AcuPad-50 & AcuPad-50 MUX

User Guide

Copyright

Trademarks

Contacts

For technical support, please contact support@acura.com.br

What's new

Date	Description	Revision
December 2023	Initial release.	A.0

A complete list of revisions is available in Revision history.

Related documents

-

Contents

1 Section – Introduction

1.1 Introduction

The AcuPad-50 is a table or wall reader designed with UHF (Ultra High Frequency) RFID technology, with small dimensions and two RF ports for external antennas. The AcuPad-50 MUX differs only on the output RF power from the AcuPad-50, 0.5W and 1W respectively.

1.2 Applications

Below are the main applications (non-restricted) where the AcuPad-50 can be used for.

- Self-checkout.
- POS checkout.
- Laundry identification.
- Tag read/write stations.
- Production line points identification.
- Small portals with two antennas.

1.3 Main features

The reader has two modes, Autonomous, and Transparent.

- ***Autonomous Mode:***
  - Automatically reads tags in a continuous loop.
  - Turn off/on the RF via command.
  - Individual power control for each antenna.
  - Active individually which antenna to transmit RF.
  - Sends the results of the tags reads in the following interfaces:
    - Via keyboard emulation (HID class).
    - Via serial communication (CDC class).
    - Via keyboard emulation + serial communication.
  - Various configuration parameters via serial terminal.
    - Reading power (individual for each antenna).
    - RF period times.
    - Gen2 parameters, (tag >> reader) and (reader << tag).
    - Reader information.
    - Read operations.
    - Read filter by RSSI value of the tag.
  - Can decode tags with **GS1 SGTIN-96** and **SGTIN-198** standards for checkout applications, sends in the following formats:
    - gs1string, e.g., "(01)03608449920322(21)1540341".
    - gs1epcuri, e.g., "urn:epc:tag:sgtin-96:1.360844.0992032.1540341".
    - gs1epcpureuri, e.g., "urn:epc:id:sgtin:360844.0992032.1540341".
    - gs1gtin13, e.g., "3608449920322".

- gs1sgtin13, e.g., "3608449920322 1540341".

- Can decode tags with the Acura's Autoid System standard for registering tags for vehicle identification application:
  - autoid + autoidcs
    - wiegand26, e.g., "172 13259".
    - clock and data, e.g., "00001034695627".
- Can decode tags with ASCII standard.
- Can convert from hexadecimal to decimal with configurable range.
- Can split the tag EPC identifier with a configurable range.
- Option to not decode and send the tag EPC hexadecimal raw value.
- Can read any memory bank of the tag as additional information to the tag EPC identifier.

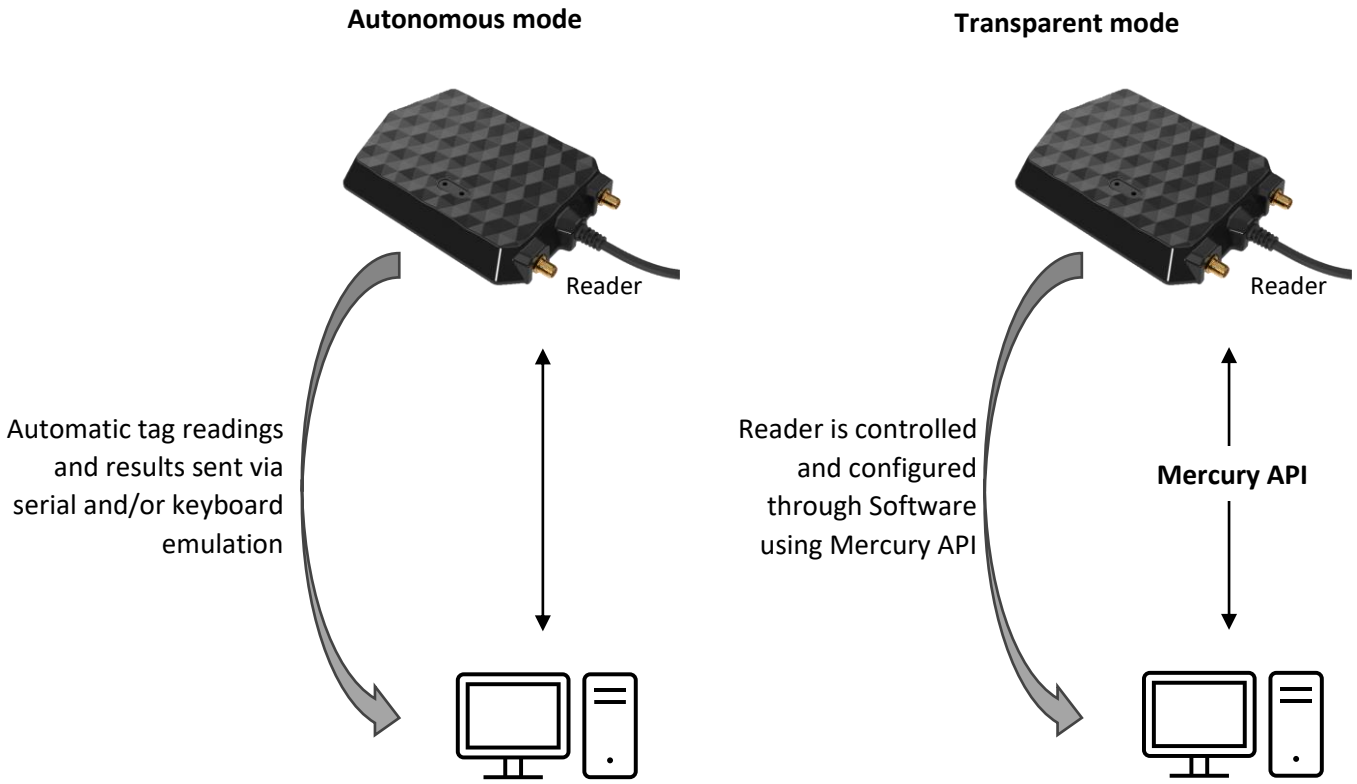
- ***Transparent Mode**:*
  - The reader stays idle, does not perform autonomous tag readings.
  - The reader must be controlled by a Software that uses the mercuryapi, a complete API/SDK for reading and writing tags supporting the following languages:
    - C, .Net (C#) and Java.
  - Compatible with Windows 10, Mac and Linux without driver required, for Windows 7 and 8 the driver is provided for installation.
  - Wide variation of reading power:
    - 0 to 27dBm for AcuPad-50 MUX. (Configurable)
    - 0 to 30dBm for AcuPad-50. (Configurable)

Interfaces

The AcuPad-50 family readers only support USB interface. They do not need an external power supply, the power transmitted through the USB cable from a host Computer is sufficient to power the readers.

HID (keyboard) and Serial modes are enabled by default. When you connect the reader USB cable to a host PC, HID will be detected under the **Human Interface Devices** section and the CDC port will be detected under the **Ports** section of the Device Manager (Windows OS). The readers can be configured to either HID mode or Serial mode or both.

Operating modes



1.4 Products covered by this document

Model	Code
AcuPad-50	Please consult.
AcuPad-50 MUX	Please consult.

1.5 Approval notes

Anatel (Brazil)

AcuPad-50 and AcuPad-50 MUX were tested and approved under the Regulation for Certification and Homologation of telecommunications Products, approved by Anatel Resolution No. 242 of November 30, 2000.

Types: Radio Frequency Identification Systems - Category II.

Service/Application: Restricted Radiation Radiocommunication.

FCC (US)

Federal Communication Commission Interference Statement (FCC)

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operating in a commercial environment. This equipment generates, uses, and can radiate radiofrequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

FCC Caution:	Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.

MPE Warning:	Radiation Exposure Statement - This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 21cm between the radiator & your body. Contains FCCID: QV5MERCURY6EN (AcuPad-50 Mux).

Attention:	The maximum output power available at the device’s RF ports is fixed at factory to be less than or equal to that permitted by the FCC equipment authorization.

Authorized antennas

Brand	MTI Wireless Edge
Model	MT009S08LH/C with 5m cable
Frequency Range (MHz)	902-928 MHz
Peak Gain w/o cable loss	6.5 dBi
Cable loss	3.1 dB
Peak Gain with cable loss	3.4 dBi
Polarization	Circular (LHCP)
Antenna Type	RFID UHF Antenna
Cable connector type	Antenna: SMA male Reader: RP-SMA male

Brand	MTI Wireless Edge
Model	MT-262006/TRH/A/K/F with 1.5m cable
Frequency Range (MHz)	902-928 MHz
Peak Gain w/o cable loss	7 dBi
Cable loss	0.9 dB
Peak Gain with cable loss	6.1 dBi
Polarization	Circular (RHCP)
Antenna Type	RFID UHF Antenna
Cable connector type	Antenna: RP-TNC male Reader: RP-SMA male

Professional installation instruction

- Installation personal
  - The product is designed for specific applications and needs to be installed by a qualified personal who has RF and related rule knowledge. The general user should not attempt to install or change the settings.
- Installation location
  - The product shall be installed at a location where the radiating antenna can be kept 21cm from nearby person in normal operation condition to meet regulatory RF exposure requirement.
- External antenna
  - Use only the antennas and cables which have been approved by the applicant. The non approved antennas and cables may produce unwanted spurious or excessive RF transmitting power which may lead to the violation of FCC limit and is prohibited.
- Installation procedure
  - Please refer to this User Guide for details.

