Subject: TripTick Product Manual

Revision: 1.4

Issue Date: 16.02.2021

Product names mentioned herein are for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

TripTick®

2D barcode/NFC reader with EMV Level 2 capability

Product Manual

Access-IS
18 Suttons Business Park, Reading
Berkshire, RG6 1AZ, United Kingdom
Tel: +44 (0) 118 966 3333
Web: www.access-is.com
Email: support@access-is.com

Warnings

This manual contains important information regarding the installation and operation of the TripTick 2D barcode/NFC/EMV reader (ATR200, ATR210, ATR220). For safe and reliable operation of the imager, installers must ensure that they are familiar with, and fully understand, all instructions contained herein.

Warranty

Access Ltd warrants that this product shall be free from defects in workmanship and materials for a period of one year from the date of original purchase. If the product should fail to operate correctly in normal use during the warranty period, Access will replace or repair it free of charge. No liability can be accepted for damage due to misuse or circumstances outside Access' control. Access will not be responsible for any loss, damage or injury arising directly or indirectly from the use of this product. Access' total liability under the terms of this warranty shall in all circumstances be limited to the replacement value of this product.

Radio Frequency Energy

European EMC directive 89/336/EEC

This equipment has been tested and found to comply with the limits for a class A computing device in accordance with the specifications in the European standard EN 55022. These limits are designed to provide reasonable protection against harmful interference. This equipment generates, uses and can radiate radio frequency energy and if not installed and used in accordance with the instructions may cause harmful interference to radio or television reception. However, there is no guarantee that harmful interference will not occur in a particular installation.

If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment on and off, the user is encouraged to correct the interference with one or more of the following measures: (a) Reorient or relocate the receiving antenna. (b) Increase the separation between the equipment and the receiver. (c) Connect the equipment to an outlet on a circuit different from that to which the receiver is connected. (d) Consult the supplier or an experienced radio / TV technician for help.

FCC Compliance Statement (United States)

This equipment generates, uses and can radiate radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio communication. It has been tested and found to comply with the limits for a class A computing device in accordance with the specifications in Subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against such interference when the equipment is operated in a commercial environment. Operation of this equipment in a residential area may cause interference, in which case the user at his own expense will be required to take whatever measures may be necessary to correct the interference. Changes or modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment.

Canadian Department of Communications RFI statement

This equipment does not exceed the class A limits for radio noise emissions from digital apparatus set out in the radio interference regulations of the Canadian Department of Communications.

Le présent appareil numérique n'émet pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de la classe A prescrites dans le règlement sur le brouillage radioélectriques publié par le ministère des Communications du Canada.

Contents

1. Overview

The Access-IS TripTick® (ATR200, ATR210, ATR220) reads 2D barcodes from mobile devices and paper, and optionally reads and writes to contactless smart cards, NFC labels and NFC-enabled mobile devices. As an option, TripTick is available with EMV Level 2 certification for use with applications that require contactless payments.

The device has a small-footprint and its low profile, slot-in design enables fast, easy integration into third-party public access kiosks and gates. Its rugged, water-resistant construction with no moving parts, enables TripTick to withstand years of indoor and outdoor public access use.

TripTick's unique design, with optimised imaging ensures read reliability and high-speed performance, capturing and decoding all popular linear, PDF417 and 2D symbologies, in less than half a second.

Figure 1: TripTick 2D barcode and optional NFC reader with EMV Level 2 capability

Optional near-field communication (NFC)/radio-frequency identification (RFID) provides contactless reading capability of all popular contactless cards.

TripTick can also be specified with Europay, Mastercard and Visa (EMV) Level 2 capability and Payment Card Industry (PCI) secure reading and exchange of data (SRED) compliance for contactless payments using VISA, Mastercard, AMEX and Discover schemes.

- Front-face sealed for integration into indoor or outdoor kiosks, podiums and gates.
- UltraGlass option for extreme environments.
- Reads both 2D barcodes and NFC cards, labels and devices from one point-of-presentation.
- Optional EMV Level 2 capability.
- Unique, optimised focal distance design improves read performance for all paper and electronic media.
- Intuitive operation with green and red LED lights to indicate good and bad reads.
- Reads on face-down presentation of 2D, PDF417 and linear barcodes.
- Single interchangeable cable connection to the host PC.
- An option for Power over Ethernet (PoE) allows installation without a dedicated power supply and at a distance of up to 100 m from the host PC.
- RS232, RS485 and USB (serial or keyboard) interface options.
- Interfaces as USB composite or serial device, supporting both NFC and barcode readers.

Applications include:

- Integration into public-use kiosks and gates.
- Travel and transportation mobile-ticket reading.
- Retail voucher redemption and loyalty cards.
- Car park ticketing, turnstiles and automatic gates.

Product options

Flush bezel

The flush bezel option means that when installed, the reader is completely flush with the surface around it, allowing a smooth and fluid movement across the reader regardless of the media. Front-face sealed (IP67) for integration into indoor or outdoor kiosks, ticket machines, podiums, turnstiles and gates.

Raised bezel

A raised bezel can be specified providing a 2 mm raised lip above the glass. Front-face sealed (IP67) for integration into indoor or outdoor kiosks, ticket machines, podiums, turnstiles and gates, the raised lip can assist by directing users to the active area of the reader.

UltraGlass

For environments with highly abrasive elements such as sand and grit, an innovative UltraGlass option is available that delivers significantly enhanced scratch-resistance. This upgrade helps alleviate the possibility of scratching which may, in severe circumstances, affect the barcode reading performance. This option also allows the use of tools such as metal scrapers to remove paints or adhesives from the window during maintenance or repair.

SAM (Secure Access Modules)

The option of installing up to four secure access modules within the device is available.

Architecture

TripTick consists of a ticket reader, a logic unit (running Linux software), a power-conditioning unit, LEDs, and an optional network port for network or internet connectivity.

The ticket reader provides a single point of presentation for tickets and travel passes – whether presented on a card or mobile devices. The reader contains its own self-contained firmware. Data from the reader is sent over USB to the logic unit. This is an open architecture Linux board running third-party software.

The software consists of:

- a bootloader,
- a Linux kernel including peripheral drivers,
- a Linux root file system,
- application software.

Functions

1. The passenger presents a ticket to the TripTick reader. The ticket can be a barcode or RFID/NFC token. If a barcode, it can be a printed barcode, or one displayed on a mobile device.

   Ticket formats supported are listed in the ***Specifications*** (on page 9).

2. The data is captured by the reader and decoded.

   The decoded information is sent via USB for processing by the logic unit. This information is then used to validate the ticket. The ticket can be validated locally in the logic unit (using information obtained from the image), or over an internet connection to a central computer.

3. Information as to whether the ticket has been successfully validated or not is presented to the passenger via the LEDs, typically green for a valid ticket and red for an invalid ticket.

Execution

TripTick will take typically 90s to boot.

This is the time from power-up to being able to scan barcodes.

During this time, the bootloader is loaded into RAM and run, then the Linux kernel is loaded and run, then the application code is loaded and run.

At the start of the application code running, TripTick will perform a power-on self-check determined by the third-party software.

Once the application code has finished initialising, a passenger can present a printed barcode, mobile phone, or tablet and the reader will read the ticket.

An indication that the ticket is valid will typically happen within a half a second. This read performance is dependent on both hardware and software.

2. Specifications

Specification	Details
Dimensions (L x H x W)	103.1 x 106.3 x 36.9 mm (Flush bezel and without SAM option)
Weight	250 g (Without SAM option)
Environmental	Operating temperature: -20ºC to 60ºC Storage temperature: -30ºC to 80ºC Humidity: 95% RH, non-condensing
Body	Black ABS
Glass	4 mm Toughened White Soda Lime; BS EN60068-2-75 & IEC 62262:2002, rated to 3.5 J impact
Power requirements	5 V DC • ATR200-xx-U (USB) can be powered from USB only or 5 V DC 2A power supply for full speed operation • ATR200-xx-P (PoE) is powered via Ethernet PoE Ethernet port or power injector required IEEE 802.3af Type 1: 15.4W, 44.0V to 57.0V Supported modes: Alternative A (data pair) or B (spare pair) • All other units require 5 V DC power supply
Electrical interface	Serial (RS232C) and 5 V USB
Interface	USB composite device, with: • Human interface device (HID) interface for barcode reading • Chip card interface device (CCID)/personal computer/smart card (PC/SC) interface for NFC Serial device: • Barcode, NFC reader as separate devices
Barcode reading	Linear: EAN. UPC, Code 2 of 5, Interleaved 2 of 5, IATA 2 of 5, Code 39, Code 128 2D: IATA resolution 792, PDF417, Aztec, DataMatrix and QR codes Media types: Reads smartphone, tablet and smartwatch displays and paper documents and tickets
Contactless reading (optional)	Supported media: ISO14443 type A and B cards (Java cards); max baud 424K (extendable to 848K) Mifare UL, Classic 1K, Classic 4K, UL-C, UL-EV1, Mifare Plus; max baud 106K, mobile phone (PCE) Operating frequency: 13.56 MHz Operating distance: 20 mm
Contactless payment capability (optional)	EMV Level 2 Certification (VISA, Mastercard, AMEX, Discover) PCI SRED - V5.0
PSAM reading (optional)	Supports SAM cards meeting the following specification: • Compliant with BS ISO/IEC 7813-3:2006 • Supporting protocol T0 and/or T1
MTBF	250,000 hours

Specification	Details
Approvals	EMC approvals: • FCC 47CFR Part 15 Subpart B Class A, • EN 55032 Class B, EN 55024 Radio: (NFC version only): • FCC Part 15 Subpart C, • CE: EN 301 489-1 v1.8.1 (2008-04), • CE: EN 302 291-1 v1.1.1 (2005-07)
Safety	• IEC 62471: 2006 • EN 60950-1:2006+A12:2011
Ingress	Front Face ONLY - IP67 certified to BS EN 60529:1992

3. Drawings

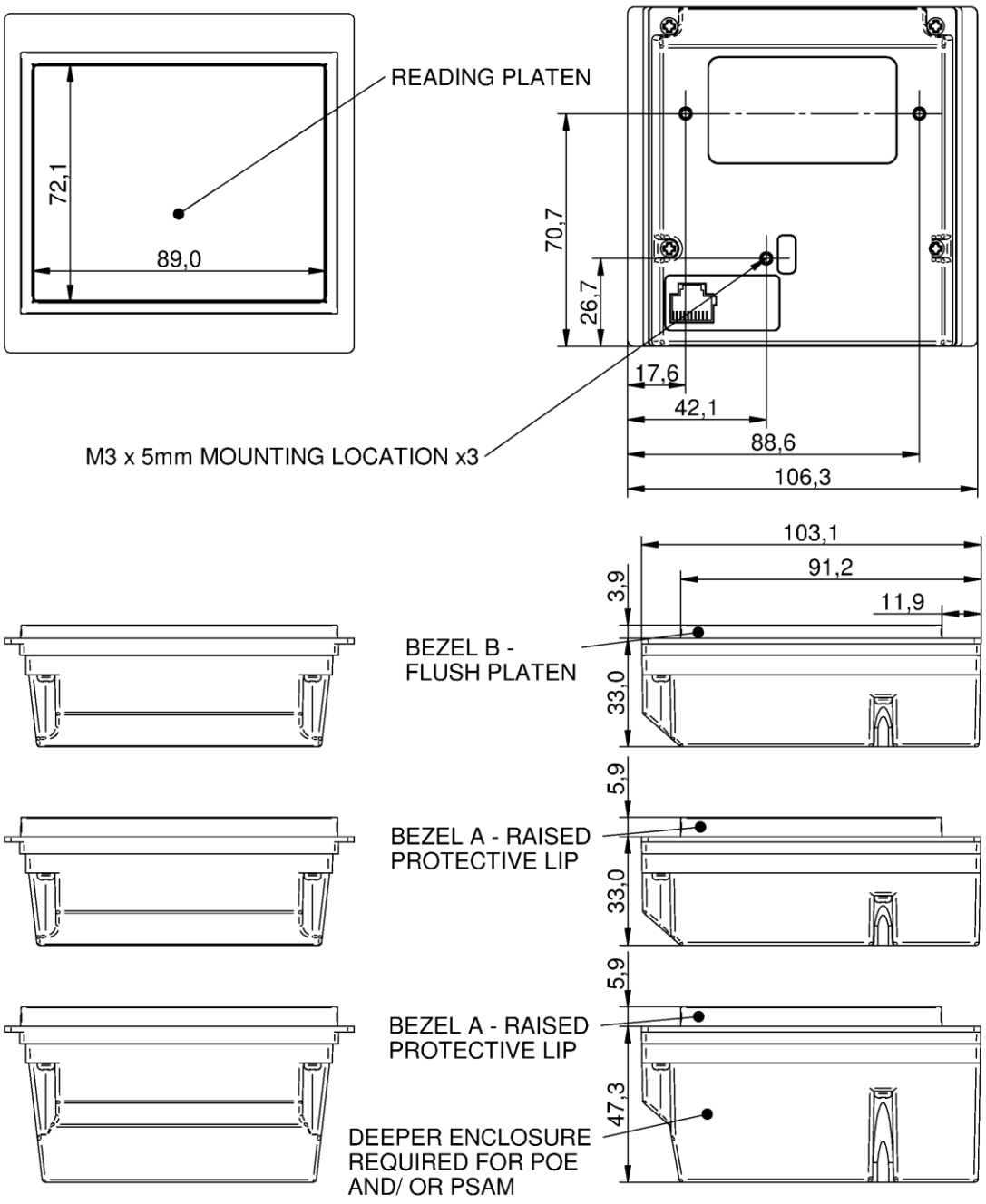

Figure 2: TripTick dimensions

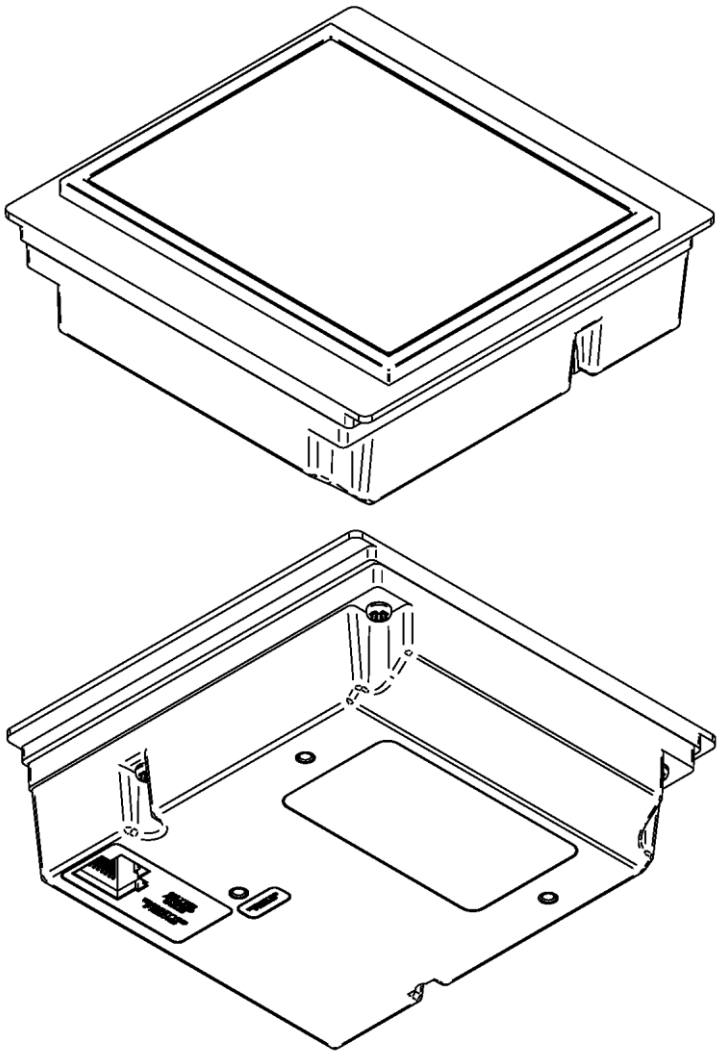

Figure 3: Reading platen and location of the connectors on the underside of the TripTick unit

4. Installation

Unpack TripTick

Unpack TripTick and ensure that you have the following items:

- Advisory notice card.
- TripTick device with appropriate cables for connection to the host.

  Cables provided depend on the TripTick model. Refer to ***Model numbers*** (on page 68) and ***Part and host cable numbers*** (on page 69).
- External 5V 3A power supply (not provided for an ATR200 (USB) or models using a PoE connection).

