with the Calypso helper library (`sprox_calypso.dll`).

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**