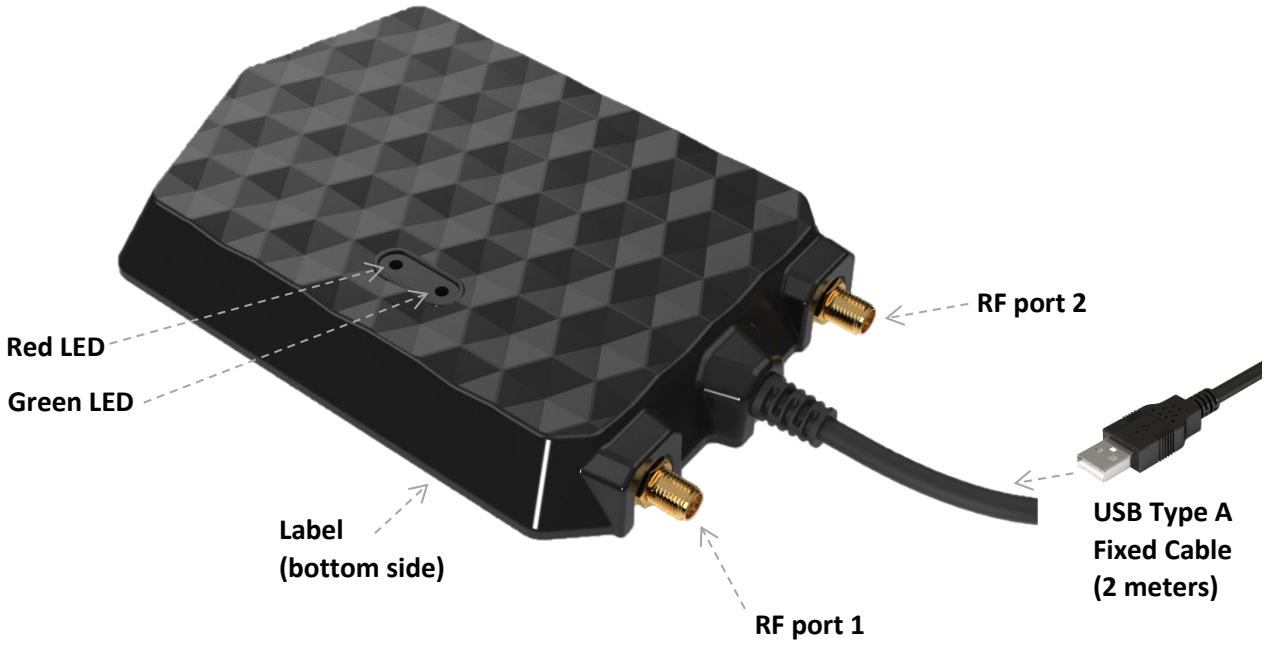
- Warning
    - Please carefully select the installation position and make sure that the final output power does not exceed the limit set in relevant rules. The violation of the rule could lead to serious federal penalties.

Professional installer can only use the antenna + cable combination provided with the EUT; more details for specs please refer to test report.

2 Section – Overview and Installation

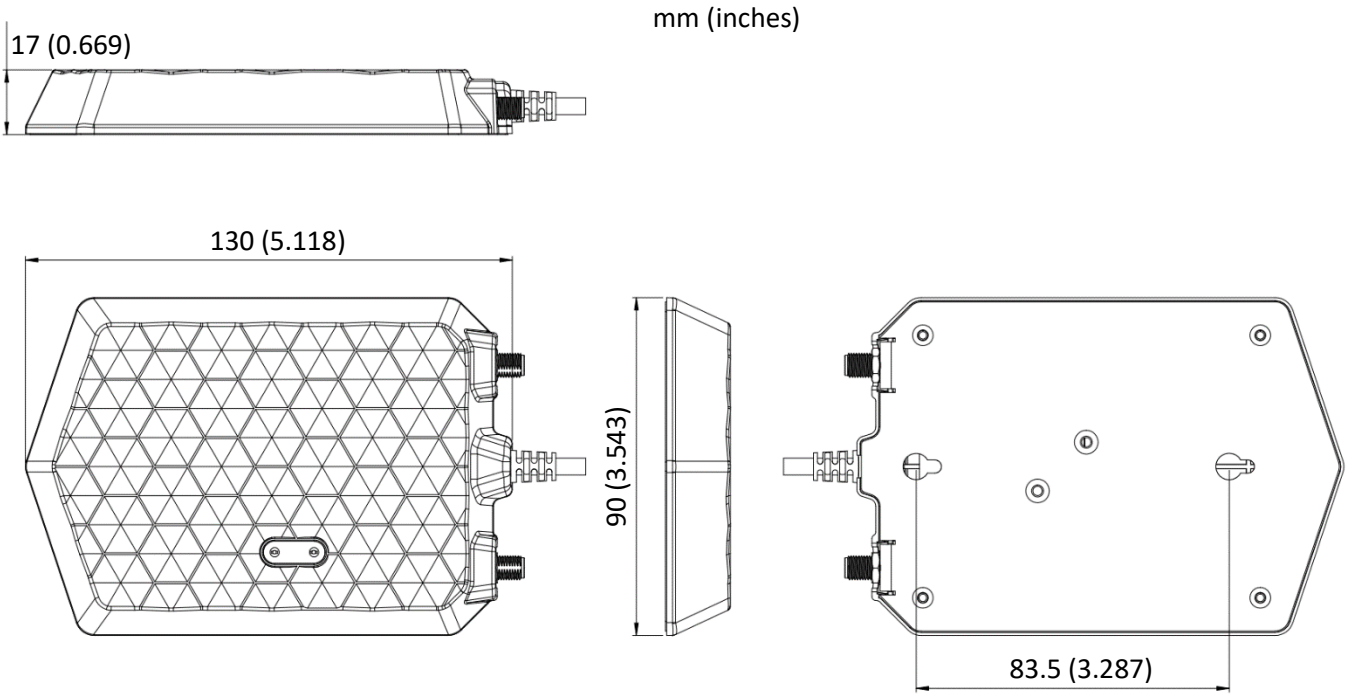
2.1 AcuPad-50 / AcuPad-50 MUX hardware

Overview

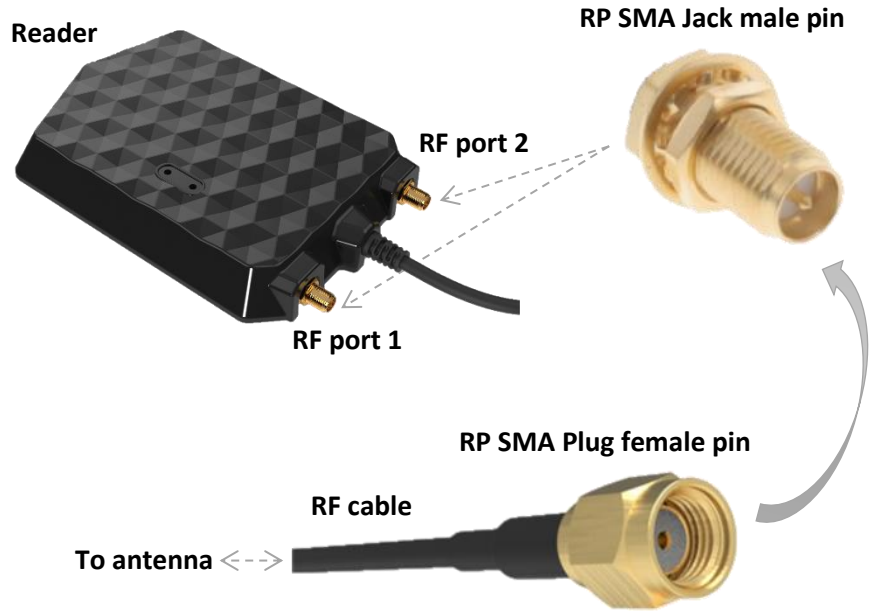


Dimensions

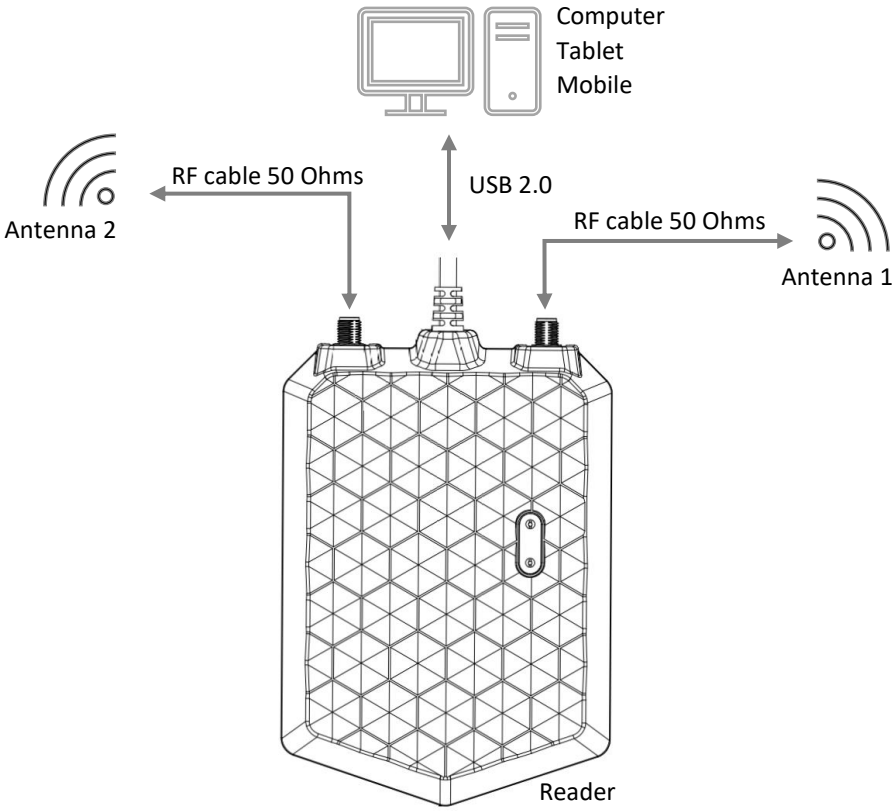
mm (inches)