Report any missing items or damage immediately to your Sales Representative.

Connection

Connect TripTick via USB, RS232/RS485 or Ethernet depending on the product version.

Note: *Cable length is 2 m for USB, Serial and Ethernet versions.*

USB connection

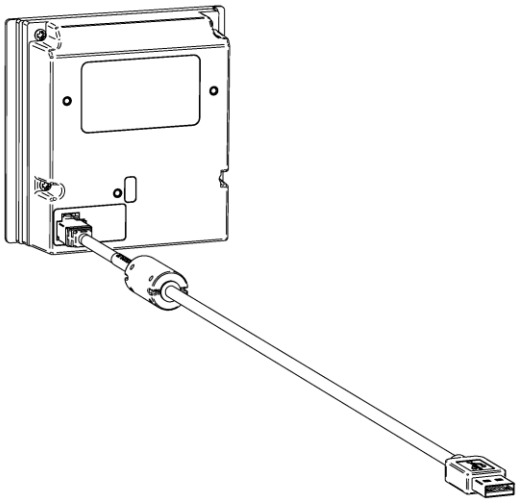

Figure 4: USB connection

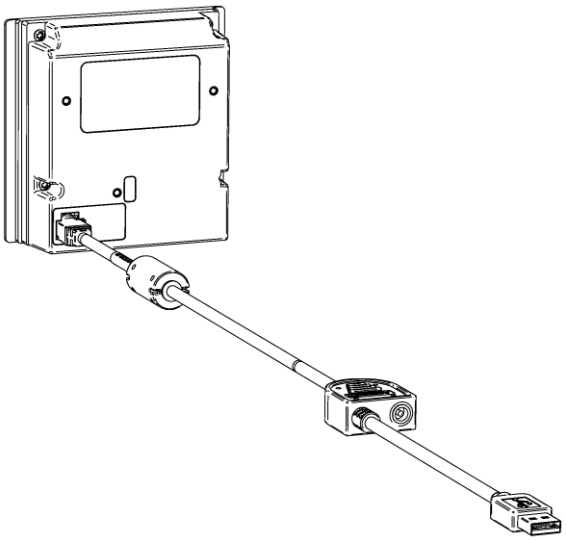

Figure 5: USB plus DC connection

Note: *When using the USB plus DC connection option, connect the 5 VDC 3 A power supply provided by Access-IS to the 3.5 mm DC socket within the cable mould.*

Serial connection

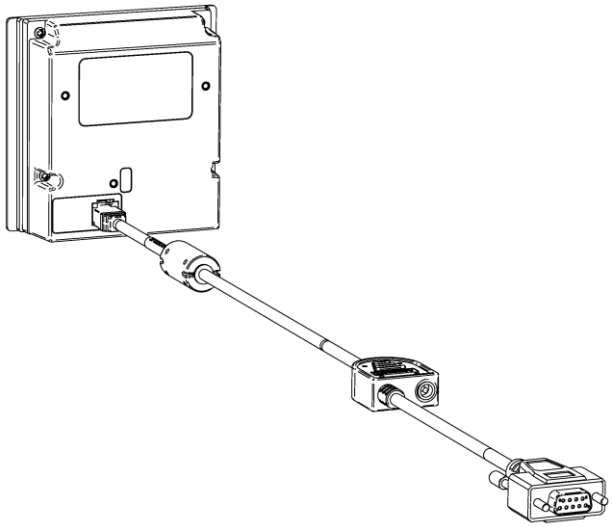

Figure 6: Single serial connection

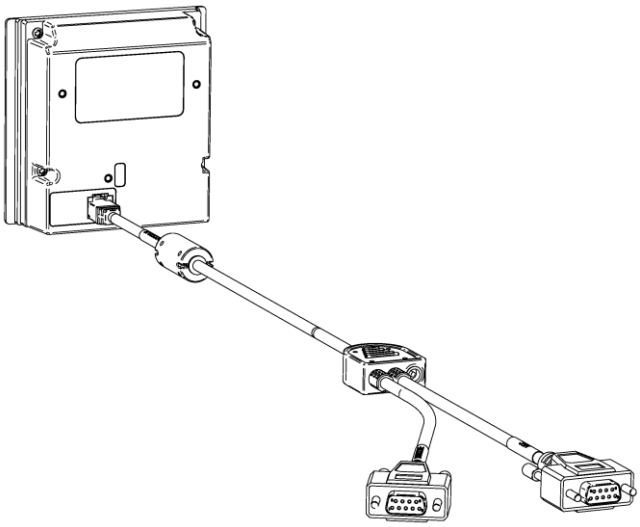

Figure 7: Dual-serial connection

Note: *When using single and dual-serial connection options, connect the 5 VDC 3 A power supply provided by Access-IS to the 3.5 mm DC socket within the cable mould.*

Ethernet connection

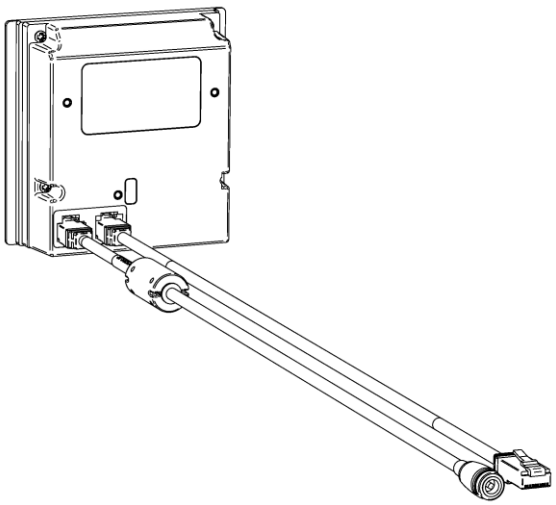

Figure 8: Ethernet plus DC connection

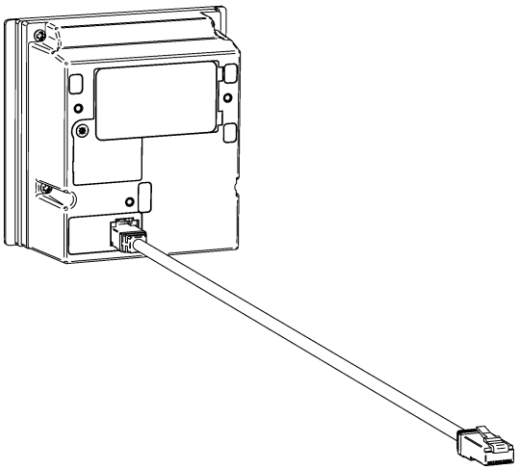

Figure 9: Power over Ethernet (PoE) connection

Note: *When using the Ethernet plus DC option, connect the 5 VDC 3 A power supply provided by Access-IS to the 3.5 mm DC socket within the cable mould.*

Mounting

Mount TripTick into a kiosk, gate or similar, if required. Refer to the following drawing for TripTick's dimensions (in millimetres) and mounting points.

Important: *To optimise the performance of your Access-IS device:*

DO NOT position TripTick in direct sunlight. Failure to observe this may lead to the scan performance deteriorating or even failing completely.

DO NOT install TripTick with its NFC/RFID antenna within 40 mm of a large metal or electrically-conductive component or structure. Failure to observe this instruction may lead to the NFC performance deteriorating or even failing completely.

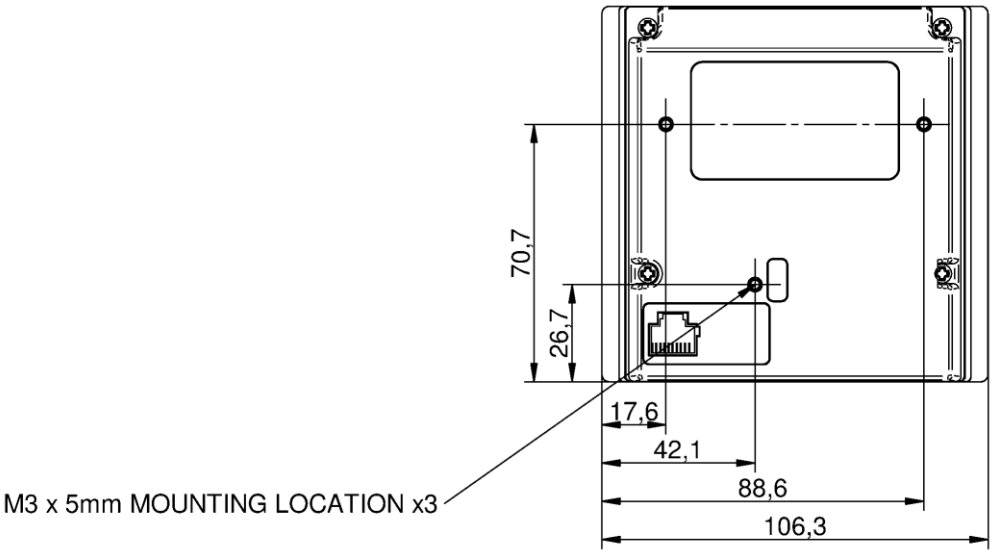

Figure 10: TripTick dimensions and mounting points

Use three M3 screws (not provided) to mount the unit. Maximum insertion depth is 5 mm; minimum recommended insertion depth is 3 mm.

Barcode interface options

Serial connection

Connect a serial TripTick device using an RS232/RS485 interface directly into a COM port. You must specify the baud rate, parity, data bits and stop bits.

Note: *A serial TripTick communicates directly with the COM port and does not require any additional drivers to be loaded.*

USB connection

Connect a USB TripTick device using one of three possible options. These options are compatible with all Linux and Windows operating systems from XP onwards.

Keyboard interface

Virtual keyboard using Windows or Linux drivers

This option allows the device to operate without additional drivers, with TripTick emulating a keyboard. This is one-way communication; it is not possible to control the device directly in this mode.

This mode will be slower than the other options as it adds an inter-character delay when typing the barcode data. For higher throughput, consider using a HID or CDC interface.

CDC interface

Virtual serial mode using the Windows CDC driver

This option assigns a COM port, and the device communicates as a virtual serial device. Due to the nature of CDC serial port drivers, the COM port disappears if the unit is unplugged.

HID interface

Access-IS recommends the use of the HID interface for reliability. A HID interface recovers properly in the event of accidental disconnects or system power fluctuations; a CDC interface may not recover in these situations.

HID interface using the Access driver (Windows only)

The Access Serial Ports Service (ASPS) driver is fully configurable and outputs data in virtual serial or virtual keyboard. The output itself can be parsed and reformatted. The serial port is permanent and does not disappear if you unplug or hot swap the unit. This is one-way communication and the only command that you can send to the device is **AIS_BO** to enable or disable barcode reading. Refer to ***Development commands*** (on page 30) for more information.

HID interface without the Access driver

This method is only suitable if you are familiar with HID programming.

It is possible to communicate directly with TripTick using the operating system's built-in HID drivers. In this instance, HID reports, exactly 64 bytes in length, are sent between the host and TripTick.

The implementation of this driver and the method of interaction will depend on the version of the host operating system. You should refer to the HID programming guide for the operating system that you are using.

Refer to ***HID reports*** (on page 72) for the details of the HID reports used with TripTick.

NFC interface options

Serial connection

Connect the NFC module using an RS232 interface directly into a COM port.

Note: *A serial TripTick communicates directly with the COM port and does not require any additional drivers to be loaded.*

USB connection

The NFC module enumerates as a standard Chip Card Interface Device (CCID) smartcard reader. When you connect the device to the host, the NFC module uses the default Windows CCID drivers. It is not necessary to install custom drivers when running Windows XP and above.

Barcode module installation (serial device)

A serial TripTick communicates directly with the COM port and does not require any additional drivers to be loaded. Serial connectors are labelled: CONN1 = Barcode, CONN2 = NFC module.

1. Switch off the computer.
2. Connect the serial cables to COM ports on the computer and finger-tighten the two thumbscrews to secure the connectors to the port.
3. If using a USB power injector cable, plug the injector cable into the coaxial power connector on the splitter cable and then plug the USB connector into a powered USB port on the computer.

   If using an Access-supplied power supply, plug the power cable into the coaxial power connector on the splitter cable and then connect the external power supply to an AC outlet.
4. Once the device is connected, switch on the computer.

Barcode module installation (USB device)

Note: *If you intend to use the Access driver, ensure that you install the driver before you connect TripTick to the computer.*

Driverless keyboard output

There is no additional driver required for this mode. Connect the USB cable from TripTick to a USB port on the computer.

CDC Windows driver

This method of USB installation uses the Windows CDC drivers.

For this method to operate, install the CDC drivers using the file, **`AccessISUSBCDC.inf`**, which you can download from the ***Access-IS website*** (***http://www.access-is.com/gettingstarted/***).

The download (USB Driver for CDC Mode) includes full instructions for use.

Windows assigns a virtual COM port to the TripTick device. You can find out the COM port number in Device Manager. You will require the port number to configure TripTick.

Custom HID

HID interface using the Access serial driver (Windows only)

The recommended method for using a USB TripTick is to configure the device to operate in HID mode. This allows the device to communicate with the Access driver.

For this method to operate, you must install first the Access driver (Access Serial Ports Service (ASPS)).

Download ASPS from the ***Access-IS website*** (***http://www.access-is.com/gettingstarted/***).

The download (ASPS Software) includes full instructions for use.

Ensure that you install the driver *before* connecting TripTick to the host.

HID interface without the Access driver

There is no additional driver required for this mode. Connect the USB cable from TripTick to a USB port on the computer.

Barcode module installation (Ethernet device)

An Ethernet TripTick communicates via a network socket interface using the same command set as the Serial and USB CDC devices.

By default, the device is configured to obtain an address via DHCP. When first connected you will need to find the Ethernet IP address from your DHCP server. Alternatively, if you have access to a TripTick USB data cable you can use this to configure the Ethernet options without having to know the IP address by following the connection instructions for the USB device above.

1. Connect the USB power-cable and Ethernet cables as shown in ***Ethernet connection*** (on page 14).
2. If using an Access-supplied power supply, plug the power cable into the coaxial power connector on the USB power cable and then connect the external power supply to an AC outlet.
3. Connect the Ethernet connector to a network switch or Ethernet port on your computer, depending on how you intend to supply a DHCP address.
   - If connecting to your computer, you will need to run a DHCP server, for example, ***DHCP server for Windows (http://www.dhcpserver.de/)*** to provide an IP address to the TripTick.
   - If connecting to a network switch, you will need to locate your DHCP server and find the IP address allocated to the MAC address which is printed on the label on the back of the TripTick unit.
4. Once you know the IP address you can connect to the TripTick barcode reader via Port 4161.
5. If you wish to use ATE or a similar serial terminal application on a Windows PC, you will need to run, for example, the ‘com2tcp’ and ‘com0com’ programs (available from here: ***http://com0com.sourceforge.net/***) to set up a virtual COM port connected to a network socket.
6. If you are using a Linux computer you can use the ‘socat’ application (***https://linux.die.net/man/1/socat***) to create a virtual serial link to your TripTick over Port 4161.

NFC module installation (serial device)

A serial TripTick communicates directly with the COM port and does not require any additional drivers to be loaded. Serial connectors are labelled: CONN1 = Barcode , CONN2 = NFC module. Refer to ***Barcode module installation (serial device)*** (on page 17) for installation instructions.

NFC module installation (USB device)

When you connect a USB-connected TripTick device to the host, Windows automatically detects the hardware and installs the standard CCID smartcard reader drivers. Some versions of Windows may prompt you to search automatically for a driver.

The NFC module also exposes a HID interface for configuration and control. Refer to ***NFC module management interface commands*** (on page 57) for the command set and its responses.

In Device Manager, the smartcard reader and HID-compliant device represent the NFC module. The barcode device appears under Ports (COM & LPT).

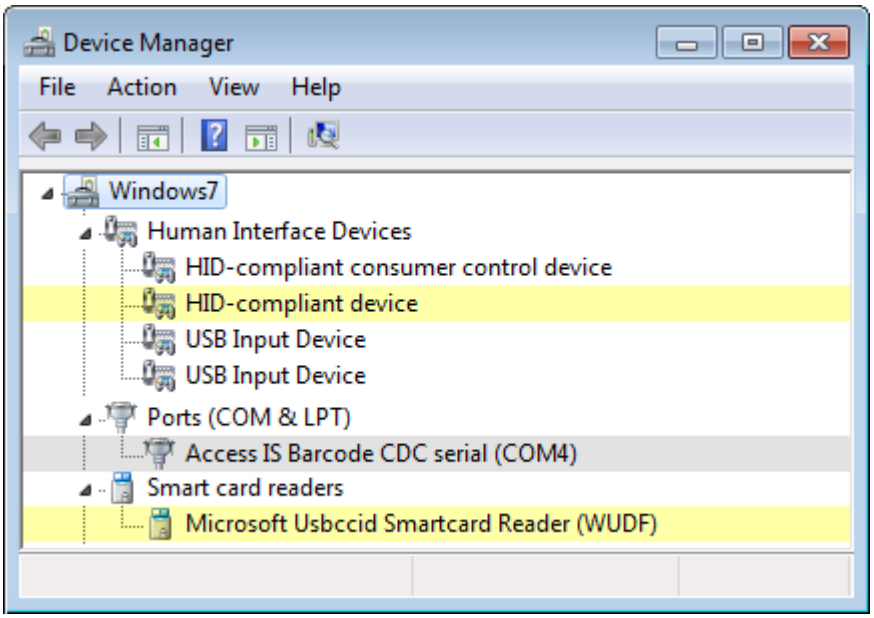

Figure 11: NFC module and barcode device in Device Manager (other device types not shown)

NFC module installation (Ethernet device)

An Ethernet TripTick communicates via a network socket interface using the same command set as for the Serial device.

By default, the device is configured to obtain an address via DHCP. When first connected you will need to find the Ethernet IP address from your DHCP server. Alternatively, if you have access to a TripTick USB data cable you can use this to configure the Ethernet options without having to know the IP address by following the connection instructions for the USB device above.

1. Connect the USB power-cable and Ethernet cables as shown in ***Ethernet connection*** (on page 14).
2. If using an Access-supplied power supply, plug the power cable into the coaxial power connector on the USB power cable and then connect the external power supply to an AC outlet.
3. Connect the Ethernet connector to a network switch or Ethernet port on your computer, depending on how you intend to supply a DHCP address.
   - If connecting to your computer, you will need to run a DHCP server, for example, ***DHCP server for Windows (http://www.dhcpserver.de/)*** to provide an IP address to the TripTick.
   - If connecting to a network switch you will need to locate your DHCP server and find the IP address allocated to the MAC address which is printed on the label on the back of the TripTick unit.
4. Once you know the IP address you can connect to the TripTick barcode reader via Port 4162.
5. If you wish to use ATE or a similar serial terminal application on a Windows PC, you will need to run, for example, the ‘com2tcp’ and ‘com0com’ programs (available from here: ***http://com0com.sourceforge.net/***) to set up a virtual COM port connected to a network socket.
6. If you are using a Linux computer you can use the ‘socat’ application (***https://linux.die.net/man/1/socat***) to create a virtual serial link to your TripTick over Port 4162.

Test the device

Once you have connected the device and installed the relevant drivers, if applicable, you can test the device. To do this, wave a piece of paper in front of the glass; the reader's LEDs should illuminate. If the device fails to respond when connected to the host, refer to the *Troubleshooting* section in this document.

Barcode configuration software

Connect to, and configure TripTick using your own configuration tool, a terminal emulation program or the Access-IS configuration tool, which you can download from the ***Access-IS website*** (***http://www.access-is.com/gettingstarted/***).

Refer to the ***Barcode command reference*** (on page 27) for details of the barcode commands, which you can use to configure TripTick.

Communicate with the NFC module

Once the NFC module is enumerated, it registers itself with the Windows Smartcard Resource Manager. Since the NFC module is Personal Computer/Smart Card **(**PC/SC) compatible, you can use standard Windows smartcard functions to communicate with the module through the Windows Smartcard Resource Manager API. Refer to the Microsoft website for more detailed information on the Smartcard Resource Manager API.

For more information on the operation of TripTick's NFC reader, see ***NFC operation*** (on page 32). For commands, refer to ***MIFARE media commands and responses*** (on page 39) and ***NFC module management interface commands*** (on page 57).

5. Troubleshooting

If TripTick does not appear to be working, refer to the following table to help identify and resolve the problem.

For further assistance, contact ***Customer Support*** (***mailto:support@access-is.com***).

Alternatively, use the *Contact Customer Support* page on the Access-IS website.

Note: *While in warranty, a faulty unit may be returned to Access-IS for repair. Do not attempt to disassemble TripTick if it does not operate correctly. Any attempt to do so may be dangerous and will invalidate the warranty, if applicable.*

Table 1: Troubleshoot TripTick

Problem	Solution
TripTick not transmitting data to host	Check that all cable connections between TripTick and host are secure. Ensure that the unit has power.
TripTick cannot scan barcode	Ensure that the unit is configured to read the barcode that you are scanning. If scanning a document, ensure that the print quality is good. If scanning a barcode on a mobile phone, ensure that you set the screen backlight on the phone to its brightest setting.

6. Maintenance

Cleaning

Clean the glass with a lint-free cloth. If the glass is dirty, wipe the glass with a lint-free cloth moistened with isopropyl alcohol or use an alcohol wipe.

Do not use abrasive cleaners.

Storage

Store the unit in its original box, at a temperature of -30°C to 80°C.

7. Barcode operating modes

TripTick operates as defined by the **AISOMD** command. Refer to the ***Barcode command reference*** (on page 27) for a list of commands that you can send to configure TripTick.

Mode summary

Dumb mode

TripTick is a one-way communication device.

The device detects the media and activates the imager and illumination. When TripTick reads the barcode, it sends the data to the host, activates the 'Good Read' indicators, and disables the imager and illumination. The imager and illumination do not reset until the TripTick sensor fails to detect any media for 0.5 seconds.

Host mode

TripTick is a two-way communication device that reads barcodes and waits for a host to accept or reject the barcodes.

The device detects the media and activates the imager and illumination. When the device reads the barcode, it sends the data to the host and disables the imager and illumination. TripTick waits for a response from the host to accept or reject the data, which activates the 'Good Read/Bad Read' indicators on the device. TripTick waits for up to two seconds for an 'Accept/Reject/Ignore' command to activate indicators. The host sends an 'Ignore' command to reset the imager if no response from the indicators is required. The imager and illumination do not reset until the TripTick sensor fails to detect any media for 0.5 seconds.

The 'Ignore' command requires version 1.0.21 (or later) of the firmware.

Dumb mode

The process for TripTick in Dumb mode is as follows:

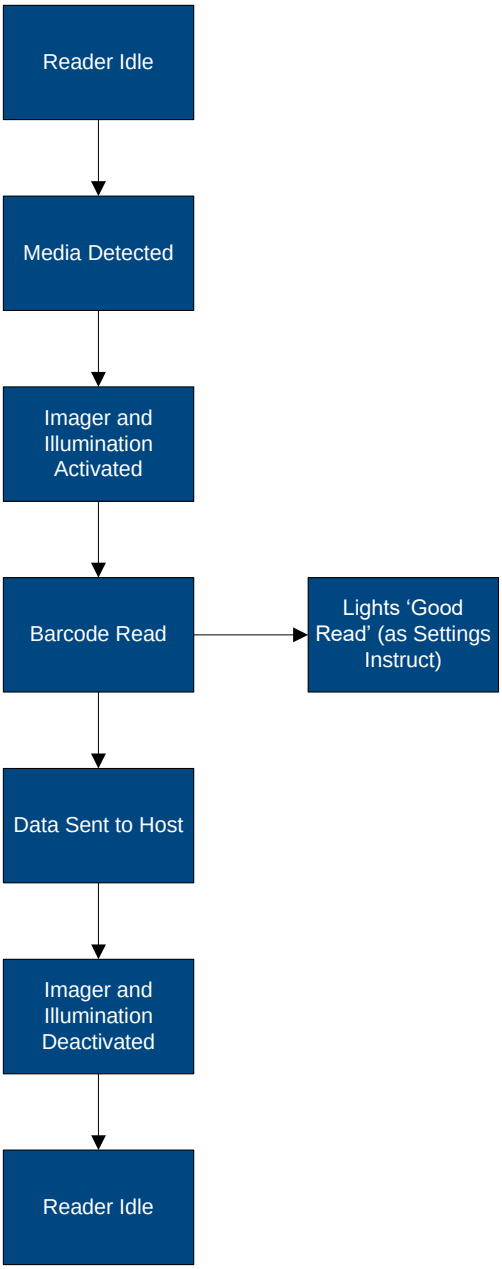

Figure 12: Dumb mode process flow

Dumb mode example

Comments	TripTick command to host	Host command to TripTick
Media placed in front of TripTick.	-	-
Imager activated and barcode scanned. Illumination activated as defined in the settings.	Data sent as configured (USB/Serial)	-
No media detected for 0.5 seconds; TripTick resets.	-	-

Host mode

The process for TripTick in Host mode is as follows:

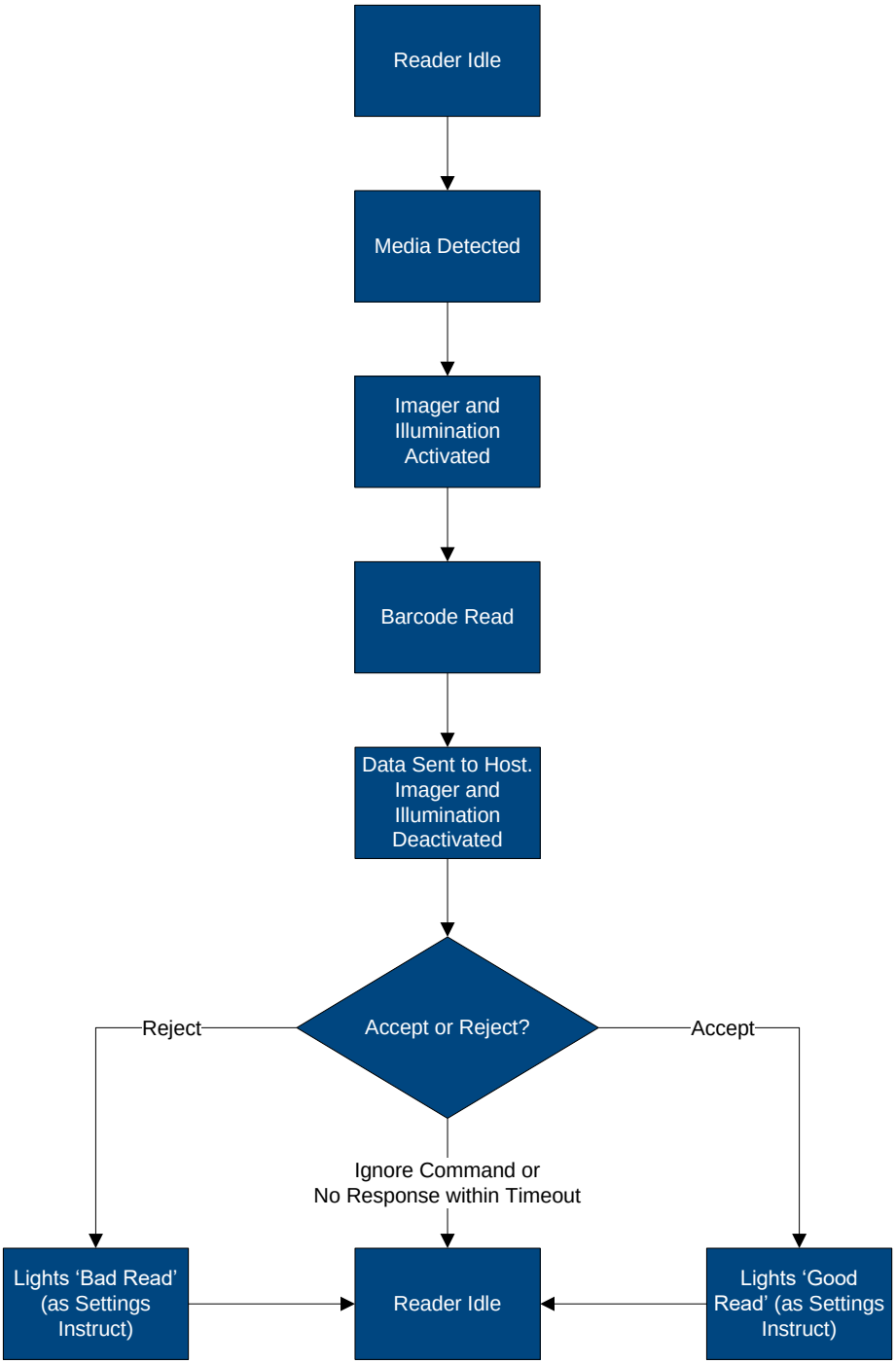

Figure 13: Host mode process flow

Host mode example

Accept

Comments	TripTick command to host	Host command to TripTick
Media placed in front of TripTick.	-	-
Imager activated and barcode scanned. Illumination activated as defined in the settings.	Data sent as configured (USB/Serial)	-
Host decides to accept or reject the data.	-	'Good Read': **AISXXR0**
Lights activated as defined in the 'Good Read' settings.	-	-
No media detected for 0.5 seconds; TripTick resets.	-	-

Reject

Comments	TripTick command to host	Host command to TripTick
Media placed in front of TripTick.	-	-
Imager activated and barcode scanned. Illumination activated as defined in the settings.	Data sent as configured (USB/Serial)	-
Host decides to accept or reject the data.	-	'Bad Read': **AISXXR1**
Lights activated as defined in the 'Bad Read' settings.	-	-
No media detected for 0.5 seconds; TripTick resets.	-	-

Ignore

Comments	TripTick command to host	Host command to TripTick
Media placed in front of TripTick.	-	-
Imager activated and barcode scanned. Illumination activated as defined in the settings.	Data sent as configured (USB/Serial)	-
Host decides to accept or reject the data.	-	'Ignore and Continue': **AISXXR2**
No media detected for 0.5 seconds; TripTick resets.	-	-

8. Barcode command reference

Commands are sent with a prefix of **[0x16][0x4D][0x0D]** causing the command sequence to take the form **[0x16][0x4D][0x0D]<Menu Command>**. The menu commands are six characters long with a parameter (if required).

To send a command to modify a configuration parameter

Send the six-character command concluded by a dot '.' or an exclamation mark '!'. The dot stores the setting permanently and the exclamation mark keeps it temporarily until power is removed from the device.

For example, **[0x16][0x4D][0x0D]AISKBL1.** sets the keyboard localisation to United States when the device is operating as a USB keyboard.

To query the current settings (including a temporary one)

Send the six-character command with a '?' instead of the parameter and the ATR110 will return the command with the current setting.

For example, **[0x16][0x4D][0x0D] AISINF?** queries the device interface and returns the current value.

To query the stored value

Send the six-character command with a '^' instead of the parameter and the ATR110 will return the command with the stored setting.

For example, **[0x16][0x4D][0x0D] AISINF^** returns the current illumination mode.

To list parameter options

Send the six-character command with a '*' instead of the parameter and the ATR110 will return the command with the parameter options.

Basic configuration

These commands set the device interface, connection parameters and specify the operating mode.

Table 2: Basic configuration commands

Command	Description	Default	Parameters/Range
AISINF	Selects the device interface. When a Serial cable is used, the configuration is overruled and **AISINF0** is used. When a USB cable is used, the configuration **AISINF0** is overruled and **AISINF1** is used.	0	0 - Serial 1 - USB serial (CDC) 2 - USB keyboard 3 - HID POS
AISBAU	Sets the baud rate for a Serial connection. Only used when **AISINF** is set to 0 (Serial).	9	0 - 300 bps 1 - 600 bps 2 - 1200 bps 3 - 2400 bps 4 - 4800 bps 5 - 9600 bps 6 - 19200 bps 7 - 38400 bps 8 - 57600 bps 9 - 115200 bps
AISSCP	Sets the connection parameters for a Serial connection. Only used when **AISINF** is set to 0 (Serial).	2	0 - 7N1 1 - 7N2 2 - 8N1 3 - 7E1 4 - 7E2 5 - 8E1 6 - 7O1 7 - 7O2 8 - 8O1
AISKBL	Keyboard localisation: this defines the Windows keyboard mapping for correct output of characters. Only used when **AISINF** is set to 2 (USB keyboard).	0	0 - US (United States) 1 - UK (United Kingdom) 2 - IT (Italy) 3 - ES (Spain) 4 - DE (Germany) 5 - CH (Switzerland) 6 - CZ (Czech Republic) 7 - FR (France) 8 - BE (Belgium) 9 - SE (Sweden)
AISCHR	Sets the inter-character delay (in milliseconds). Only used when **AISINF** is set to 2 (USB keyboard).	2	0–250 milliseconds
AISOMD	Indicator mode setting.	0	0 - Dumb mode 1 - Host mode

Command	Description	Default	Parameters/Range
DLYGRD	Sets the delay between successful reading of one barcode and the reading of another barcode. Each unit is equivalent to 1 millisecond.	2000	0–25000

Prefix and suffix solutions

These commands allow you to add a prefix and/or suffix to all barcodes.