RF connectors



Connection diagram



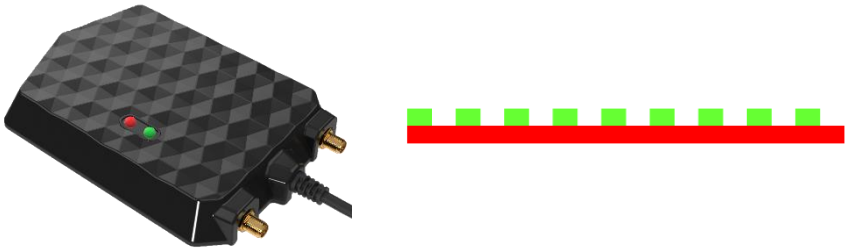
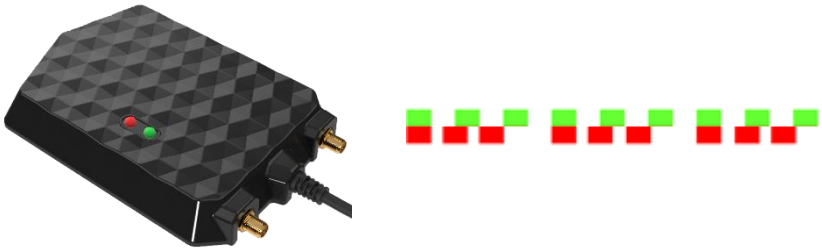
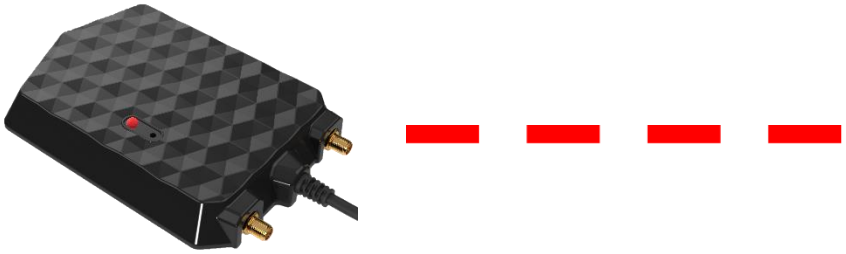
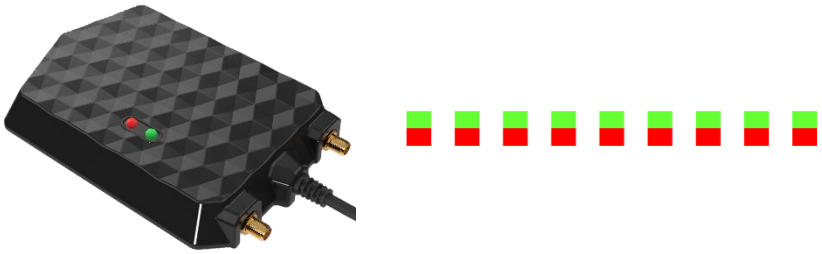
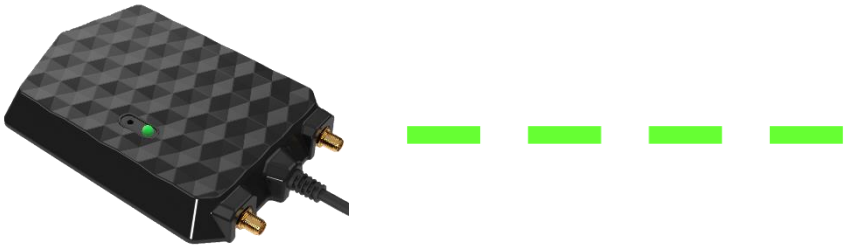
2.2 Specification

Transponder Protocol	
Protocol	ISO 18000-6C (EPC Gen2) ISO 18000-63 (EPC Gen2 V2) (Transparent Mode)
RF interface	
RF Output power	AcuPad-50 MUX: From 0 to 27 dBm with 0.5 dBm increment (Configurable) AcuPad-50: From 0 to 30 dBm with 0.5 dBm increment (Configurable)
Regulation	ANATEL (BR) 902 – 907MHz and 915 – 928MHz FCC (USA) 902 – 928MHz
Mode	Frequency Hopping
Modulation/RF Coding	AcuPad-50 MUX: PR-ASK/M4 AcuPad-50: AM
Backscatter Link Frequency (BLF)	AcuPad-50 MUX: 250KHz AcuPad-50: 160KHz
Performance	
Reading distance (the reading distance can vary depending on the antenna, rf cable, tag model, usage mode, and environment)	Depending on the setup, can vary from 3 to 12 meters (9 to 39 ft).
Data/Control Interface	
Connectors	Communication and Power: USB Type-A male connector
Communication interface	**USB HID keyboard**: The reader is recognized by the Operating System (Windows, Mac and Linux) as a HID device. **USB Serial CDC**: When the reader is connected via USB, a serial port is automatically created on Windows 10, Mac, and Linux operating systems. (For Windows 7 and 8, driver installation is required).
Programming	**Autonomous mode:** The reader can be configured via serial communication using ASCII commands through a terminal application (such as Putty, Minicom, or Screen, etc) without the need to use any API or SDK. It supports tag read-only functionality. **Transparent mode:** The reader responds to software using the Mercury API and has tag read and write functionality.
Energy	
Power	5 VDC +/- 1% (Host USB port). Max ripple: 200 mVpp @ 20 MHz
Consumption Power (defined for operation into a 17dB return loss load or better)	AcuPad-50 MUX: max. 3,8 W @+27 dBm RF transmit power AcuPad-50: max. 6W @+30 dBm RF transmit power

Physical characteristics	
IP rating	Internal use
RF connector	2x RP SMA Jack male pin
Dimension (H x W x D)	(130 x 90 x 17) mm (5.118 x 3.543 x 0.669) inches
Weight	200 g / 7.05 oz / 0.44 lb
Cable length	Fixed cable, 2 meters (6.5 ft)
Operation temperature	-10°C to 55°C (14°F to 131°F)
Storage temperature	-10°C to 60°C (14°F to 140°F)
Humidity	95%
Mounting type	Desktop/Table or Wall mount

2.3 Turning on the reader

After connecting the reader to a host Computer USB port, the LED status will exhibit the following combinations:

	**Green LED: flashes fast.** **Red LED: solid on.** Reader: in Autonomous Mode. Reading tags: activated (RF on).
	**Green LED: flashes fast.** **Red LED: flashes asynchronous with green.** Reader: in Autonomous Mode. No antenna activated to transmit RF.
	**Green LED: turned off.** **Red LED: flashing at a consistent interval.** Reader: in Autonomous Mode. Reading tags: deactivated (RF idle).
	**Green and Red LED: flashing at same time in a consistent interval.** Reader is powered on but with no USB communication with the Computer host.
	**Green LED: flashing at a consistent interval.** **Red LED: turned off.** Reader: in Transparent Mode. Note: In Transparent Mode, the red LED and beep are controlled by the user's software.

On computers running Windows Operating System prior to 10, a driver installation is required for the creation of the COMxx serial communication port. Please contact our support channel support@acura.com.br for more information.

For Linux, Android, and Mac Operating Systems, the driver is not required. When the reader is turned on, the following devices will be created in the Operating System:

- HID class type keyboard.
- Serial port, (COM&LPT on Windows), (CDC on Linux, Android, and Mac).

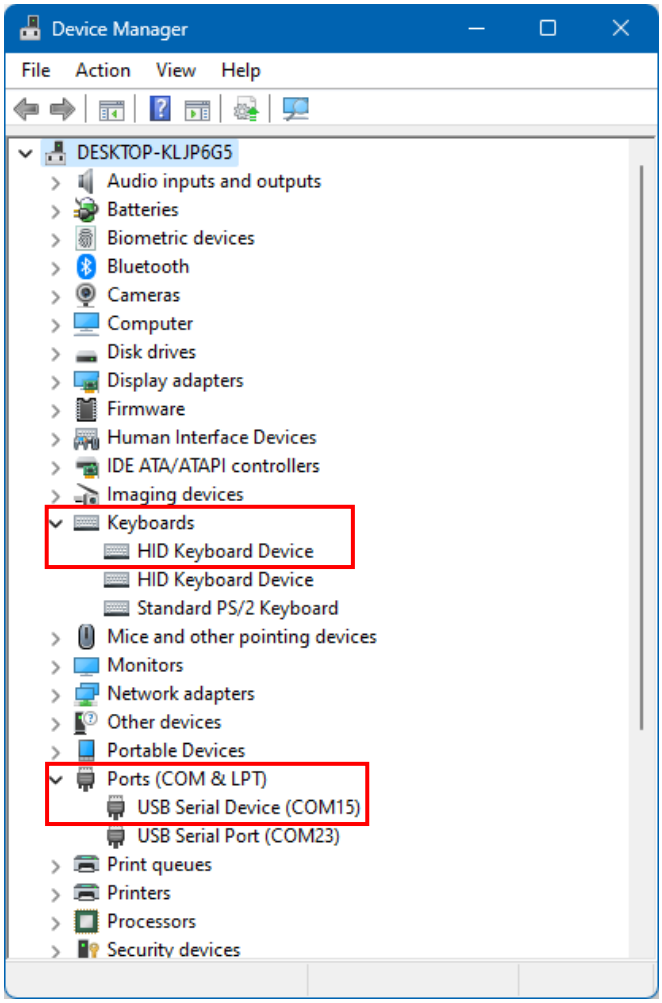
Since the reader uses the HID class keyboard type for keyboard emulation when sending reading results in Autonomous Mode, the Serial Port will be used for communication and configuration in both Autonomous and Transparent modes.

Identifying serial port (Windows)

After turning on the reader, go to the "Control Panel" or with the "Windows + X" key open the "***Device Manager***". In "***Ports (COM and LPT)***", a virtual serial communication port should appear 4s after the time the reader is connected, in addition to a new HID Keyboard type device in "***Keyboards***".

The images below show that the reader has been recognized by the Operating System and is ready to work.

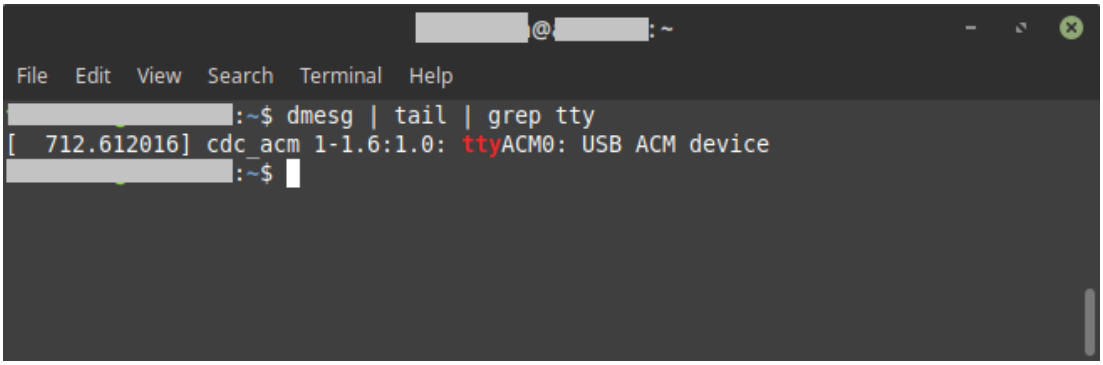
Windows 10/11 - OK



Identifying serial port (Linux)

To identify the name of the serial port created after connecting the reader to USB, type the following in the Terminal:

$dmesg | tail | grep tty



In the example above, serial communication with the reader will be through **/dev/ttyACM0**.