Note: *If you send more than one prefix or suffix to the device, they will stack in chronological order. You must send a clear command if you want to use a single prefix or suffix.*

Table 3: Prefix and suffix commands

Command	Description	Default	Parameters/Range
PREBK299xx	Adds a prefix to all barcode symbologies. Any two-character hex ASCII code can replace xx. For example, to add STX (Start of Text) as a prefix, use the command **PREBK29902.** You can add more than one prefix, as required.	-	xx - Hex value
PRECA2	Clears all prefixes.	-	-
SUFBK299xx	Adds a suffix to all barcode symbologies. Any two-character hex ASCII code can replace xx. You can add more than one suffix, as required. For example, to add CR (Carriage Return) and ETX (End of Text) as a suffix, use the command **SUFBK2990D03.**	-	xx - Hex value
SUFCA2	Clears all suffixes.	-	-

Indicator control

These commands control the behaviour of the ‘Good Read’ and ‘Bad Read’ LEDs.

Note: *There are no lid lights on the TripTick device; these are replaced by the NFC antenna.*

Table 4: Indicator LED commands

Command	Description	Default	Parameters/Range
AISGDT	‘Good Read’ LED indicator duration. Each unit is equivalent to 100 milliseconds.	5	0–200
AISBDT	‘Bad Read’ LED indicator duration. Each unit is equivalent to 100 milliseconds.	8	0–200
AISGSL	Switches between LED locations for the ‘Good Read’ indicator.	1	0 - No lights 1 - Board green lights 2 - Board red lights

Command	Description	Default	Parameters/Range
AISBSL	Switches between LED locations for the 'Bad Read' indicator.	2	0 - No lights 1 - Board green lights 2 - Board red lights

Development commands

Firmware and imager levels

The firmware levels identify the release and build of a unit. Send the command **AISFWV?** to obtain this information. For example: TTU01.00.00 is a first generation TripTick.

To check the latest firmware version or to update firmware, contact ***Customer Support*** (***support@access-is.com***).

Table 5: Firmware and imager commands

Command	Description	Default	Parameters/Range
AISXXR	Simulates read outcome. Only applicable to Host mode.	-	0 - 'Good Read' 1 - 'Bad Read' 2 - 'Ignore' (requires version 1.0.21 of the firmware)
AISRDS	Changes the configuration back to its default values. ***Warning*****:** *This command resets all parameters to their default values, including any values specific to your stored configuration.*	-	1
AISFWV	Returns the version of the firmware.	-	-
AIS_WA	Returns the firmware version of the imager.	-	-
AIS_TD	Returns the timestamp of the firmware release.	1	-
AIS_BO	Enables or disables barcode reading. This command is stored in volatile memory so will return to the default setting on power cycle.	1	0 - Off 1 - On
AISDLE	Include DLEs (Data Link Escape).	0	0 - Off 1 - On
232CRD	CTS is raised when a 'Good Read' output is received.	0	0 - Off 1 - On
232CTS	Hardware handshaking - requires the CTS to be high.	0	0 - Off 1 - On

Counter

These commands display the number of 'Good' or 'Bad' reads made by the device.

Table 6: Counter commands

Command	Description	Default	Parameters/Range
AISGRC	'Good Read' counter. Cannot be reset.	0	-
AISBRC	'Bad Read' counter. Cannot be reset.	0	-

Network configuration

These commands set the device interface address, mask and operation mode.

Table 7: Network configuration commands

Command	Description	Default	Parameters/Range
ETHADn	Selects the device interface IP address octets. n is a number between 1 and 4 representing the octet number. For example, **ETHAD1** is 192 if the address is 192.168.1.1, **ETHAD2** is 168 and so on.	0	0–255
ETHMKn	Selects the device interface IP mask octets. n is a number between 1 and 4 representing the octet number. For example, **ETHMK1** is 255 if the mask is 255.255.255.0, **ETHMK4** is 0 and so on.	0	0–255
ETHMOD	Sets the network addressing mode.	0	0 – DHCP 1 - Static

9. NFC operation

Near Field Communication (NFC) is a standard form of communication between an NFC reader and NFC supported media like smartcards, tags and smart phones.

NFC is a short-range wireless technology, which allows two devices to exchange small amounts of data in a secure manner over a distance of a few centimetres.

The adoption of NFC technology by mobile devices and passports has seen NFC technology gain in popularity. Consumers can now perform contactless transactions with a single touch, and use NFC devices for public transport, ticketing and access control.

TripTick operates in NFC reader mode and processes NFC (and barcode) data from a single point of presentation in any orientation.

The NFC module in TripTick is Personal Computer/Smart Card **(**PC/SC) compatible and you can use standard Windows smartcard functions to communicate with the module through the Windows Smartcard Resource Manager API.

SMC reader (available only on PSAM versions)

Apart from the NFC reader, CPM can support four additional PSAM readers. The PSAM readers are named as SMCx (where x ranges from 0 to 3). All PSAM readers are ISO7816 compliant both at hardware and software level. Further, each PSAM reader is interfaced to the host system as an independent USB CCID reader. The host application can send Application Protocol Data Unit (APDU) commands to the SMCx slots using the PC/SC smartcard API.

Note: *The format of the command APDUs and the responses depend on the type of PSAM used. Refer to the user manual of the PSAM for the command and response formats.*

Summary of operation

An NFC reader reads/writes blocks from/to a microprocessor or MIFARE card. When NFC media connects to the TripTick's NFC reader, the device retrieves an Answer To Reset (ATR) from the card.

The ATR specifies certain communication parameters, including the card's nature and state.

- If the ATR identifies a microprocessor card, the host application sends and retrieves APDU commands and responses. The format of the command and response APDUs depend on the type of media.
- If the media type is a MIFARE card, the NFC module constructs an ATR from the fixed elements that identify the card. See ***MIFARE cards*** (on page 38) for more information.

Once the application detects a MIFARE-type card, it can then use MIFARE commands to communicate with it. See ***MIFARE media commands and responses*** (on page 39) for more information.

The host sends APDU or MIFARE commands to the card over the PC/SC interface using the `SCardTransmit` function in the Windows Smartcard API and gets data back from the card.

Once communication is complete, or a user removes the card, the NFC module disconnects from the card and waits for another card connection.

Figure 14 (on page 33) shows an overview of the process that the NFC module in TripTick uses to identify and communicate with contactless media.

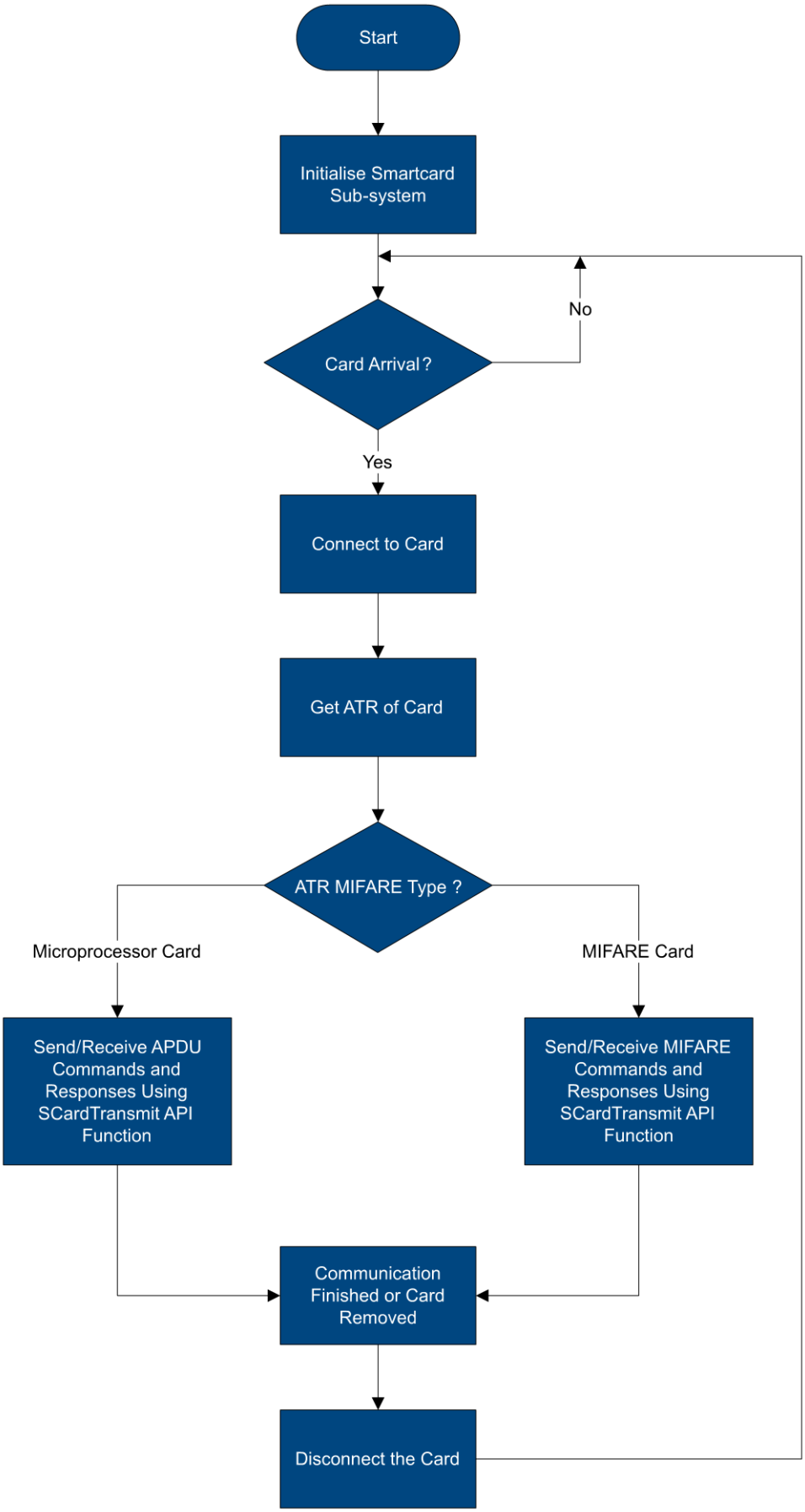

Figure 14: NFC module-contactless media process flow

Serial communication

Communication parameters

The NFC module in a serial-connected TripTick uses the following default serial communication settings. Change the serial device baud rate using the ***Set serial interface baud rate*** command (on page 66).

Table 8: Serial communication default parameters

Parameter	Value
Baud rate	115200
Data format	8 bits
Parity	None
Stop bits	1
Flow control	RTS/CTS

Communicating with individual readers

The individual readers inside the NFC module are identified by unique slot IDs. All CCID packets should have a CCID header, which has a byte field called **`bSlot`**. This field specifies the slot ID of the reader with which the application wishes to communicate.

The slot ID value uniquely identifies the reader (within NFC module) to which the CCID packet is sent, and identifies the reader which is sending the response back to the application.

The following table shows the Slot ID values and their mapping to the readers.

Table 9: Slot ID and reader mappings

Slot ID value	Reader
0	NFC
1	Smartcard #0 (SMC0)
2	Smartcard #1 (SMC1)
3	Smartcard #2 (SMC2)
4	Smartcard #3 (SMC3)
`[0xFF]`	Management

The Management Interface is used to set the operating and debug parameters of the NFC module.

Note: *Do not send commands to a reader that is not enabled.*

Communication format

The data is sent to and from the NFC module in 64-byte chunks. The following diagram shows how the NFC module transfers a 256-byte CCID packet.

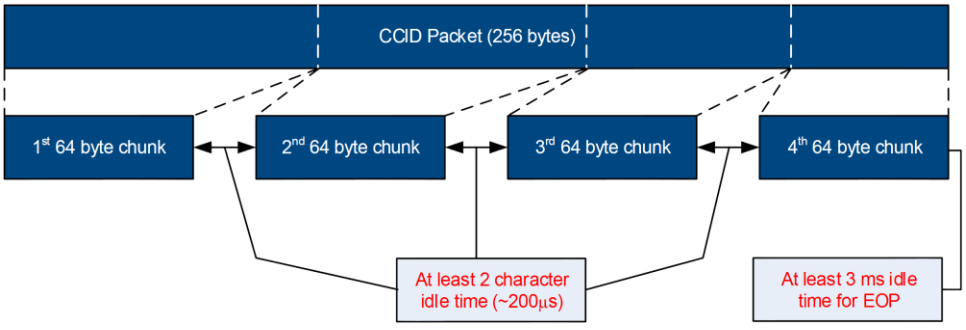

Figure 15: Example data transfer of a 256-byte data packet

The entire CCID packet including the header is broken down into 64-byte packets, which are transmitted one at a time. Ensure that there is at least a two-character idle time (approximately 200 microseconds) between consecutive 64-byte chunks. This idle time gives an opportunity for the serial device to save and clear its receiving buffer. The last data packet can be less than 64 bytes long. End-of-packet (EOP) is indicated by at least three milliseconds of idle time.

When EOP is received by the module, it internally checks the CCID packet's length indicated in the CCID packet header.

- If the CCID packet header's length is equal to (or less than) the received CCID packet length, then the module processes that packet.
- If the CCID packet header's length is more than the received CCID packet size length, then the module waits for the next 64-byte chunk to arrive.

Note: *When a CCID packet transmission starts, the first 64-byte chunk includes the CCID header, which indicates the slot ID where the data is being sent to or received from.*

Each command message sent to a particular reader receives an appropriate response from the NFC module. The serial host does not send another command message to a reader until it receives a response. However, the serial host may send command messages concurrently to different readers at the same time. The NFC module may not respond in the same sequence as the sent commands. The response depends on the internal priority of the readers and the time taken to process the request by the media.

Notifications and data exchanges (serial connection)

Media arrival and removal notification

The NFC reader sends out three bytes to notify media arrival or removal.

The following table shows the format of these three bytes:

Table 10: Media arrival or removal

1st byte	2nd byte	3rd byte
Always `[0x50]`	Slot / media status	Media type

Slot / media status

The first most significant 4 bits denote the reader slot ID. The least 4 bits denote the media status.

If media status is 0 then the media is not present. If it is 1, then the media is present.

Media type

The value of the media type byte indicates the media type.

Table 11: Media type byte values

Media type value	Type	Current support
No Media present	`[0x00]`	Yes
ISO14443-4 A	`[0x01]`	Yes
ISO14443-4 B	`[0x02]`	Yes
Mifare Classic 1K	`[0x03]`	Yes
Mifare Classic 4K	`[0x04]`	Yes
Mifare Ultralight	`[0x05]`	Yes
Mifare Plus	`[0x06]`	Yes
Felica media	`[0x07]`	No
ISO15693	`[0x08]`	No

Media type value	Type	Current support
NFC Type 1 Tag	[0x09]	No
NFC DEP media	[0x0A]	No

The notifications can be disabled if not required using the ***Disable media arrival and removal notifications*** command (on page 66).

Media data exchange

To exchanged data with the media, the serial host constructs the data with a CCID header. The CCID header should have a valid slot ID where the data is received. The following table summarises the supported CCID exchanges.

Table 12: Supported CCID exchanges

Message name	Command bMessageType	Response message	Response bMessageType
PC_to_RDR_IccPowerOn	[0x62]	RDR_to_PC_DataBlock	[0x80]
PC_to_RDR_IccPowerOff	[0x63]	RDR_to_PC_SlotStatus	[0x81]
PC_to_RDR_GetSlotStatus	[0x65]	RDR_to_PC_SlotStatus	[0x81]
PC_to_RDR_XfrBlock	[0x6F]	RDR_to_PC_DataBlock	[0x80]
PC_to_RDR_GetParameters	[0x6C]	RDR_to_PC_Parameters	[0x82]
PC_to_RDR_ResetParameters	[0x6D]	RDR_to_PC_Parameters	[0x82]
PC_to_RDR_SetParameters	[0x61]	RDR_to_PC_Parameters	[0x82]

To exchange data with the media, use a PC_to_RDR_XfrBlock message.