For some Linux distribution like Ubuntu, Mint, Fedora, OpenSuse, etc. it is recommended to disable the ModemManager service to not try to connect to the serial port every time the reader is connect to the USB port:

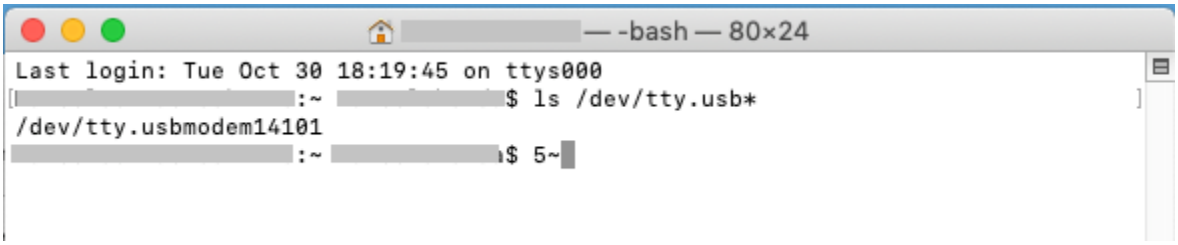
$sudo systemctl stop ModemManager.service

$sudo systemctl disable ModemManager.service

Identifying serial port (Mac OS)

To identify the serial port name created after connecting the reader to USB, type the following in the Terminal:

$ls /dev/tty.usb*



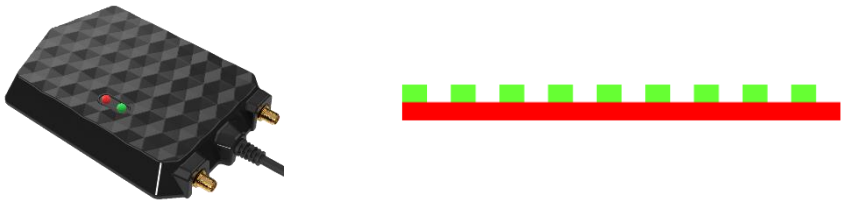
In the example above, serial communication with the reader will be via **/dev/tty.usbmodem14101**.

3 Section – Reader operating modes

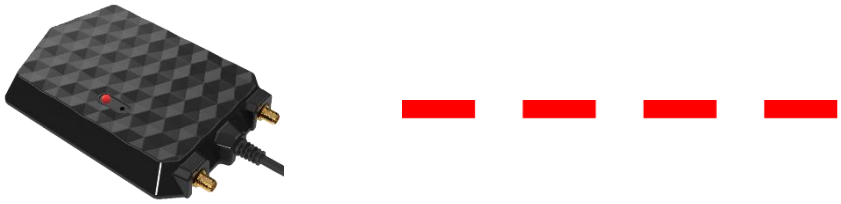
3.1 Reader in Autonomous Mode

In Autonomous Mode, the reader continuously attempts to read a tag, and upon successfully reading a tag, it transmits the information through the configured communication interface. Starting from firmware version 1.1.3, the option to disable tag reading has been introduced. This can be achieved by deactivating the RF signal using the "**readtag**" command or the CapsLock key, effectively putting the reader into an idle state until tag reading is reactivated. The "**antennaport**" command must be used to indicate which antenna will transmit RF.

Visual feedback from the reader when it's in **Autonomous Mode and performing tag reading**:

	**Green LED: flashes fast.** **Red LED: solid on.**

Visual feedback from the reader when you are in the **Autonomous Mode not performing tag reading**:

	**Green LED: turned off.** **Red LED: flashes slowly.**

Turning Read tag ON and OFF in Autonomous Mode

To turn reading tag ON and OFF, the following options can be used:

1. Using the **readtag command**. For example
    a. The command "**readtag off<CR>**" disables the RF (Radio Frequency) and stops reading tags and results report.
    b. The command "**readtag on<CR>**" activates the RF (Radio Frequency) and resumes tag reading and results report.

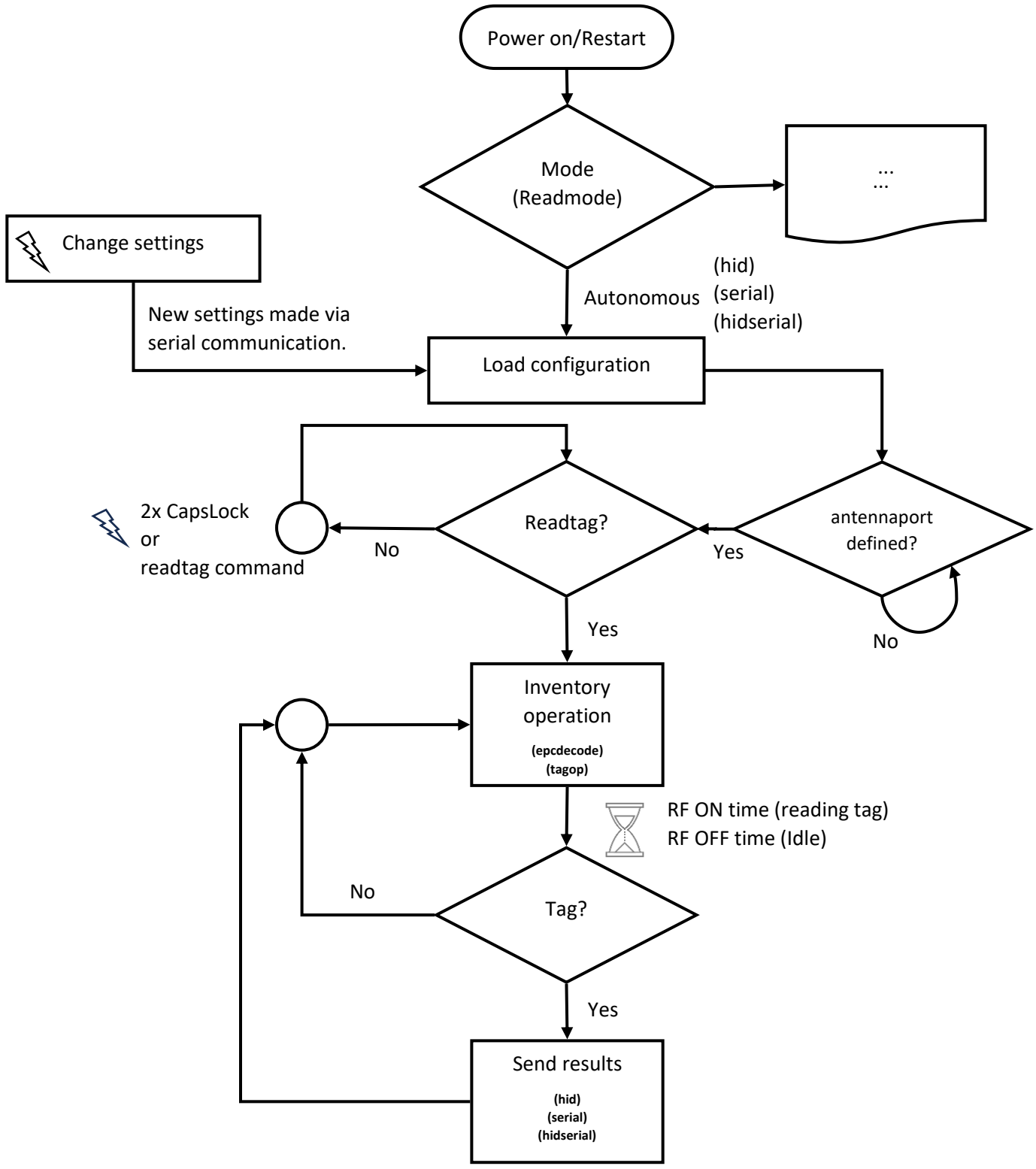
NOTE: <CR> indicates a carriage return.

2. Using the **Caps Lock key** from the keyboard (Windows and Linux only). For example:

To switch between ON and OFF for reading tags, simply ***quickly click the Caps Lock key 2 times in sequence*** and check the reader's LEDs and beep to see the change.

NOTE: The command **enablecapslock** enables or disables this functionality, to use it the **enablecapslock** must be on.

General Diagram



Reading tags engine

The algorithm for reading tags in Autonomous mode is described as follows:

- The tag reading result string format is fixed, not configurable, explained in next section.
- The EPC id of the tag will always be reported by the reader in a format determined by the **epcdecode** command.
- Additionally, to the EPC id, a tag memory bank read operation, configured by the **tagop** command, will be reported by the reader concatenated with a configurable **separator** character.
- More tag reading data can be included in the tag reading result report, such as **readcount** and signal level **rssi**, separated by the **separator** character.

If the interface chosen by the **readmode** command is serial and a tag is read while there is no connection in the serial port generated, the reader will generate audible feedback with fast sound beeps every time the tag is read.

Reading result format

In Autonomous Mode, the reader transmits a fixed formatted string containing the results of tag reads via the selected communication interface.

The format of the string is as follows:

<epcdecode><separator><tagop><separator><readcount><separator><rssi><endofline>
<epcdecode> EPC id read result, decoded as specified by the **epcdecode** command. *Note: Always reported by the reader in Autonomous Mode.*
<separator> Separator character specified by the **separator** command. *Note: It is only reported by the reader if there are more fields in addition to <epcdecode> to report.*
<tagop> The result of the reading operation specified by the **tagop** command. *Note: Optional.*
<readcount> The quantity of readings for the same tag during an inventory period. The choice to display this value or not is determined by the **reportreadcount** command. *Note: Optional.*
<rssi> The RF signal level value in dBm and negative of tag response. The decision to display or hide this value is determined by the **reportrssi** command. *Note: Optional.*
<endofline> The end-of-line type is configured using the **endofline** command. *Note: The reader reports this in Autonomous Mode if it is configured as anything other than "none".*

NOTE: The size of the result string may vary depending on the tags read. The Software that will receive this string, must always handle it using the <separator> and <endofline> fields to split the desired fields.

Configuring the reader (Autonomous Mode)

When the reader is in Autonomous Mode, the following configurations can be made:

- Type of communication
- Perform tag reading or stay idle;
- Reading power, individual for each RF port;
- Antenna port to transmit RF;
- Reading time;
- Idle time;
- EPC id tag decoding type;
- Reading operation in the tag's memory bank;
- EPC Gen2 protocol parameters:
    - BLF (Backscatter Link Frequency);
    - Tari;
    - Tag Encoding;
    - Session;
    - Target;
    - Q (Anti-collision);
    - Access password;
- Character that separates the fields in the result string;
- Type of end-of-line in the result string;
- Typing speed for keyboard emulation;
- Filter by RSSI (Received Signal Strength Indicator);
- Show or do not show the RSSI value of the read tag;
- Show or do not show the quantity of tag reads;
- Audible tag reading alert (beep);
- Check internal temperature, status, and firmware versions.

How to set up the reader?

The reader in Autonomous Mode must be configured using ASCII commands sent through the serial port created by the operating system when the reader is powered on.

NOTE: To find out on which serial port the reader will respond, please consult **Turning on the reader**.

Operating logic

The configuration of the reader via ASCII commands works with the Message/Response structure:

A. An external host sends a command to the reader.
B. The reader interprets and executes the command included in the received message.
C. The reader responds to the external host.

The messages and responses use ASCII characters with a carriage return terminator **<CR>** (13d) (0x0D), which is the [ENTER] in terminal. The messages can be in uppercase, lowercase, or both.

NOTE: The reader echoes each character typed in the serial terminal, and with each new line, the reader sends the character "**>**" to indicate the cursor.

Syntax of messages and responses

Below is the syntax of messages from the Host to the Reader:

COMMAND VALUE<CR>	Changes the value of the command/parameter.
COMMAND<CR>	Reads the value of the command/parameter.

NOTE: Only one value must be passed for each command, the reader will not accept more than one value per command.

Below is the syntax of reader response messages to host:

RESULT<CR>	Returns the result of the execution of the last command sent to the reader.

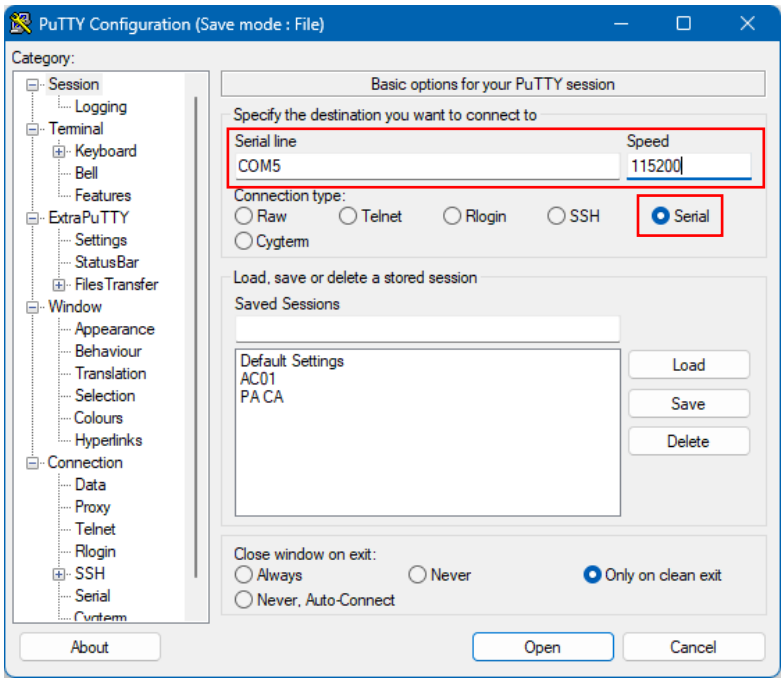
How to send commands to the reader? (Windows)

To change the reader's settings in Autonomous mode, a terminal emulation software can be used. The most well-known open-source and free one is **Putty**, the download link for which is provided below:

https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

To configure Putty, follow the steps below:

1. Open Putty and select "Connection type": "**Serial**", enter the "Serial line" as the port generated by the reader (COMXX), and set the "Speed" to **115200**.



2. Click "Open" to open the terminal screen for entering commands.

How to send commands to the reader? (Linux and MacOS)

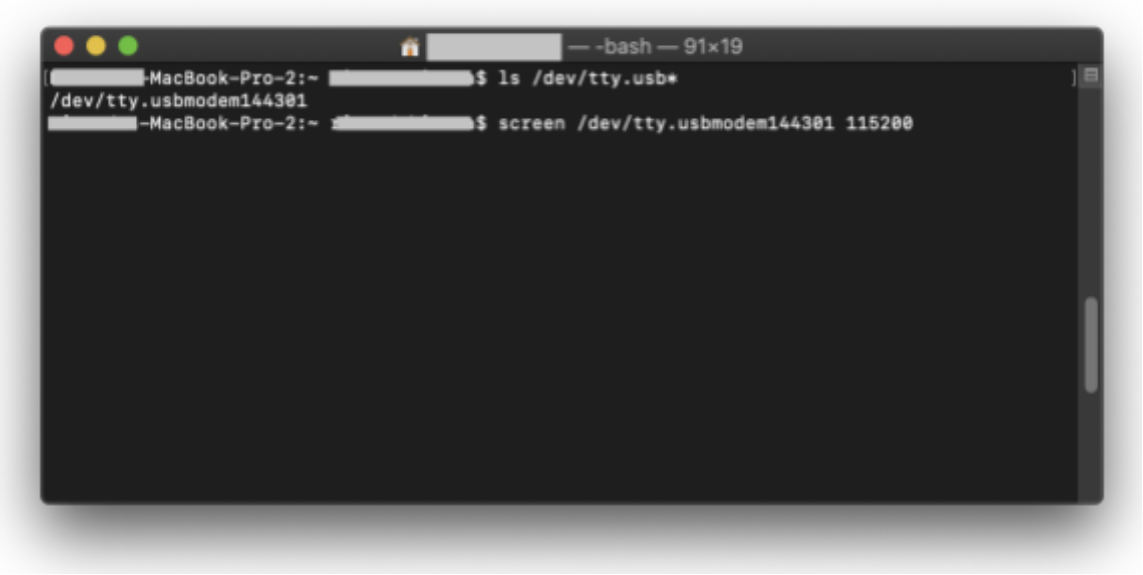
On Linux operating systems you can also use Putty as explained above for Windows.

For Mac OS and Linux you can use the "screen" program as below.

NOTE: To find out on which serial port the reader will respond, please consult **Turning on the reader**.

- MacOS:

$ screen /dev/tty.usbmodemXXXXXX 115200

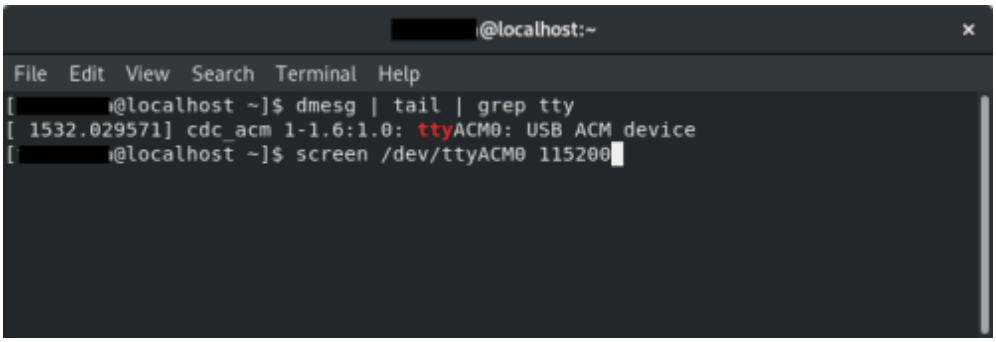


- Linux – Using screen:

NOTE: In Linux, to have permission to use the serial port, run the command with superuser privileges (sudo) or add your user to the dialout group using the following command:
$ sudo usermod -a -G dialout $USER
For this to take effect, you need to restart the system.

To connect using the screen:

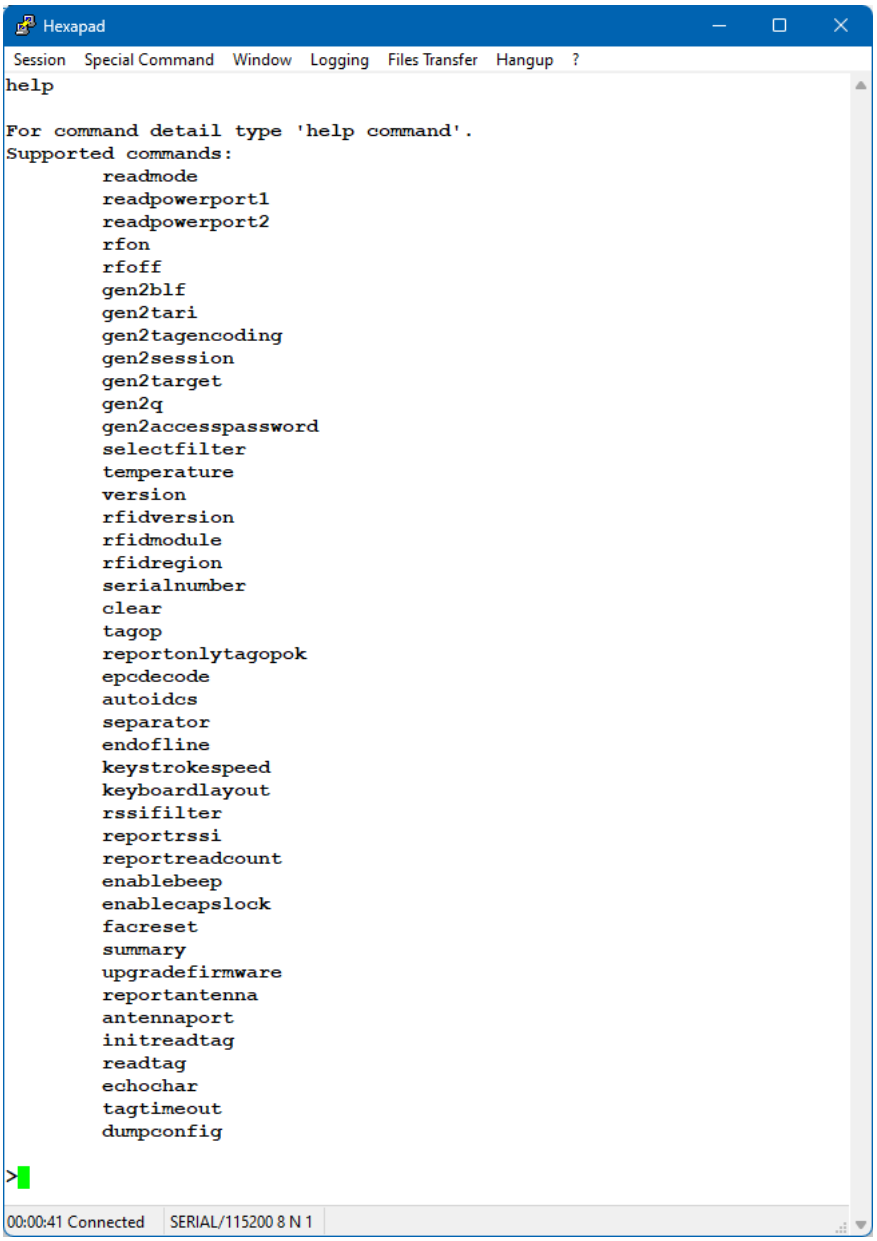
$ screen /dev/ttyACMx



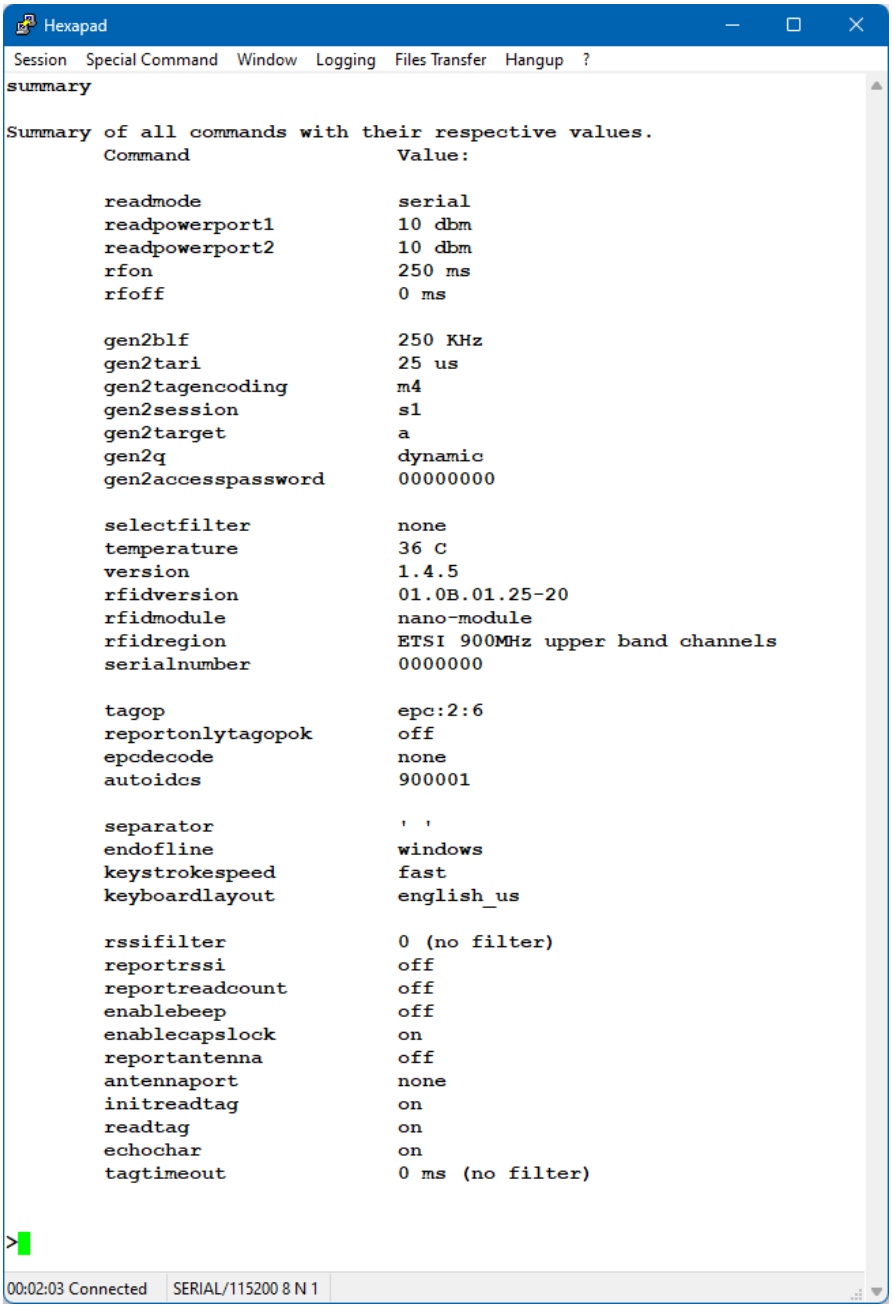
Configuration commands

Important tips

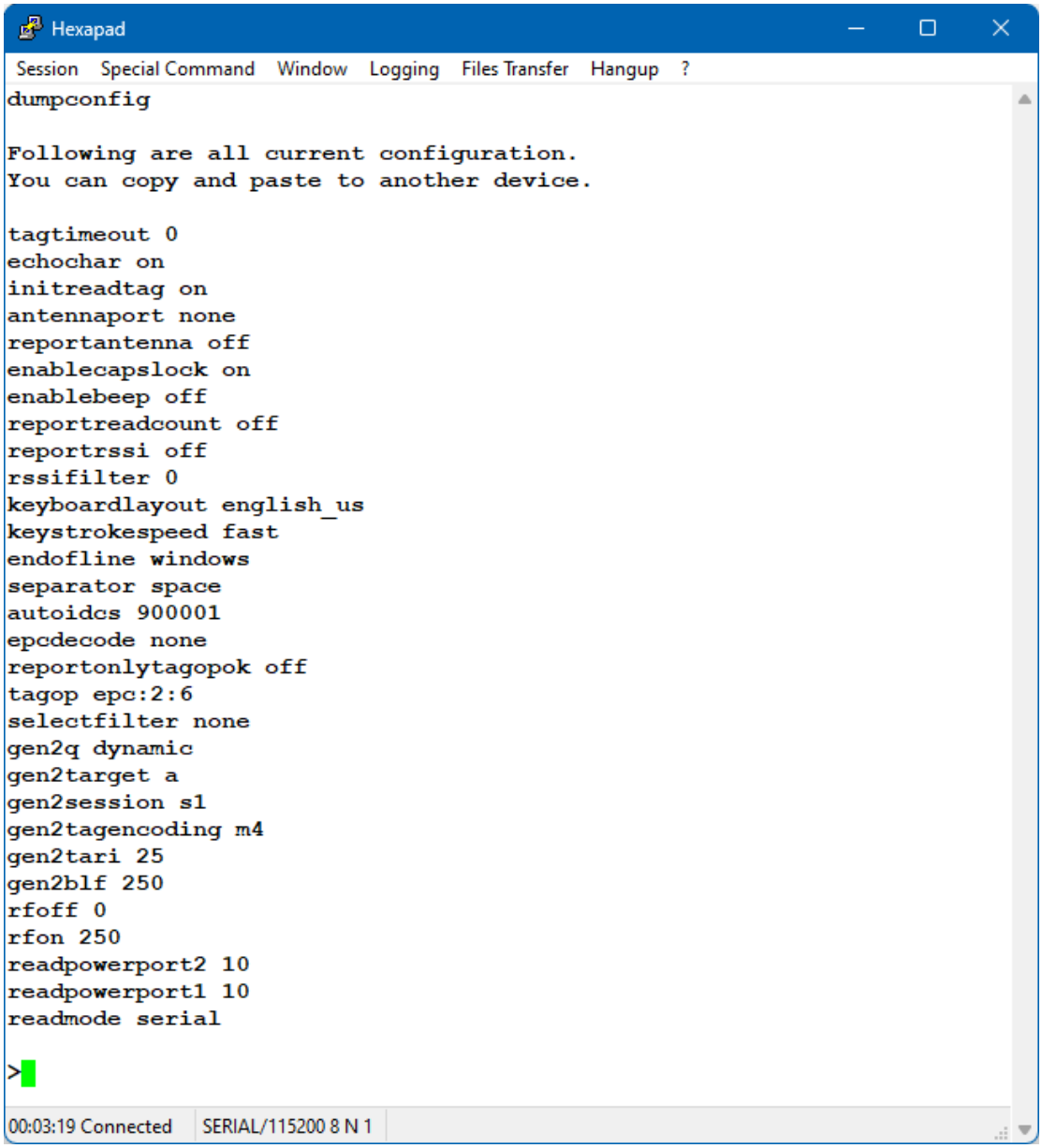
It's not always easy to remember the command name or accepted values, for that, use the **help** command. When executed without any values, it lists all the commands supported by the reader. To get details on a specific command, simply type **help (command)<CR>** – this is a quick way to access the reader's configuration information. Below are images of the **help** command.



The **summary** command returns all the current values and status of the reader parameters in a formatted list as follows:



The **dumpconfig** command returns all the current values of the reader parameters in a simple formatted list. This entire list can be copied from the terminal and be pasted to another terminal connected to another reader.