Get the ATR of the media

Once media has been detected, the ATR of the media can be retrieved by sending the message, PC_to_RDR_IccPowerOn. This message is sent from the serial host as shown in the following table.

Table 13: PC_to_RDR_IccPowerOn message format

Field	Offset	Size in bytes	Value/Description
bMessageType	0	1	[0x62]
dwLength	1	4	0 (Little endian format)
bSlot	5	1	Destination reader slot ID. Please refer to ***Communication parameters*** (on page 34).
bSeq	6	1	0
bPowerSelect	7	1	0
abRFU	8	2	0

The NFC module respond with a RDR_to_PC_DataBlock message conveying the ATR of the media it has found. If there is an error, then appropriate error message will be conveyed back in bStatus and bError fields and ATR may not be present in abData field.

Table 14: RDR_to_PC_DataBlock message format

Field	Offset	Size in bytes	Value/Description
bMessageType	0	1	[0x80]
dwLength	1	4	Size of abData field (Contains the ATR, if present)
bSlot	5	1	Source Reader slot ID. Please refer to ***Communication parameters*** (on page 34).

Field	Offset	Size in bytes	Value/Description
bSeq	6	1	Same as command message
bStatus	7	1	0
bError	8	1	0
bChainParameter	9	1	0
abData	10	Size of the ATR	ATR of the media, if present

Communicate with the media

To communicate with the media, PC_to_RDR_XfrBlock message is sent from the serial host as shown in Table 17.

Table 15: PC_to_RDR_XfrBlock message

Field	Offset	Size in bytes	Value/Description
bMessageType	0	1	[0x6F]
dwLength	1	4	Size of abData field in little endian format
bSlot	5	1	Destination Reader slot ID. Please refer to ***Communication parameters*** (on page 34).
bSeq	6	1	0
bBWI	7	1	0
wLevelParameter	8	2	0
abData	10	Size of the data to be sent to the media	Contains the data to be sent to the media

The NFC module sends out the data in the abData field to the media. The media processes the data and replies with an appropriate response.

The NFC module receives the media response and communicates back to the serial host by sending a RDR_to_PC_DataBlock message as in Table 17.

Table 16: RDR_to_PC_DataBlock message

Field	Offset	Size in bytes	Value/Description
bMessageType	0	1	[0x80]
dwLength	1	4	Size of abData field in little endian
bSlot	5	1	Source Reader slot ID. Please refer to ***Communication parameters*** (on page 34).
bSeq	6	1	Same as command message
bStatus	7	1	0
bError	8	1	0
bChainParameter	9	1	0
abData	10	Size of the media response	Media response, if present

If PC_to_RDR_XfrBlock message is sent when a media is not present, the reader responds with bStatus and bError fields set to [0x42] and [0xFE] respectively. The field dwLength is also set to 0 and no abData field is present.

MIFARE cards

When the reader detects NFC media, the application gets the ATR of the media. Since there is no ATR present for MIFARE media, the NFC module constructs an ATR from the fixed elements that identify the card.

The ATR for MIFARE media is 20 bytes long. It has fixed values, with the exception of the 15th byte, which indicates the type of MIFARE media. The following table shows the ATRs for different types of MIFARE media.

Table 17: MIFARE media ATR bytes

MIFARE media type	ATR bytes
1K Classic	3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 03 00 00 00 00 68
4K Classic	3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 04 00 00 00 00 6F
Ultralight	3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 05 00 00 00 00 6E

Note that the value of the 15th byte indicates the type of MIFARE media.

The application software can look for these specific ATR bytes to detect MIFARE-type media. Once it detects a MIFARE-type medium, the application can then use the MIFARE commands to communicate with it.

Refer to ***MIFARE media commands and responses*** (on page 39) for details of the MIFARE media commands and responses that you can use.

Contactless microprocessor smartcards

The NFC module detects contactless microprocessor smartcards such as Java cards, ACOS, Desfire, SmartMX cards and most e-Passports. These media have an ATR, which the NFC module retrieves. The host application can send APDU commands to these media using the Windows Smartcard API.

Note: *The format of the command and response APDUs depend on the type of media. Refer to the media's user manual for the command and response formats.*

10. MIFARE media commands and responses

This section describes the MIFARE media commands and responses for the NFC module.

In serial mode, all MIFARE commands use a `CCID PC_to_RDR_XferBlock` message to communicate with the module/card. The application software should be aware of this and it should add/remove the CCID header from the command/responses.

Note: *All of the MIFARE commands for a serial device have an attached CCID header, which is shown in black text in the examples. The header is always 10 bytes in length; the second byte indicates the length of the command or response. The example commands and responses omit trailing zeroes.*

The command bytes have a command code, which is bit encoded as follows:

Bit							
7	6	5	4	3	2	1	0
1 - RF selection prior to command operation 0 - No RF selection	1 - Authentication after RF selection (if enabled) but before the command operation 0 - No authentication*	Command function code					

* Some commands will automatically perform authentication if you enable RF selection.

When you enable RF selection, the reader resets each time it polls.

When you disable RF selection, the reader polls using the Universally Unique Identifier (UUID) that it had last time it polled; the reader looks for the same card.

Note: *The MIFARE command code has two possible values depending on whether the command includes authentication (bit 6). For example, RF select and no authentication: binary =* `10000000`*, hex =* `[0x80]`*, RF select with authentication: binary =* `11000000`*, hex =* `[0xC0]`*. Authentication may or may not apply depending on the command.*

MIFARE get media type

Use this command to return the MIFARE media type.

MIFARE command bytes

MIFARE command bytes	
Command header	**Command code**
`[0x00]`	`[0x00]` or `[0x40]` Get type
	`[0x80]` or `[0xC0]` RF select and get type (No authentication performed)

MIFARE response bytes

MIFARE response bytes				
Response header	**Response code (1)**	**MIFARE type (2)**	**Media UID (2)**	**Status bytes (3)**

`[0x00]`	Any one of the following values
	`[0x01]`
	`[0x41]`
	`[0x81]`
	`[0xC1]`
		`[0x03]`
		MIFARE Classic 1K
		`[0x04]`
		MIFARE Classic 4K
		`[0x05]`
		MIFARE Ultralight
			4 bytes for cascade level 1
			7 bytes for cascade level 2
			10 bytes for cascade level 3
				`[0x90][0x00]`
				Success
				`[0x69][Status Code]`
				Failure

(1) The response code is the command code plus one (for example, command `[0x00]` - response `[0x01]`). **(2)** This field is only present if the command is successful. **(3)** Refer to ***MIFARE failure status codes*** (on page 56) for information on MIFARE failure status codes.

Example

This command successfully retrieves the MIFARE media type (MIFARE Classic 4K) and the UID of the card.

USB

Command: `[0x00][0x00]`

Response: `[0x00][0x01][0x04][0x02][0x0A][0xA8][0x9C][0x90][0x00]`

Serial

Command: `[0x6f][0x02][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]`

Response: `[0x80][0x09][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x01][0x04][0x02][0x0A][0xA8][0x9C][0x90][0x00]`

MIFARE load key

Use this command to load the MIFARE key to access the protected sectors of the MIFARE media. You must execute this command before any operation that involves MIFARE authentication.

Note: *This command format is different from other MIFARE command formats as the block number is not required.*

MIFARE command bytes

MIFARE command bytes		
Command header	**Command code**	**MIFARE key**
`[0x00]`	`[0x02]` or `[0x42]` Load key	6 bytes of MIFARE key
	`[0x82]` or `[0xC2]` RF select and load key (No authentication performed)	

MIFARE response bytes

MIFARE response bytes		
Response header	**Response code**	**Status bytes (1)**

`[0x00]`	Any one of the following values (Command code + 1)
	`[0x03]` or `[0x43]`
	`[0x83]` or `[0xC3]`
		`[0x90][0x00]`
		Success
		`[0x69][Status Code]`
		Failure

(1) Refer to ***MIFARE failure status codes*** (on page 56).

Example

This command successfully loads the 6-byte key into the NFC module for MIFARE authentication.

USB

Command: `[0x00][0x02][0xFF][0xFF][0xFF][0xFF][0xFF][0xFF]`

Response: `[0x00][0x03][0x90][0x00]`

Serial

Command: `[0x6F][0x08][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]`
`[0x00][0x02][0xFF][0xFF][0xFF][0xFF][0xFF][0xFF]`

Response `[0x80][0x04][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]`
`[0x00][0x03][0x90][0x00]`

MIFARE authenticate block (key A or key B)

Use this command to authenticate the specified MIFARE block against the MIFARE media's internal Key A or B.

You must load the MIFARE key using ***MIFARE load key*** (on page 40) before sending this command.

The MIFARE authenticate block command is largely used for test purposes. The other MIFARE commands use this command internally to check that the key is loaded and responds. It checks whether the MIFARE keys are correctly loaded.

Note: *This command is NOT applicable to MIFARE Ultralight cards and fails if executed on Ultralight cards. Ultralight cards do not support the authenticate command.*

MIFARE command bytes

MIFARE command bytes		
Command header	**Command code**	**Block number**
`[0x00]`	`[0x04]` or `[0x44]` Authenticate block (Key A)	Block number
	`[0x14]` or `[0x54]` Authenticate block (Key B)	
	`[0x84]` or `[0xC4]` RF select and authenticate block (Key A)	
	`[0x94]` or `[0xD4]` RF select and authenticate block (Key B)	

MIFARE response bytes

MIFARE response bytes			
Response header	**Response code**	**Block number**	**Status bytes (1)**

`[0x00]`	Any one of the following values (Command code + 1)
	`[0x05]` or `[0x45]`
	`[0x15]` or `[0x55]`
	`[0x85]` or `[0xC5]`
	`[0x95]` or `[0xD5]`
		Block number	`[0x90][0x00]`
			Success
			`[0x69][Status Code]`
			Failure

(1) Refer to ***MIFARE failure status codes*** (on page 56).

Example

This command successfully authenticates block number 0 against Key A in the media.

USB

Command: `[0x00][0x04][0x00]`

Response: `[0x00][0x05][0x00][0x90][0x00]`

Serial

Command: `[0x6F][0x03][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x04][0x00]`

Response `[0x80][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x05][0x00][0x90][0x00]`

MIFARE read block (key A or key B)

Use this command to authenticate the specified MIFARE block against the MIFARE media's internal Key A or B and then read the contents of the block.

You must load the MIFARE key using ***MIFARE load key*** (on page 40) before sending this command.

Note: *This command is NOT applicable to MIFARE Ultralight cards and fails if executed on Ultralight cards. To read Ultralight cards, use the* ***MIFARE Ultralight read block*** *command (on page 49).*

MIFARE command bytes

MIFARE command bytes		
Command header	**Command code**	**Block number**
`[0x00]`	`[0x06]` Read block (Key A)	Block number
	`[0x16]` Read block (Key B)	
	`[0x46]` Authenticate and read block (Key A)	
	`[0x56]` Authenticate and read block (Key B)	
	`[0x86]` or `[0xC6]` RF select, authenticate and read block (Key A)	
	`[0x96]` or `[0xD6]` RF select, authenticate and read block (Key B)	

MIFARE response bytes

MIFARE response bytes				
Response header	**Response code**	**Block number**	**Block data (1)**	**Status bytes (2)**

`[0x00]`	Any one of the following values (Command code + 1)
	`[0x07]` or `[0x17]`
	`[0x47]` or `[0x57]`
	`[0x87]` or `[0xC7]`
	`[0x97]` or `[0xD7]`
		Block number	16 bytes	`[0x90][0x00]`
				Success
				`[0x69][Status Code]`
				Failure

(1) This field is present only if the command is successful. **(2)** Refer to ***MIFARE failure status codes*** (on page 56).

Example

This command successfully reads block number 0, using the loaded key authenticated against Key A in the media.

USB

Command: `[0x00][0x46][0x00]`

Response: `[0x00][0x47][0x00][0x02][0x0A][0xA8][0x9C][0x3C][0x98][0x02][0x00][0x64][0x5D][0x04][0x11][0x5D][0x50][0x44][0x01][0x90][0x00]`

Serial

Command: `[0x6F][0x03][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x46][0x00]`

Response `[0x80][0x15][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x47][0x00][0x02][0x0A][0xA8][0x9C][0x3C][0x98][0x02][0x00][0x64][0x5D][0x04][0x11][0x5D][0x50][0x44][0x01][0x90][0x00]`

MIFARE write block (key A or key B)

Use this command to authenticate the specified MIFARE block against the MIFARE media's internal Key A or B and then write the specified data into that block.

You must load the MIFARE key using ***MIFARE load key*** (on page 40) before sending this command.

Note: *This command is NOT applicable to MIFARE Ultralight cards and fails if executed on Ultralight cards. To write to an Ultralight card use the* ***MIFARE Ultralight write block*** *command (on page 50).*

MIFARE command bytes

MIFARE command bytes			
Command header	**Command code**	**Block number**	**Block data**
`[0x00]`	`[0x08]` Write block (Key A)	Block number	16 bytes
	`[0x18]` Write block (Key B)		
	`[0x48]` Authenticate and write block (Key A)		
	`[0x58]` Authenticate and write block (Key B)		
	`[0x88]` or `[0xC8]` RF select, authenticate and write block (Key A)		
	`[0x98]` or `[0xD8]` RF select, authenticate and write block (Key B)		

MIFARE response bytes

MIFARE response bytes			
Response header	**Response code**	**Block number**	**Status bytes (1)**

`[0x00]`	Any one of the following values
	(Command code + 1)
	`[0x09]` or `[0x19]`
		Block number	`[0x90][0x00]`
			Success
			`[0x69][Status Code]`
			Failure

(1) Refer to ***MIFARE failure status codes*** (on page 56).

Example

This command successfully writes 16 bytes to block number 2, using the loaded key authenticated against Key A in the media.

USB

Command: `[0x00][0x48][0x02][0x01][0x02][0x03][0x04][0x05][0x06][0x07]`
`[0x08][0x09][0x10][0x11][0x12][0x13][0x14][0x15][0x16]`

Response: `[0x00][0x49][0x02][0x90][0x00]`

Serial

Command: `[0x6F][0x13][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]`
`[0x00][0x48][0x02][0x01][0x02][0x03][0x04][0x05][0x06][0x07]`
`[0x08][0x09][0x10][0x11][0x12][0x13][0x14][0x15][0x16]`

Response `[0x80][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]`
`[0x00][0x49][0x02][0x90][0x00]`

MIFARE create value block (key A or key B)

Use this command to authenticate the specified MIFARE block against the MIFARE media's internal Key A or B and then create a value block in that block number. The value block is initialised to the specified 32-bit initial value.

A value block is a normal block reserved for storing numeric data, for example, the number of times data writes to the card. Change a value block back to a normal block using the ***MIFARE write block (key A or key B)*** command (on page 44).

You must load the MIFARE key using ***MIFARE load key*** (on page 40) before sending this command.

Note: *This command is NOT applicable to MIFARE Ultralight cards and fails if executed on Ultralight cards. Ultralight cards do not support value blocks.*

MIFARE command bytes

MIFARE command bytes			
Command header	**Command code**	**Block number**	**Initial value**
`[0x00]`	`[0x0A]` Create value block (Key A)	Block number	32-bit initial value (Most Significant Bit (MSB) first)
	[`0x1A]` Create value block (Key B)		
	`[0x4A]` Authenticate and create value block (Key A)		
	`[0x5A]` Authenticate and create value block (Key B)		
	`[0x8A]` or `[0xCA]` RF select, authenticate and create value block (Key A)		
	`[0x9A]` or `[0xDA]` RF select, authenticate and create value block (Key B)		

MIFARE response bytes

MIFARE response bytes			
Response header	**Response code**	**Block number**	**Status bytes (1)**

`[0x00]`	Any one of the following values
	(Command code + 1)
	`[0x0B]` or `[0x1B]`
	`[0x4B]` or `[0x5B]`
	`[0x8B]` or `[0xCB]`
	`[0x9B]` or `[0xDB]`
		Block number	`[0x90][0x00]`
			Success
			`[0x69][Status Code]`
			Failure

(1) Refer to ***MIFARE failure status codes*** (on page 56).

Example

This command successfully creates a value field in block number 4 and initialises the value to 0x00000001. This command uses the loaded key and authenticates against Key A in the media.