Summary of configuration commands and default values

	Command	Short description	Default value
Operation	**readmode**	Reader operating mode.	hid
	readpowerport1	Read power in dBm for antenna port 1.	10
	readpowerport2	Read power in dBm for antenna port.	10
	rfon	Time in milliseconds of RF ON.	250
	rfoff	Time in milliseconds of RF OFF.	250
	antennaport	Antenna port to use, physically connected.	none
Tag Gen2 protocol	**gen2blf**	Air Backscatter Link Frequency in KHz.	250
	gen2tari	Tari (bit pulse length) in us.	25
	gen2tagencoding	Air tag-reader communication encoding.	m4
	gen2session	Tag Session value.	s1
	gen2target	Tag Target value.	a
	gen2q	q value, which determine the tag population.	dynamic
	gen2accesspassword	Tag gen2 access password.	00000000
	selectfilter	Tag select filter.	none
Tag reading	**initreadtag**	Initial state of reading tag.	on
	readtag	Current reading tag condition.	on
	tagtimeout	Same tag id filter, in milliseconds.	0
	rssifilter	Filter by the value of tag RSSI in dbm.	0
	tagop	Read tag operation.	none
	reportonlytagopok	Report only if the tagop was successfully executed.	off
	epcdecode	How to decode the tag EPC.	none
	autoidcs	Secure code for the Acura Autoid tags.	900001
Messages reply	**separator**	Character between the results fields.	"space"
	endofline	Endofline for tag reads report string.	windows
	keystrokespeed	Keystroke speed for Wedge emulation.	fast
	keyboardlayout	Layout for Wedge emulation.	english_us
	reportrssi	Report the tag rssi value.	off
	reportantenna	Report the antenna the tag was read.	off
	reportreadcount	Report the tag readcount value of each rfon.	off

	Command	Short description	Default value
Information	**summary**	List all commands with their respective values.	
	temperature	Temperature of the reader in Celsius degree.	
	version	Processor firmware version.	
	rfidversion	RFID firmware version.	
	rfidmodule	RFID module type.	
	rfidregion	Current RFID region.	
	serialnumber	Reader serial number.	
	dumpconfig	Lists all the current configuration of the device in a format that can be copied and pasted via terminal in another device.	
Miscellaneous	**enablebeep**	Enable beep sound.	on
	enablecapslock	CapsLock to switch readmode Transparent/Hid.	on
	echochar	Echo character through serial comm.	on
	facreset	Returns all configuration to factory default.	
	upgradefirmware	Run the Bootloader to perform a firmware upgrade.	
	clear	To clear the terminal screen.	
	status	Status of the reader.	
	help	List all commands or display details of specific one.	

Response Messages to Commands

When receiving a command, the reader executes and responds with one of the following messages:

Type	Message
Success	ok - command executed. ok - parameter has been changed. ok - parameter already has this value. ok - transparent mode off, resetting reader. !!! Factory Reset in action !!! The reader will be restarted. !!! Bootloader will be executed !!! The reader will be restarted.
Errors	error - command not found. error - too many args. error - value not supported for this RF Module. error - command not supported for this hardware. error - command not supported for this comm interface. error - value out of range. error - gen2 wrong parameters combination. error - internal buffer full. error - need a command to send.

Configuration command details

readmode

Syntax	**readmode<CR>** **readmode value<CR>**
Function	Check (no arguments) or change the mode of operation of the reader.
Description	The reader can work in Autonomous Mode, with automatic tag readings, or in Transparent Mode, where a Software can have full control of the reader for tag reading and writing.
Values	**hid** - Keyboard emulation mode. **serial** - Serial communication mode, the read result is sent via serial communication on the USB virtual port. **hidserial** - Keyboard emulation mode and USB Serial, the reading result is sent via keyboard emulation and via serial communication on the USB virtual port. **transparent** - The reader must be controlled by a Software that has full control to read and write tags.
Default	**hid**
Example	To get the current operation mode: **readmode<CR>** **hid<CR>** To change the mode of operation (Autonomous reporting via serial only): **readmode serial<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

readpowerport1

Syntax	**readpowerport1<CR>** **readpowerport1 value<CR>**
Function	Check (no arguments) or change the read power in dBm for antenna port 1.
Description	Tag reading distance is directly related with the reading power.
Values	**0..30** for AcuPad-50 **0..27** for AcuPad-50 MUX
Default	**10**
Example	To get the current reading power: **readpowerport1<CR>** **10<CR>** To change the reading power to 18 dBm: **readpowerport1 18<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

NOTE: The reading power directly influences the tag's reading distance, the temperature at which the reader operates, and its power consumption. We recommend adjusting the reading power according to your specific application.

readpowerport2

Syntax	**readpowerport2<CR>** **readpowerport2 value<CR>**
Function	Check (no arguments) or change the read power in dBm for antenna port 2.
Description	Tag reading distance is directly related with the reading power.
Values	**0..30** for AcuPad-50 **0..27** for AcuPad-50 MUX
Default	**10**
Example	To get the current reading power: **readpowerport2<CR>** **10<CR>** To change the reading power to 18 dBm: **readpowerport2 18<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

NOTE: The reading power directly influences the tag's reading distance, the temperature at which the reader operates, and its power consumption. We recommend adjusting the reading power according to your specific application.

rfon

Syntax	**rfon<CR>** **rfon value<CR>**
Function	Check (no arguments) or change the read time in ms (milliseconds).
Description	The reading time is equivalent to the time when the reader transmits RF power to its antenna. When you have large number of tags to read at the same time, the reading time should be increased.
Values	**50..5000**
Default	**250**
Note	Tag reading results are reported by the reader after the sum rfon + rfoff time periods.
Example	To get the current reading period: **rfon<CR>** **250<CR>** To change the reading time to 320ms: **rfon 320<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

NOTE: The reading results are reported by the reader after the sum of the RF On and RF Off time periods.

NOTE: If the application requires reading multiple tags simultaneously, set RF On time to more than 250ms to allow sufficient time to power up all tags within the reader's field.

rfoff

Syntax	**rfoff<CR>** **rfoff value<CR>**
Function	Check (no arguments) or change the period of RF idle in ms (milliseconds).
Description	This is the period of RF idle with no tag reading.
Values	**0..1000**
Default	**250**
Note	Tag reading results are reported by the reader after the sum rfon + rfoff periods.
Example	To get the current reading period: **rfoff<CR>** **250<CR>** To change the RF idle to 0ms: **rfoff 0<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

NOTE: The reading results are reported by the reader after the sum of the RF On and RF Off time periods.

antennaport

Syntax	**antennaport<CR>** **antennaport value<CR>**
Function	Get (no arguments) or set the antenna(s) to use, physically connected.
Description	The physically antenna connected to the reader must be informed using this command.
Values	**none** No antenna connected, the reader will keep in an idle loop. **1** Use port 1. **2** Use port 2. **12** or **21** Use both ports.
Default	**none**
Note	It is a good practice to use this command only after the physically antenna connection is made.
Example	To check the value: **antennaport<CR>** **none<CR>** To use antenna connected on port 2 to read tags: **antennaport 2<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

gen2blf

Syntax	**gen2blf<CR>** **gen2blf value<CR>**
Function	Check (no arguments) or change the "Backscatter Link Frequency" parameter of the Gen2 air protocol.
Description	BLF is the data rate in KHz of transmission by air between a tag and the reader.
Values	**250** **320** **640**
Default	**250**
Note	By changing the BLF to 640 or 320 KHz, the tari value and tag encoding will automatically change to 6.25 us and fm0 respectively.
Example	To get the current BLF: **gen2blf<CR>** **250<CR>** To change the BLF to 640Khz **gen2blf 640<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR> error - value not supported for this RF Module.<CR>

gen2tari

Syntax	**gen2tari<CR>** **gen2tari value<CR>**
Function	Check (no arguments) or change the "Tari" parameter of the Gen2 air protocol.
Description	Tari is the time in us (microseconds) of the pulse of each bit transmitted by the tag.
Values	**6.25** **12.5** **25**
Default	**25**
Note	Values 12.5 us and 25 us can only be set if the gen2blf is set to 250 KHz. The values 6.25 us and 12.5 us are supported only by the internal RF module micro-module type, which can be checked by rfidmodule command.
Example	To get the current Tari: **gen2tari<CR>** **25<CR>** To change the Tari to 6.25Khz **gen2tari 6.25<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR> error - value not supported for this RF Module.<CR> error - gen2 wrong parameters combination.<CR>

NOTE: The value of Tari should be combined with the value of BLF.

gen2tagencoding

Syntax	**gen2tagencoding<CR>** **gen2tagencoding value<CR>**
Function	Check (no arguments) or change the "Tag data encoding" parameter of the Gen2 air protocol.
Description	It is the encoding format of the tag response.
Values	**fm0** **m2** **m4** **m8**
Default	**m4**
Note	m2, m4 and m8 can only be configured if the blf is 250 KHz. fm0 has a low read sensitivity rate but a higher speed in decoding each bit. m8 has a high read sensitivity rate but a lower speed in decoding each bit.
Example	To get the current Tari: **gen2tagencoding<CR>** **m4<CR>** To change the Encoding format to FM0Khz **gen2tagencoding FM0<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR> error - value not supported for this RF Module.<CR> error - gen2 wrong parameters combination.<CR>

gen2session

Syntax	**gen2session<CR>** **gen2session value<CR>**
Function	Check (no arguments) or change the "Session" parameter the Gen2 air protocol inventory operation.
Description	In which session the reader will read the tag.
Values	**s0** **s1** **s2** **s3**
Default	**s1**
Note	The combination of gen2session and gen2target values reflects the read rate of the same tag.
Example	To get the current Session: **gen2session<CR>** **s1<CR>** To change the Session to S0 **gen2session s0<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

gen2target

Syntax	**gen2target<CR>** **gen2target value<CR>**
Function	Check (no arguments) or change the "Target", or Inventory Flag of the Gen2 air protocol.
Description	Which inventory flag (target), the reader should search for a tag.
Values	**a** **b** **ab** **ba**
Default	**a**
Note	The combination of gen2session and gen2target values reflects the read rate of the same tag.
Example	To get the current Target: **gen2target<CR>** **a<CR>** To change the Target to ab **gen2target ab<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

gen2q

Syntax	**gen2q<CR>** **gen2q value<CR>**
Function	Check (no arguments) or change the "Q" parameter of the Gen2 air protocol.
Description	It is a numeric value that the reader uses to regulate the probability of a tag's response, used for anti-collision when multiple tags are in the reader's reading field. 2^q equals the number of tags to be read at the same time.
Values	**dynamic** - The reader automatically adjusts the Q value during readings. **0..15** - Fixed value for 2^q number of tags in the reading field of the reader.
Default	**dynamic**
Note	If the number of tags to be read at the same time varies, leave the q value in dynamic, so the reader automatically adjusts the best value for each reading.
Example	To get the current Q value: **gen2q<CR>** **dynamic<CR>** To change the Q value to an estimated tag population of 8 tags. **gen2q 3<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