USB

Command: `[0x00][0x4A][0x04][0x00][0x00][0x00][0x01]`

Response: `[0x00][0x4B][0x04][0x90][0x00]`

Serial

Command: `[0x6F][0x07][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x4A][0x04][0x00][0x00][0x00][0x01]`

Response `[0x80][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x4B][0x04][0x90][0x00]`

MIFARE increment value block (key A or key B)

Use this command to authenticate the given MIFARE block against the MIFARE media's internal Key A or B and then increase the value stored in the value block.

You must load the MIFARE key using ***MIFARE load key*** (on page 40) before sending this command. The specified block number must also be a value block, or the command will fail. To create a value block, use the ***MIFARE create value block (key A or key B)*** command (on page 45).

Note: This command is NOT applicable to MIFARE Ultralight cards and fails if executed on Ultralight cards. Ultralight cards do not support value blocks.

MIFARE command bytes

MIFARE command bytes			
Command header	**Command code**	**Block number**	**Initial value**
`[0x00]`	`[0x0C]` Increment value block (Key A)	Block number	32-bit increment value (MSB first)
	`[0x1C]` Increment value block (Key B)		
	`[0x4C]` Authenticate and increment value block (Key A)		
	`[0x5C]` Authenticate and increment value block (Key B)		
	`[0x8C]` or `[0xCC]` RF select, authenticate and increment value block (Key A)		
	`[0x9C]` or `[0xDC]` RF select, authenticate and increment value block (Key B)		

MIFARE response bytes

MIFARE response bytes			
Response header	**Response code**	**Block number**	**Status bytes (1)**

`[0x00]`	Any one of the following values (Command code + 1)
	`[0x0D]` or `[0x1D]`
	`[0x4D]` or `[0x5D]`
	`[0x8D]` or `[0xCD]`
	`[0x9D]` or `[0xDD]`
		Block number	`[0x90][0x00]`
			Success
			`[0x69][Status Code]`
			Failure

(1) Refer to ***MIFARE failure status codes*** (on page 56).

Example

This command successfully increments the previously created value field at block number 4 by [x00000001]. The command uses the loaded key and authenticates against Key A in the media.

USB

Command: `[0x00][0x4C][0x04][0x00][0x00][0x00][0x01]`

Response: `[0x00][0x4D][0x04][0x90][0x00]`

Serial

Command: `[0x6F][0x07][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]`
`[0x00][0x4C][0x04][0x00][0x00][0x00][0x01]`

Response `[0x80][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]`
`[0x00][0x4D][0x04][0x90][0x00]`

MIFARE decrement value block (key A or key B)

Use this command to authenticate the specified MIFARE block against the MIFARE media's internal Key A or B and then decrease the value stored in the value block.

You must load the MIFARE key using ***MIFARE load key*** (on page 40) before sending this command. The specified block number must also be a value block, or the command will fail. To create a value block, use the ***MIFARE create value block (key A or key B)*** command (on page 45).

Note: *This command is NOT applicable to MIFARE Ultralight cards and fails if executed on Ultralight cards. Ultralight cards do not support value blocks.*

MIFARE command bytes

MIFARE command bytes			
Command header	**Command code**	**Block number**	**Initial value**
`[0x00]`	`[0x0E]` Decrement value block (Key A)	Block number	32-bit decrement value (MSB first)
	`[0x1E]` Decrement value block (Key B)		
	`[0x4E]` Authenticate and decrement value block (Key A)		
	`[0x5E]` Authenticate and decrement value block (Key B)		
	`[0x8E]` or `[0xCE]` RF select, authenticate and decrement value block (Key A)		
	`[0x9E]` or `[0xDE]` RF select, authenticate and decrement value block (Key B)		

MIFARE response bytes

MIFARE response bytes			
Response header	**Response code**	**Block number**	**Status bytes (1)**

`[0x00]`	Any one of the following values (Command code + 1)
	`[0x0F]` or `[0x1F]`
	`[0x4F]` or `[0x5F]`
	`[0x8F]` or `[0xCF]`
	`[0x9F]` or `[0xDF]`
		Block number	`[0x90][0x00]`
			Success
			`[0x69][Status Code]`
			Failure

(1) Refer to ***MIFARE failure status codes*** (on page 56).

Example

This command successfully decrements the previously created value field at block number 4 by 0x00000001. The command uses the loaded key and authenticates against Key A in the media.

USB

Command: `[0x00][0x4E][0x04][0x00][0x00][0x00][0x01]`

Response: `[0x00][0x4F][0x04][0x90][0x00]`

Serial

Command: `[0x6F][0x07][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]`
`[0x00][0x4E][0x04][0x00][0x00][0x00][0x01]`

Response `[0x80][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]`
`[0x00][0x4F][0x04][0x90][0x00]`

MIFARE Ultralight read block

Use this command to read the contents of the specified block. The Ultralight card blocks are only 4 bytes long.

Note: *This command is applicable ONLY to MIFARE Ultralight cards and fails if executed on other MIFARE card types.*

MIFARE Ultralight command bytes

MIFARE Ultralight command bytes		
Command header	**Command code**	**Block number**
`[0x00]`	`[0x20]` MIFARE Ultralight read block	Block number

MIFARE Ultralight response bytes

MIFARE Ultralight response bytes				
Response header	**Response code**	**Block number**	**Block data (1)**	**Status bytes (2)**

`[0x00]`	`[0x21]`	Block number	16 bytes
			(4 consecutive block data are retrieved)
				`[0x90][0x00]`
				Success
				`[0x69][Status Code]`
				Failure

(1) This field is present only if the command is successful. **(2)** Refer to ***MIFARE failure status codes*** (on page 56).

Example

This command successfully reads four blocks (block numbers 4, 5, 6 and 7) from the Ultralight MIFARE card.

There is no authentication feature on Ultralight cards.

USB

Command: `[0x00][0x20][0x04]`

Response: `[0x00][0x21][0x04][0x04][0x01][0x02][0x03][0x55][0x66][0x77][0x88][0x05][0x06][0x07][0x08][0x09][0x0A][0x0B][0x0C][0x90][0x00]`

Serial

Command: `[0x6F][0x03][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x20][0x04]`

Response `[0x80][0x15][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x21][0x04][0x04][0x01][0x02][0x03][0x55][0x66][0x77][0x88][0x05][0x06][0x07][0x08][0x09][0x0A][0x0B][0x0C][0x90][0x00]`

MIFARE Ultralight write block

Use this command to write data to the specified block.

Note: *This command is applicable ONLY to MIFARE Ultralight cards and fails if executed on other MIFARE card types.*

MIFARE Ultralight command bytes

MIFARE Ultralight command bytes			
Command header	**Command code**	**Block number**	**Block data**
`[0x00]`	`[0x22]` MIFARE Ultralight write block	Block number	4 bytes of data

MIFARE Ultralight response bytes

MIFARE Ultralight response bytes			
Response header	**Response code**	**Block number**	**Status bytes (1)**

`[0x00]`	`[0x23]`	Block number	`[0x90][0x00]`
			Success
			`[0x69][Status Code]`
			Failure

(1) Refer to ***MIFARE failure status codes*** (on page 56).

Example

This command successfully writes one block (block number 4) with 4 bytes of data.

There is no authentication feature on Ultralight cards.

USB

Command: `[0x00][0x22][0x04]]0x01][0x02][0x03][0x04]`

Response: `[0x00][0x23][0x04][0x90][0x00]`

Serial

Command: `[0x6F][0x07][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]`
`[0x00][0x22][0x04][0x01][0x02][0x03][0x04]`

Response `[0x80][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00]`
`[0x00][0x23][0x04][0x90][0x00]`

MIFARE Ultralight-C authenticate - part 1

Use this command to perform the first part of the MIFARE Ultralight-C authentication.

Note: *This command is applicable ONLY to MIFARE Ultralight-C cards and fails if executed on other MIFARE card types.*

MIFARE Ultralight-C command bytes

MIFARE Ultralight-C command bytes	
Command header	**Command code**
`[0x00]`	`[0x24]` MIFARE Ultralight-C Authenticate part 1

MIFARE Ultralight-C response bytes

MIFARE Ultralight-C response bytes				
Response header	**Response code**	**Block number**	**Response cryptogram(1)**	**Status bytes (2)**
`[0x00]`	`[0x25]`	Ignored	9 bytes starting with `[0xAF]`	`[0x90][0x00]` Success `[0x69][Status Code]` Failure

(1) This field is present only if the command is successful. **(2)** Refer to ***MIFARE failure status codes*** (on page 56).

Note: *Once the NFC module receives a command, it waits for 250 milliseconds for another command to arrive. If no command arrives, it resets the MIFARE card. This interval, the 'Command Wait Time', is configurable using the* ***Set NFC timings*** *command (on page 60).*

The NFC reader resets the MIFARE card after the 'Command Wait Time' has expired, and any authentication done before this event is lost.

This means that you must send the next Authenticate part-2 command after a successful Authenticate part-1 command before the 'Command Wait Time' expires.

If you want to preserve the authentication for a longer period, without changing the 'Command Wait Time', send the ***MIFARE get media type*** *command (on page 39) periodically to keep the session active.*

Example

This command performs authentication (part 1) on an Ultralight-C card.

USB

Command: `[0x00][0x24]`

Response: `[0x00][0x25][0x00][0xAF][0x19][0xA0][0xC9][0xF4][0xC2][0x19][0x16][0x2F][0x90][0x00]`

Serial

Command: `[0x6F][0x02][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x24]`

Response `[0x80][0x0E][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x25][0x00][0xAF][0x19][0xA0][0xC9][0xF4][0xC2][0x19][0x16][0x2F][0x90][0x00]`

This command fails to authenticate with an error 'MIFARE Ultralight-C Authentication Part 1 failed'.

USB

Command: `[0x00][0x24]`

Response: `[0x00][0x25][0x00][0x69][0x8B]`

Serial	
Command:	`[0x6F][0x02][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x24]`
Response	`[0x80][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x25][0x00][0x69][0x8B]`

MIFARE Ultralight-C authenticate - part 2

Use this command to perform the second part of the MIFARE Ultralight-C authentication.

Note: *This command is applicable ONLY to MIFARE Ultralight-C cards and fails if executed on other types of MIFARE cards.*

MIFARE Ultralight-C command bytes

MIFARE Ultralight-C command bytes			
Command header	**Command code**	**Block number**	**Cryptogram bytes**
`[0x00]`	`[0x26]` MIFARE Ultralight-C Authenticate part 2	Ignored Set to `[0x00]`	16 bytes of cryptogram

MIFARE Ultralight-C response bytes

MIFARE Ultralight-C response bytes				
Response header	**Response code**	**Block number**	**Response cryptogram (1)**	**Status bytes (2)**

`[0x00]`	`[0x25]`	Ignored	9 bytes starting with `[0x00]`	`[0x90][0x00]`
				Success
				`[0x69][Status Code]`
				Failure

(1) This field is present only if the command is successful. **(2)** Refer to ***MIFARE failure status codes*** (on page 56).

Note: *Once the NFC module receives a command, it waits for 250 milliseconds for another command to arrive. If no command arrives, it resets the MIFARE card. This interval, the 'Command Wait Time', is configurable using the* ***Set NFC timings*** *command (on page 60). The NFC reader resets the MIFARE card after the 'Command Wait Time' has expired, and any authentication done before this event is lost. This means that you must send the next Authenticate part-2 command after a successful Authenticate part-1 command before the 'Command Wait Time' expires.*

If you want to preserve the authentication for a longer period, without changing the 'Command Wait Time', send the ***MIFARE get media type*** *command (on page 39) periodically to keep the session active.*

Example

This command performs authentication (part 2) on an Ultralight-C card.

USB

Command: `[0x00][0x26][0x00][0x6C][0x29][0x02][0x40][0x6B][0x7C][0x74][0x02][0x5A][0xCE][0x65][0x93][0xD8][0x4E][0x36][0xA1]`

Response: `[0x00][0x27][0x00][0x00][0x21][0x65][0x40][0x23][0xCF][0xD5][0x46][0xEB][0x90][0x00]`

Serial

Command: `[0x6F][0x13][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x26][0x00][0x6C][0x29][0x02][0x40][0x6B][0x7C][0x74][0x02][0x5A][0xCE][0x65][0x93][0xD8][0x4E][0x36][0xA1]`

Response `[0x80][0x0E][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x27][0x00][0x00][0x21][0x65][0x40][0x23][0xCF][0xD5][0x46][0xEB][0x90][0x00]`

This command fails to authenticate with an error 'MIFARE Ultralight-C Authentication Part 2 failed'.

USB

Command: `[0x00][0x26][0x00][0x6C][0x29][0x02][0x40][0x6B][0x7C][0x74][0x02][0x5A][0xCE][0x65][0x93][0xD8][0x4E][0x36][0xA1]`

Response: `[0x00][0x27][0x00][0x69][0x8C]`

Serial

Command: `[0x6F][0x13][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x26][0x00][0x6C][0x29][0x02][0x40][0x6B][0x7C][0x74][0x02][0x5A][0xCE][0x65][0x93][0xD8][0x4E][0x36][0xA1]`

Response `[0x80][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x27][0x00][0x69][0x8C]`

MIFARE transceive direct

Use this command to send commands directly to the MIFARE media.

Warning: *This is for advanced users only and provides low-level access to send and receive raw data. It is applicable to all MIFARE card types. Refer to the MIFARE card datasheet for data specifications.*

MIFARE command bytes

MIFARE command bytes			
Command header	**Command code**	**Block number**	**Data bytes**
`[0x00]`	`[0x28]` MIFARE transceive direct	Ignored	Bytes to send to the MIFARE media

MIFARE response bytes

MIFARE response bytes				
Response header	**Response code**	**Block number**	**Response bytes (1)**	**Status bytes (2)**
`[0x00]`	`[0x29]`	Same as command	Response bytes from the MIFARE media	`[0x90][0x00]` Success `[0x69][Status Code]` Failure

(1) This field is present only if command is successful. **(2)** Refer to ***MIFARE failure status codes*** (on page 56).

Example

This command reads block 0 on an Ultralight-C card.

USB

Command: `[0x00][0x28][0x00][0x30][0x00]`

Response: `[0x00][0x29][0x00][0x04][0xC7][0x64][0x2F][0x00][0x00][0x00][0x00][0x00][0x48][0x00][0x00][0x00][0x00][0x00][0x00][0x90][0x00]`

Serial

Command: `[0x6F][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x28][0x00][0x30][0x00]`

Response `[0x80][0x15][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x29][0x00][0x04][0xC7][0x64][0x2F][0x00][0x00][0x00][0x00][0x00][0x48][0x00][0x00][0x00][0x00][0x00][0x00][0x90][0x00]`

This command fails with an error ‘MIFARE direct transceive failed’.

USB

Command: `[0x00][0x28][0x00][0x30][0x00]`

Response: `[0x00][0x29][0x00][0x69][0x8D]`

Serial

Command: `[0x6F][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x28][0x00][0x30][0x00]`

Response `[0x80][0x05][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x29][0x00][0x69][0x8D]`

MIFARE failure status codes

The following table gives the status codes for MIFARE command failures.

Table 18: MIFARE failure status codes

MIFARE failure code	Failure description
`[0x80]`	Missing parameters
`[0x81]`	Invalid command header; command header is not `[0x00]`
`[0x82]`	Invalid command
`[0x83]`	Authentication failed
`[0x84]`	Read block failed
`[0x85]`	Write block failed
`[0x86]`	Restore value block failed (this is an internal command failure)
`[0x87]`	Create value block failed
`[0x88]`	Increment value block failed
`[0x89]`	Decrement value block failed
`[0x8A]`	Transfer value block failed (this is an internal command failure)
`[0x8B]`	MIFARE Ultralight-C Authentication Part 1 failed
`[0x8C]`	MIFARE Ultralight-C Authentication Part 2 failed
`[0x8D]`	MIFARE direct transceive failed

11. NFC module management interface commands

The NFC module exposes an interface, known as the Management Interface. Your application software uses to manage and configure the NFC module. This section of the manual describes the command set and its responses.

Note: *If there is a response to the command, the command is successful. If the command times out, it has failed.*

USB-connected device:

- The Management Interface sends commands as HID reports.
- The report length is always 64 bytes long, even if the commands are just a few bytes.
- The NFC module ignores unused bytes at the end of the commands, but the recommendation is that you should initialise unused bytes to `[0x00]`. The example commands and responses omit trailing zeroes.

Serial-connected device:

- The NFC module exposes a Management Interface on slot `[0xFF]`.
- The management commands are always less than 64 bytes long, while the responses are always 64 bytes long. Even though the response is just a few bytes, it is always padded with `[0x00]` to make it a 64-byte packet to preserve compatibility across different host interfaces.
- All management commands use a `CCID PC_to_RDR_XferBlock` message to communicate with the module. The application software should be aware of this and it should add/remove the CCID header as required.

  Note: *All commands have an attached CCID header, which is shown in black text in the examples. The example responses omit trailing zeroes.*

Get firmware version

Use this command to retrieve the firmware version on the NFC module.

Management command bytes

Byte	Command/Value	Comments
0	`[0x00]`	Command byte
1–63	`[0x00]`	Unused bytes, set to `[0x00]`

Management response bytes

Byte	Response/Value	Comments
0	`[0x00]`	Command echoed
1	Firmware major version	
2	Firmware minor version	
3–63	`[0x00]`	Ignored (61 bytes)

Example

This command retrieves the firmware version (0112).

USB

Command: `[0x00]`

Response: `[0x00][0x01][0x12]`

Serial

Command: `[0x6F][0x01][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x00]`

Response `[0x80][0x36][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x00][0x01][0x12]`

Get bootloader version

Use this command to retrieve the bootloader version on the NFC module.

Management command bytes

Byte	Command/Value	Comments
0	`[0x01]`	Command byte
1–63	`[0x00]`	Unused bytes, set to `[0x00]`

Management response bytes

Byte	Response/Value	Comments
0	`[0x01]`	Command echoed
1	Bootloader major version	
2	Bootloader minor version	
3–63		Ignored (61 bytes)

Example

This command retrieves the bootloader version (0106).

USB

Command: `[0x01]`

Response: `[0x01][0x01][0x06]`

Serial

Command: `[0x6F][0x01][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x01]`

Response `[0x80][0x36][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x01][0x01][0x06]`

Switch to bootloader

Use this command to switch to the bootloader to load new firmware. The module resets itself in bootloader mode soon after the sending the response.

Warning: *Only use this command when loading new firmware. Using this command at any other time may cause the device to become inoperable.*

Management command bytes

Byte	Command/Value	Comments
0	`[0x02]`	Command byte
1–63	`[0x00]`	Unused bytes, set to `[0x00]`

Management response bytes

Byte	Response/Value	Comments
0	`[0x02]`	Command echoed
1	`[0xAA]`	
2	`[0x55]`	
3–63	`[0x00]`	Ignored (61 bytes)

Example

This command switches to the bootloader and then resets the NFC module.

USB

Command: `[0x02]`

Response: `[0x02][0xAA][0x55]`

Serial

Command: `[0x6F][0x01][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x02]`

Response `[0x80][0x36][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x02][0xAA][0x55]`

Get serial number

Use this command to retrieve the serial number of the NFC module.

Management command bytes

Byte	Command/Value	Comments
0	`[0x03]`	Command byte
1–63	`[0x00]`	Unused bytes, set to `[0x00]`

Management response bytes

Byte	Response/Value	Comments
0	[0x03]	Command echoed
1–20	Serial number of the NFC module	
21–63	[0x00]	Ignored (43 bytes)

Example

This command retrieves the serial number of the NFC module (12345678901234567890).

USB

Command: [0x03]

Response: [0x03] [0x31] [0x32] [0x33] [0x34] [0x35] [0x36] [0x37] [0x38] [0x39] [0x30] [0x31] [0x32] [0x33] [0x34] [0x35] [0x36] [0x37] [0x38] [0x39] [0x30]

Serial

Command: [0x6F] [0x01] [0x00] [0x00] [0x00] [0xFF] [0x00] [0x00] [0x00] [0x00] [0x03]

Response [0x80] [0x36] [0x00] [0x00] [0x00] [0xFF] [0x00] [0x00] [0x00] [0x00] [0x03] [0x31] [0x32] [0x33] [0x34] [0x35] [0x36] [0x37] [0x38] [0x39] [0x30] [0x31] [0x32] [0x33] [0x34] [0x35] [0x36] [0x37] [0x38] [0x39] [0x30]

Set NFC timings

Use this command to set various operating timings for the NFC reader. This is an 11-byte command starting for the command byte.

Warning: *Access-IS optimise the NFC timings for the ATR110 and these values should not need to be changed. The NFC module does not modify a timing value if its value is set to zero (0) when you send the command. Using this command incorrectly may cause the device to become inoperable.*

Management command bytes

Byte	Command/Value	Comments
0	[0x06]	Command byte
1	[0x00]	Reserved for future use, should be set to [0x00] for future compatibility
2	[0x00]	Set NFC timings (sub command code)
3	RF reset time for media polling (Least Significant Bit (LSB))	Default - 100 milliseconds [0x64]
4	RF reset time for media polling (LSB)	Default - 20 milliseconds [0x14]
5	Media warm up time in milliseconds	Default - 0 milliseconds [0x00]
6	RF reset time during MIFARE select (MSB)	Default - 5 milliseconds [0x05]
7	RF reset time during MIFARE select (LSB)	Default - 5 milliseconds [0x05]
8	Media warm up time for MIFARE media	Default - 0 milliseconds [0x00]

Byte	Command/Value	Comments
9	Command waiting time for MIFARE media (MSB)	Default - 250 milliseconds `[0xFA]`
10	Command waiting time for MIFARE media (LSB)	Default - 100 milliseconds `[0x64]`

Management response bytes

Byte	Response/Value	Comments
0	`[0x06]`	Command echoed
1	`[0x90]` or `[0x69]`	`[0x90]` - Success `[0x69]` - Failure
2	`[0x00]`	
3–63		Ignored 61 unused bytes

Example

This command sets the ‘RF reset time for media polling (LSB)’ to 200 milliseconds.

USB

Command: `[0x06][0x00][0x00][0x00][0xC8][0x00][0x00][0x00][0x00][0x00][0xC8]`

Response: `[0x06][0x90][0x00]`

Serial

Command: `[0x6F][0x0B][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x06][0x00][0x00][0x00][0xC8][0x00][0x00][0x00][0x00][0x00][0xC8]`

Response: `[0x80][0x36][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x06][0x90][0x00]`

Get NFC timings

Use this command to get the operating timings of the NFC reader.

Management command bytes

Byte	Command/Value	Comments
0	`[0x06]`	Command byte
1	`[0x00]`	Reserved for future use, should be set to `[0x00]` for future compatibility
2	`[0x80]`	Get NFC timings (sub command code)

Management response bytes

Byte	Response/Value	Comments
0	`[0x06]`	Command echoed
1	`[0x00]`	Same as command byte 1
2	`[0x80]`	Same as command byte 2
3	RF reset time for media polling (LSB)	Default - 0 milliseconds `[0x00]`
4	RF reset time for media polling (LSB)	Default - 100 milliseconds `[0x64]`
5	Media warm up time	Default - 20 milliseconds `[0x14]`
6	RF reset time during MIFARE select (MSB)	Default - 0 milliseconds `[0x00]`
7	RF reset time during MIFARE select (LSB)	Default - 5 milliseconds `[0x05]`
8	Media warm up time for MIFARE media	Default - 5 milliseconds `[0x05]`
9	Command waiting time for MIFARE media (MSB)	Default - 0 milliseconds `[0x00]`
10	Command waiting time for MIFARE media (LSB)	Default - 250 milliseconds `[0xFA]`

Example

This command retrieves the NFC timings from the device.

USB

Command: `[0x06][0x00][0x80]`

Response: `[0x06][0x00][0x80][0x00][0x64][0x14][0x00][0x05][0x05][0x00][0xFA]`

Serial

Command: `[0x6F][0x02][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x06][0x00][0x80]`

Response `[0x80][0x36][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x06][0x00][0x80][0x00][0x64][0x14][0x00][0x05][0x05][0x00][0xFA]`

Enter sleep mode

Use this command to enter sleep mode and switch off the RF field. You may want to switch off the RF field when the device is not in use.

Management command bytes

Byte	Command/Value	Comments
0	`[0x07]`	Command byte
1–63	`[0x00]`	Unused bytes, set to `[0x00]`

Management response bytes

Byte	Response/Value	Comments
0	`[0x07]`	Command echoed
1	`[0x90]` or `[0x69]`	`[0x90]` - Success `[0x69]` - Failure
2	`[0x00]`	
3–63	`[0x00]`	Ignored (61 bytes)

Example

This command sets the device in sleep mode.

USB

Command: `[0x07]`

Response: `[0x07][0x90][0x00]`

Serial

Command: `[0x6F][0x01][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x07]`

Response `[0x80][0x36][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x07][0x90][0x00]`

Exit sleep mode

Use this command to exit from sleep mode, turn RF on and resume normal operation.

Management command bytes

Byte	Command/Value	Comments
0	`[0x09]`	Command byte
1–63	`[0x00]`	Unused bytes, set to `[0x00]`

Management response bytes

Byte	Response/Value	Comments
0	`[0x09]`	Command echoed
1	`[0x90]` or `[0x69]`	`[0x90]` - Success `[0x69]` - Failure
2	`[0x00]`	
3–63	`[0x00]`	Ignored (61 bytes)

Example

This command exits sleep mode.

USB

Command: `[0x09]`

Response: `[0x09][0x90][0x00]`

Serial

Command: `[0x6F][0x01][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x09]`

Response `[0x80][0x36][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x09][0x90][0x00]`

Get NFC kernel version

Use this command to retrieve the NFC kernel version of the NFC module.

Management command bytes

Byte	Command/Value	Comments
0	`[0x0B]`	Command byte
1–63	`[0x00]`	Unused bytes, set to `[0x00]`

Management response bytes

Byte	Response/Value	Comments
0	`[0x0B]`	Command echoed
1	NFC kernel major version	BCD hexadecimal value
2	NFC kernel minor version	BCD hexadecimal value
3–63	`[0x00]`	Ignored (61 bytes)

If both the major and minor kernel versions are `[0x00]`, the NFC reader is not enabled in firmware. If you believe this is in error, contact ***support@access-is.com***.

Get media serial number

Use this command to get the media serial number. The following table shows the data element returned by the reader for different types of media.

Table 19: Media types and data elements returned by TripTick

Media type	Data element
MIFARE 1K, 4K and Ultralight ISO14443-4 type A microprocessor card	Unique Identifier (UID)
ISO14443-4 type B microprocessor card	Pseudo-Unique PICC Identifier (PUPI)

Management command bytes

Byte	Command/Value	Comments
0	`[0x0D]`	Command byte
1–63	`[0x00]`	Unused bytes, set to `[0x00]`

Management response bytes

Byte	Response/Value	Comments
0	`[0x0D]`	Command echoed
1	Media type value	See ***Media type values*** (on page 65)
2	Number of valid bytes to follow	Specifies the number of bytes in the serial number
3–xx	Media serial number	Most significant bit first
xx–63		Ignored (unused bytes)

Example

This command retrieves the serial number from the media. The detected media type is MIFARE Ultralight.

USB

Command: `[0x0D]`

Response: `[0x0D][0x05][0x07][0x04][0xBC][0xB8][0xD2][0x3B][0x3C][0x80]`

Serial

Command: `[0x6F][0x01][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x0D]`

Response `[0x80][0x36][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x0D][0x05][0x07][0x04][0xBC][0xB8][0xD2][0x3B][0x3C][0x80]`

Media type values

The value of byte 2 indicates the media type, as shown in the following table.

Table 20: Media type values for supported (and non-supported) media types

Media type	Type value	Current support
No media present	`[0x00]`	Yes
ISO14443-4 A	`[0x01]`	Yes
ISO14443-4 B	`[0x02]`	Yes
MIFARE Classic 1K	`[0x03]`	Yes
MIFARE Classic 4K	`[0x04]`	Yes
MIFARE Ultralight	`[0x05]`	Yes
MIFARE Plus	`[0x06]`	Yes
Felica media	`[0x07]`	No
ISO15693	`[0x08]`	No
NFC Type 1 Tag	`[0x09]`	No
NFC DEP media	`[0x0A]`	No

Disable media arrival and removal notifications

Use this command to disable the media arrival/removal notifications sent from the NFC module; the NFC module becomes a slave unit. The application software should poll for the card by sending the ***Get media serial number*** command (on page 64).

Note: *This command is applicable only to the serial host interface. This command fails if sent to a USB device.*

Management command bytes

Byte	Command/Value	Comments
0	`[0x0E]`	Command byte
1	Slot ID	As defined in ***Communicating with individual readers*** (on page 34)
2	`[0x01]`	Disables notifications Any other value enables notifications

Management response bytes

Byte	Response/Value	Comments
0	`[0x0E]`	Command echoed
1	`[0x90]` or `[0x69]`	`[0x90]` - Success `[0x69]` - Failure
2	`[0x00]`	
3–63		Ignored (unused bytes)

Example

This command disables notifications from the NFC reader.

Serial

Command: `[0x6F][0x01][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x0E][0x00][0x01]`

Response `[0x80][0x36][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x0E][0x90][0x00]`

Set serial interface baud rate

Use this command to change the serial interface baud rate. If the command succeeds, then subsequent commands are sent using the new baud rate.

Note: *This command is applicable only to the serial host interface. This command fails if sent to a USB device.*

Management command bytes

Byte	Command/Value	Comments
0	`[0x10]`	Command byte
1	`[0x00]` - > 9600 Kbps `[0x01]` - > 19200 Kbps `[0x02]` - > 38400 Kbps `[0x03]` - > 57600 Kbps `[0x04]` - > 115200 Kbps (Default) `[0x05]` - > 153600 Kbps `[0x06]` - > 211200 Kbps `[0x07]` - > 230400 Kbps `[0x08]` - > 460800 Kbps `[0x09]` - > 921600 Kbps	Sets the required baud rate
2	`[0x55]`	Signature bytes
3	`[0x5A]`	
4	`[0xA5]`	
5	`[0xAA]`	

Management response bytes

Byte	Response/Value	Comments
0	`[0x10]`	Command echoed
1	`[0x90]` or `[0x69]`	`[0x90]` - Success, new baud rate accepted `[0x69]` - Failure, no change in baud rate
2	`[0x00]`	
3–63		Ignored (unused bytes)

Example

This command sets the baud rate to 57600 Kbps.

Serial

Command: `[0x6F][0x01][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x10][0x03][0x55][0x5A][0xA5][0xAA]`

Response: `[0x80][0x36][0x00][0x00][0x00][0xFF][0x00][0x00][0x00][0x00][0x10][0x90][0x00]`

A. Model numbers

Use the following diagram to identify a TripTick device.

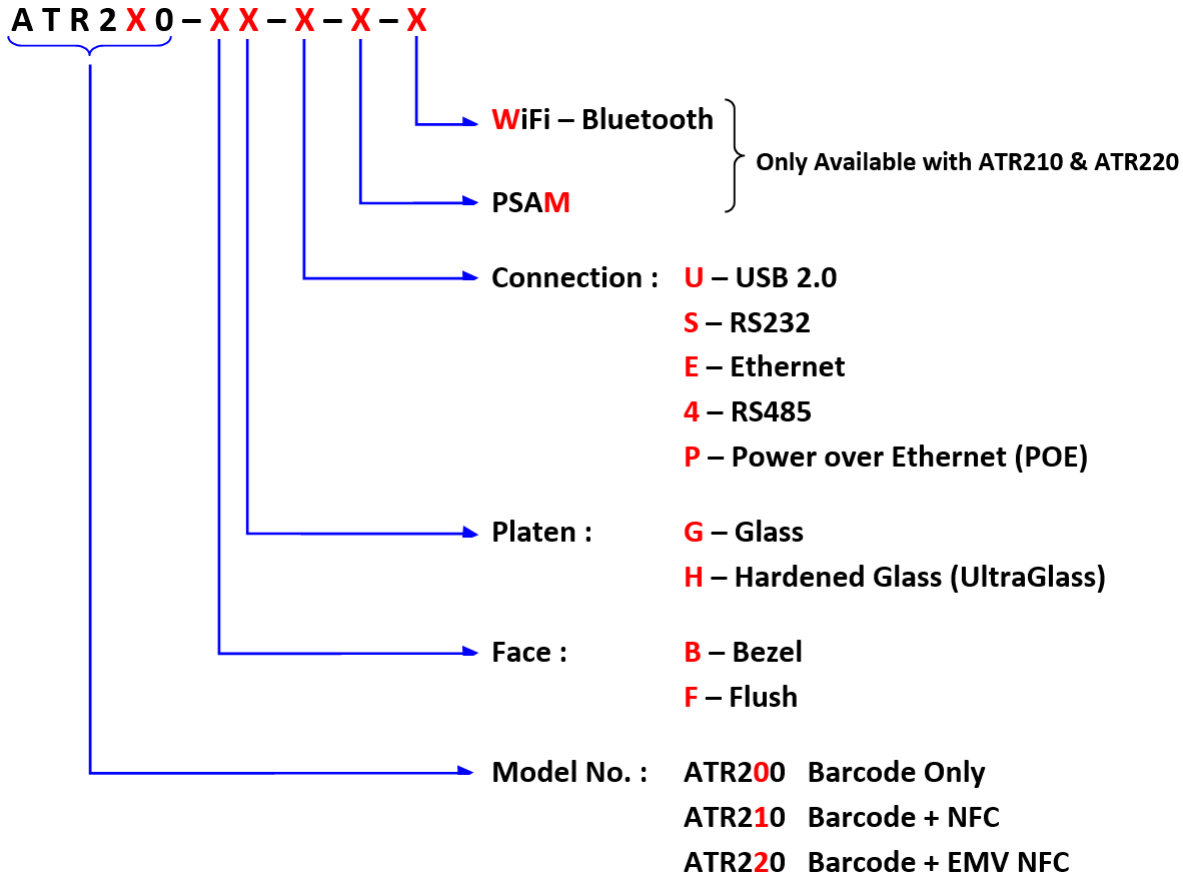

Figure 16: TripTick ordering information

Examples

ATR200–FG–U	Barcode – Flush face – Glass – USB connection.
ATR210–BH–S	Barcode with NFC – Bezel face – Hardened Glass – RS232 connection.
ATR220–FG–E–M–W	Barcode with EMV NFC - Flush face – Glass – Ethernet connection – PSAM – Wi-Fi Bluetooth.

B. Part and host cable numbers

Product	Part number
Input power supply (5V 3AMP IEC I/P VI RATED)	PSU5V3A-VIM

Model	Connection	RFID	EMV	PSAM	Host cable number and description
ATR200-xx-U	USB				5KBD4736 ATR200 USB host cable
ATR200-xx-S	RS232				5KBD4738 ATR200 barcode RS232 serial host cable
ATR200-xx-E	Ethernet				5KBD4777 ATR2x0 power cable
ATR200-xx-4	RS485				5KBD4786 ATR200 barcode RS485 serial host cable
ATR200-xx-P	PoE				No cable supplied
ATR210-xx-U	USB	✓			5KBD4735 ATR2x0 USB & power host cable
ATR210-xx-S	RS232	✓			5KBD4737 ATR2x0 barcode & NFC RS232 serial host cable
ATR210-xx-E	Ethernet	✓			5KBD4777 ATR2x0 power cable
ATR210-xx-4	RS485	✓			5KBD4778 ATR2x0 RS485 barcode & NFC host cable
ATR210-xx-P	PoE	✓			No cable supplied
ATR210-xx-U-M	USB	✓		✓	5KBD4735 ATR2x0 USB & power host cable
ATR210-xx-S-M	RS232	✓		✓	5KBD4737 ATR2x0 barcode & NFC RS232 serial host cable
ATR210-xx-E-M	Ethernet	✓		✓	5KBD4777 ATR2x0 power cable
ATR210-xx-4-M	RS485	✓		✓	5KBD4778 ATR2x0 RS485 barcode & NFC host cable
ATR210-xx-P-M	PoE	✓		✓	No cable supplied
ATR220-xx-U	USB	✓	✓		5KBD4735 ATR2x0 USB & power host cable
ATR220-xx-S	RS232	✓	✓		5KBD4737 ATR2x0 barcode & NFC RS232 serial host cable
ATR220-xx-E	Ethernet	✓	✓		5KBD4777 ATR2x0 Power cable

Model	Connection	RFID	EMV	PSAM	Host cable number and description
ATR220-xx-4	RS485	✓	✓		5KBD4778 ATR2x0 RS485 barcode & NFC host cable
ATR220-xx-P	PoE	✓	✓		No cable supplied
ATR220-xx-U-M	USB	✓	✓	✓	5KBD4735 ATR2x0 USB & power host cable
ATR220-xx-S-M	RS232	✓	✓	✓	5KBD4737 ATR2x0 barcode & NFC RS232 serial host cable
ATR220-xx-E-M	Ethernet	✓	✓	✓	5KBD4777 ATR2x0 Power cable
ATR220-xx-4-M	RS485	✓	✓	✓	5KBD4778 ATR2x0 RS485 barcode & NFC host cable
ATR220-xx-P-M	PoE	✓	✓	✓	No cable supplied

C. NFC module serial number matching

The NFC module exposes a maximum of five CCID smartcard readers and one HID interface for management. It may be possible that a system is connected to two or more modules. In this scenario, the application software should perform serial number matching to determine which readers are physically present together in one module. The interfaces with same serial numbers are physically present together in one module.

Read the serial number of a CCID reader using the **SCardGetAttrib** function.

The following piece of code shows an example of the **SCardGetAttrib** function.

```
// connect to smart card reader
lReturn = SCardConnect( hSC,
                (LPCWSTR)pCardReaderName,
                SCARD_SHARE_DIRECT,
                NULL,
                &hCardHandle,
                NULL );
if ( SCARD_S_SUCCESS != lReturn )
{
    Console::WriteLine("Failed SCardConnect\n");
    exit(1);  // Or other appropriate action.
}

// get reader serial no
LPBYTE   pbAttr = NULL;
DWORD    cByte = SCARD_AUTOALLOCATE;

lReturn = SCardGetAttrib(hCardHandle,
                SCARD_ATTR_VENDOR_IFD_SERIAL_NO,
                (LPBYTE)&pbAttr,
                &cByte);

if ( SCARD_S_SUCCESS != lReturn )
{
    Console::WriteLine("Failed to retrieve Reader Serial\n");
    exit(1);  // Or other appropriate action.
}

printf("serial no: %s", pbAttr);
```

For more information on the **SCardGetAttrib** function, please refer to the Microsoft website. Similar to the CCID readers, you can retrieve the serial number of the HID interface using the **HidD_GetSerialNumberString** function.

Please refer to the Microsoft website for more information on **HidD_GetSerialNumberString**.

D. HID reports - barcode only

Receive data

Data received from TripTick will be in a HID input report, structured as below:

Bit								
Byte	7	6	5	4	3	2	1	0
0	Report ID = 2							
1	Length of data field							
2	AIM symbology Identifier (always ']')							
3	AIM symbology Identifier 1							
4	AIM symbology Identifier 2							
5 60	Data from TripTick (up to 56 bytes)							
61	Further symbology identifier							
62	Reserved							
63	-	-	-	-	-	-	-	Data cont'd

Example HID input reports, as sent by the device

In this example, the decoded barcode contained 60 bytes of data, which the device split into two HID reports. Note that byte 63 in `[0x01]` in the first report and `[0x00]` in the second report indicates whether to expect more data or not. In the second packet, the remaining 52 bytes of data are set to `[0x00]`.

Bit								
Byte	7	6	5	4	3	2	1	0
0	[0x02]							
1	[0x38] - (56)							
2	[0x5D] - ']'							
3	[0x51] - 'Q'							
4	[0x30] - '0'							
5 60	M1GATHERGOOD/M ICHAEL YABCDE F LHRDURAK 354 2 052Y003D23[0x20][0x20]							
61	[0x73] - 's'							
62	[0x00]							
63	0	0	0	0	0	0	0	1

Bit								
Byte	7	6	5	4	3	2	1	0
0	[0x02]							
1	[0x04]							
2	[0x5D] - ']'							
3	[0x51] - 'Q'							
4	[0x30] - '0'							
5	[0x20]100							
..	[0x00] [0x00]							
..	...							
60	[0x00] [0x00]							
61	[0x73] - ‘s’							
62	[0x00]							
63	0	0	0	0	0	0	0	0

Send commands

To send commands to TripTick, use a HID out report with the following structure:

Bit								
Byte	7	6	5	4	3	2	1	0
0	Report ID = 253							
1	Data length							
2	Output data (up to 62 bytes)							
..								
..								
..								
..								
63								

Example output report to request firmware version

This example requests the firmware version from TripTick.

Bit								
Byte	7	6	5	4	3	2	1	0
0	253							
1	[0x0B]							
2	[0x16][0x4D][0x0D]AISFWV1.							
..								
..								
..								
..								
63								

Trigger controls

To set the device status to activate the 'read' lights on the device, send an HID output report with the following structure:

Bit								
Byte	7	6	5	4	3	2	1	0
0	Report ID = 4							
1	-	Activate 'Good Read' light (Green)	Activate 'Bad Read' light (Red)	-	-	Initiate barcode read (trigger)	Prevent barcode read (untrigger)	-

Note: *You can only use 'trigger' and 'untrigger' commands in Interactive mode. 'Good Read' and 'Bad Read' indicator controls are only available in Host or Interactive modes.*

E. NFC module example code and API functions

This section presents code snippets for the NFC module process flow (see Figure 14 on page 33)). The main API functions in are shown in red.

Initialise smartcard sub-system

```
// Try to establish the Smartcard sub-system context
while(SCardEstablishContext(SCARD_SCOPE_SYSTEM, NULL, NULL, &hRFIDContext) !=
SCARD_S_SUCCESS)
{
Sleep(1000); // Wait for some time and retry
}

// Optional: List all the readers available in the smartcard sub-system
// If the reader name is already known then this step is not required. However, there may
// be more than one reader connected to the system. Hence it is recommended to list all the
// readers and select the one that is required
RcvLength = 128;
memset(Reader_Tracker_Buffer, 0 , sizeof(Reader_Tracker_Buffer));
while(SCardListReaders(hRFIDContext, NULL, (LPSTR)_Buffer, &RcvLength) != SCARD_S_SUCCESS)
{
	Sleep(1000); // Wait for some time and retry
```

Poll for card arrival

```
if(SCardGetStatusChange(hRFIDContext, INFINITE, RdrState, 1) == SCARD_S_SUCCESS)
{
	if(RdrState[0].dwEventState & SCARD_STATE_PRESENT)
	{
		MessageBox("Card Present");
	}
	else if(RdrState[0].dwEventState & SCARD_STATE_EMPTY)
	{
		MessageBox("Card Removed");
	}
}
else
{
MessageBox("Reader removed");
}
```

Connect to the card

```
if(SCardConnect(hRFIDContext, _str, SCARD_SHARE_EXCLUSIVE, SCARD_PROTOCOL_T0 |
SCARD_PROTOCOL_T1, &hCrd, &dwProtocol) != SCARD_S_SUCCESS)
{
	// Unable to connect to card
MessageBox("Unable to connect to card");

}
else
{
// Connected to card
MessageBox("Connected to card");
}
```

Get ATR of the card

```
RcvLength = 128;
if(SCardStatus(hCrd, (LPSTR)_Buffer, &RcvLength, NULL, NULL, ATR, &_ATRLen) !=
SCARD_S_SUCCESS)
{
SCardDisconnect(hCrd, SCARD_LEAVE_CARD);
MessageBox("Unable to get ATR of card");
}
else
{
MessageBox("ATR successful");
}
```

Communicate with card

```
RcvLength = RX_BUFFER_SIZE;
if(SCardTransmit(hCrd, &SendPci, TX_Buffer, Transmit_Length, NULL, RX_Buffer, &RcvLength) !=
SCARD_S_SUCCESS)
{
	MessageBox("Communication failed");
}
else
{
	MessageBox("Communication successful");
}
```

Determine if ATR indicates MIFARE type

Refer to ***MIFARE cards*** (on page 38).

Disconnect the card

```
SCardDisconnect(hCrd, SCARD_LEAVE_CARD);
```

F. ASCII character reference

DEC	HEX	Symbol	Description
0	00	NUL	Null char
1	01	SOH	Start of heading
2	02	STX	Start of text
3	03	ETX	End of text
4	04	EOT	End of transmission
5	05	ENQ	Enquiry
6	06	ACK	Acknowledgment
7	07	BEL	Bell
8	08	BS	Back space
9	09	HT	Horizontal tab
10	0A	LF	Line feed
11	0B	VT	Vertical tab
12	0C	FF	Form feed
13	0D	CR	Carriage return
14	0E	SO	Shift out / X-on
15	0F	SI	Shift in / X-off
16	10	DLE	Data line escape
17	11	DC1	Device control 1 (oft. XON)
18	12	DC2	Device control 2
19	13	DC3	Device control 3 (oft. XOFF)
20	14	DC4	Device control 4
21	15	NAK	Negative acknowledgement
22	16	SYN	Synchronous idle
23	17	ETB	End of transmit block
24	18	CAN	Cancel
25	19	EM	End of medium
26	1A	SUB	Substitute
27	1B	ESC	Escape
28	1C	FS	File separator
29	1D	GS	Group separator
30	1E	RS	Record separator
31	1F	US	Unit separator
32	20	SPACE	Space
33	21	!	Exclamation mark
34	22	"	Double quotes (or speech marks)
35	23	#	Number
36	24	$	Dollar
37	25	%	Percent sign
38	26	&	Ampersand
39	27	'	Single quote
40	28	(	Open parenthesis (or open bracket)
41	29	)	Close parenthesis (or close bracket)
42	2A	*	Asterisk

DEC	HEX	Symbol	Description
43	2B	+	Plus
44	2C	,	Comma
45	2D	-	Hyphen
46	2E	.	Period, dot or full stop
47	2F	/	Slash or divide
48	30	0	Zero
49	31	1	One
50	32	2	Two
51	33	3	Three
52	34	4	Four
53	35	5	Five
54	36	6	Six
55	37	7	Seven
56	38	8	Eight
57	39	9	Nine
58	3A	:	Colon
59	3B	;	Semicolon
60	3C	<	Less than (or open angled bracket)
61	3D	=	Equals
62	3E	>	Greater than (or close angled bracket)
63	3F	?	Question mark
64	40	@	At symbol
65	41	A	Uppercase A
66	42	B	Uppercase B
67	43	C	Uppercase C
68	44	D	Uppercase D
69	45	E	Uppercase E
70	46	F	Uppercase F
71	47	G	Uppercase G
72	48	H	Uppercase H
73	49	I	Uppercase I
74	4A	J	Uppercase J
75	4B	K	Uppercase K
76	4C	L	Uppercase L
77	4D	M	Uppercase M
78	4E	N	Uppercase N
79	4F	O	Uppercase O
80	50	P	Uppercase P
81	51	Q	Uppercase Q
82	52	R	Uppercase R
83	53	S	Uppercase S
84	54	T	Uppercase T
85	55	U	Uppercase U
86	56	V	Uppercase V
87	57	W	Uppercase W
88	58	X	Uppercase X

DEC	HEX	Symbol	Description
89	59	Y	Uppercase Y
90	5A	Z	Uppercase Z
91	5B	[	Opening bracket
92	5C	\	Backslash
93	5D	]	Closing bracket
94	5E	^	Caret - circumflex
95	5F	_	Underscore
96	60	`	Grave accent
97	61	a	Lowercase a
98	62	b	Lowercase b
99	63	c	Lowercase c
100	64	d	Lowercase d
101	65	e	Lowercase e
102	66	f	Lowercase f
103	67	g	Lowercase g
104	68	h	Lowercase h
105	69	i	Lowercase i
106	6A	j	Lowercase j
107	6B	k	Lowercase k
108	6C	l	Lowercase l
109	6D	m	Lowercase m
110	6E	n	Lowercase n
111	6F	o	Lowercase o
112	70	p	Lowercase p
113	71	q	Lowercase q
114	72	r	Lowercase r
115	73	s	Lowercase s
116	74	t	Lowercase t
117	75	u	Lowercase u
118	76	v	Lowercase v
119	77	w	Lowercase w
120	78	x	Lowercase x
121	79	y	Lowercase y
122	7A	z	Lowercase z
123	7B	{	Opening brace
124	7C	|	Vertical bar
125	7D	}	Closing brace
126	7E	~	Tilde
127	7F	DEL	Delete

G. Document history

Issue	Date	Description
1.0	26.11.2018	First issue.
1.1	22.07.2019	Added Ethernet commands and configurations sections: Barcode module installation (Ethernet device), NFC module installation (Ethernet device) and Network configuration.
1.2	05.10.2020	Added PoE Ethernet port/power injector specifications to Specifications section
1.3	11.11.2020	Added PSAM reading information to Specifications section. Added SMC reader to the NFC operation section. Added PSAM variants to the Part and host cable numbers section.
1.4	16.02.2021	Changed power supply part number and specification.