NOTE: Use **gen2q** as **dynamic** if the quantity of tags in the reading field varies or is unknown.

gen2accesspassword

Syntax	**gen2accesspassword<CR>** **gen2accesspassword value<CR>**
Function	Check (no arguments) or change the "Access password" of the tag.
Description	It is 32 bits length hexa number.
Values	**00000000..ffffffff**
Default	**0**
Note	To read the reserved memory bank of the tag with the tagop command, the password must be the same as the tag.
Example	To get the current access password: **gen2accesspassword<CR>** **00000000<CR>** To change the access password to 12345678. **gen2accesspassword 12345678<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

NOTE:	To read the protected memory region of the tag (Reserved), the tag's access password must be provided to the reader through this command.

selectfilter

Syntax	**selectfilter<CR>** **selectfilter value<CR>**
Function	Check (no arguments) or set the Select filter tag using a mask value in a specific bank.
Description	This filter allows you to specify which tags should respond in an inventory. This feature allows you to isolate specific tags in large population. Tags that do not match the specified filter will not transmit a response, which greatly reduces the amount of RF traffic.
Values	**res:mask:bp:bl -** Reserved bank.* **epc:mask:bp:bl -** EPC bank.* **tid:mask:bp:bl -** TID bank.* **user:mask:bp:bl -** User bank.* **none -** No Select filter. * mask - mask value [hexa], bp - bit pointer, bl - bit length, max 96.
Default	**none**
Example	To get the current Select filter: **selectfilter<CR>** **none<CR>** To read only tags with EPC id starting with 0xAC01: **selectfilter epc:AC01:32:16** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

initreadtag

Syntax	**initreadtag<CR>** **initreadtag value<CR>**
Function	Check (no arguments) or set the read tag state (on/off) of the reader when it is initialized.
Description	The reader can always be preset to have the RF signal either turned off or turned on when initialized.
Values	**on** **off**
Default	**on**
Example	To check the initreadtag value: **initreadtag<CR>** **on<CR>** To disable the reader (RF signal) when it is initialized: **initreadtag off<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

readtag

Syntax	**readtag<CR>** **readtag value<CR>**
Function	Check (no arguments) or turn RF on/off.
Description	Enables or disables tag reading by turning off the reader's RF signal.
Values	**on** - Enables tag reading, RF on. **off** - Disables tag reading, RF idle.
Default	Depends on how initreadtag is setup. Volatile data.
Note	RF on/off can be toggled by pressing 2x fast the CapsLock key, the enablecapslock must be set to on.
Example	To check the current value: **readtag<CR>** **on<CR>** To change the RF to idle: **readtag off<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

tagtimeout

Syntax	**tagtimeout<CR>** **tagtimeout value<CR>**
Function	Check (no arguments) or change the reading interval time in milliseconds (ms) for the same reported tag (same EPC filter).
Description	With this time filter it is possible to manipulate the same tag read rate.
Values	**0..300000** (time in milliseconds)
Default	**0 ms**
Example	To check the reading interval time of the same tag: **tagtimeout<CR>** **1000<CR>** To change the reading interval time of the same tag to 500 ms (0.5 s): **tagtimeout 500<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

rssifilter

Syntax	**rssifilter<CR>** **rssifilter value<CR>**
Function	Check (no arguments) or change the cutoff value of the RSSI filter for reading.
Description	The signal level of the tag indicates whether the tag is near or away from the antenna, when closer to the antenna, the stronger the RSSI (closer to zero) and vice versa. If a value is specified for the filter, only tags with RSSI equal to or greater than that specified will be reported by the reader. The signal level of the tag is measured in dBm and is a negative value, i.e. the power in Watts is always less than 1mW.
Values	**0..99**
Default	**0**
Notes	To disable the filter, write the value 0. The specified value should not contain the minus sign.
Example	To check the RSSI filter value: **rssifilter<CR>** **0<CR>** To change the RSSI filter value to 45: **rssifilter 45<CR>** **ok - parameter has been changed.<CR>**
Returns	"value"<CR> ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

NOTE: The filter based on the tag's RSSI value is used to restrict undesired reading areas around the reader. It should always be used in conjunction with the reading power.

tagop

Syntax	**tagop<CR>** **tagop value<CR>**
Function	Check (no arguments) or change the tag read operation.
Description	With the tag read operation, it is possible to read any memory bank of the tag.
Values	**none** - No memory bank read operation. **autoid** - Reads Acura's Autoid System tags. **reserved:wp:wl** - Reads the Reserved memory bank of the tag. **epc:wp:wl** - Reads the EPC memory bank of the tag. **tid:wp:wl** - Reads the TID memory bank of the tag. **user:wp:wl** - Reads the USER memory bank of the tag.
Default	**none**
Note	**wp** - word pointer, indicates the initial word (16 bits) read position in the specified memory bank. **wl** - word length, indicates the number of words to read in the specified memory bank. Maximum value for wp and wl is 32. To read the entire memory bank, simply specify **zero** for **wp** and **wl**.
Example	To read all the contents of the User memory bank of the tag: **tagop user:0:0<CR>** **ok - parameter has been changed.<CR>** To read four words of the Tid memory bank from the second word: **tagop tid:2:4<CR>** **ok - parameter has been changed.<CR>** To not read any specific tag memory bank, just the EPC number: **tagop none<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

reportonlytagopok

Syntax	**reportonlytagopok <CR>** **reportonlytagopok value<CR>**
Function	Check (no arguments) or set the choice of report only if the tagop was successfully executed.
Description	This command is used to force the reader to send only the results when the tagop command is executed successfully.
Values	**on** **off**
Default	**off**
Note	This has effect only if the command tagop is other than **none**.
Example	To check the reportonlytagopok value: **reportonlytagopok <CR>** **off<CR>** To change the value to set the reportonlytagopok on: **reportonlytagopok on <CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

epcdecode

Syntax	**epcdecode<CR>** **epcdecode value<CR>**
Function	Check (no arguments) or change the type of decodification will be used to process the EPC id of the tag.
Description	The tag to be read can be encoded for a specific application; for example, in checkout applications, the tag will be encoded using the GS1 standard, while in maintenance applications, the tag may be encoded using the ASCII table. The EPCID decoding provides a readable result for each respective application.
Values	**none** - No decoding is performed and the epcid result of the tag is reported in hexadecimal. **gs1string** - GS1 standard, that returns the "GS1 element string". **gs1epcuri** - GS1 standard, that returns the "EPC Tag URI". **gs1epcpureuri** - GS1 standard, that returns the "EPC Pure Identity URI". **gs1gtin13** - GS1 standard, that returns code GTIN13 without the serial number. **gs1sgtin13** - GS1 standard, that returns code GTIN13 + serial number. **gs1crystal –** Crystal custom standard, that returns GTIN13 + RFID + serial 12. **ascii** - Decodes the EPCID of the tag according to the Ascii table. **wiegand26** - Decodes the last 3 bytes of tag EPCID in Sitecode + ID. **magstripe** - Decodes the last 4 bytes of the EPCID of the tag in decimal. **decimal:dp:dl –** Decimal value of EPC split range of dp:dl. **mid:dp:dl –** Substring of EPC in hexadecimal. Max of 64 digits.
Default	**None**
Note	For the GS1 standard, the sgtin-96 and sgtin-198 schemes are supported. The wiegand26 and magstripe values can be used in conjunction with the autoid type **tagop**. If the reader is set to a specific epcdecode type and the tag being reading has a wrong format, a beep sound pattern different from the read sound pattern is executed by the reader. **dp –** digit pointer, indicates the initial digit to read in the EPC. **dl –** digit length, indicates the number of digits to read in the EPC. Maximum value for dl is 64.
Example	To read the GTIN13 + serial number of a tag encoded using the GS1 standard: **epcdecode gs1sgtin13<CR>** **ok - parameter has been changed.<CR>** To read the GS1 element string of a tag encoded using the GS1 standard: **epcdecode gs1string<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

separator

Syntax	**separator<CR>** **separator value<CR>**
Function	Check (no arguments) or change the separation character of the fields of the reading result of a tag.
Description	It is a character that separates the fields in the tag reading result string.
Values	**space** To have a space as a separator. **'char'** Any readable character in the Ascii table.
Default	**' ' (space)**
Notes	For space, you must write space as a value for the command.
Example	To check the separator: **separator<CR>** **space<CR>** To change the tab to '|': **separator |<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

NOTE: The separator character, along with the end-of-line, should be used to split the result string by the software that will receive this string.

endofline

Syntax	**endofline<CR>** **endofline value<CR>**
Function	Check (no arguments) or change the end-of-line character.
Description	Get (no arguments) or set the endofline for tag reads report string.
Values	**windows:qt** Windows standard end-of-line <CR><LF>. **unix:qt** Unix standard end-of-line <LF>. **macintosh:qt** Mac standard end-of-line <CR>. **tab:qt** The end-of-line will be the tabulation character Tab. **none** No end-of-line in the result string.
Default	**windows**
Notes	**qt** - Number of end-of-line repetitions, values 1 to 10. If **:qt** is omitted, it is considered only one end of line.
Example	To check which end-of-line is configured: **endofline<CR>** **windows<CR>** To change the end of line to Tab and repeat 3 times: **endofine tab:3<CR>** **ok - parameter has been changed.<CR>** To change the end-of-line to Windows and repeat it 5 times: **endofline windows:5<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

NOTE: The separator character, along with the end-of-line, should be used to split the result string by the software that will receive this string.

keystrokespeed

Syntax	**keystrokespeed<CR>** **keystrokespeed value<CR>**
Function	Check (no arguments) or change the typing speed of the keyboard emulation.
Description	The typing speed in keyboard emulation, when readmode is as hid or hidserial, can be changed to three different values (fast, medium and slow).
Values	**fast** Keystroke at fast speed, T=4ms. **medium** Keystroke at medium speed, T=32ms. **slow** Keystroke at slow speed, T=64ms.
Default	**fast**
Example	To check the typing speed: **keystrokespeed<CR>** **fast<CR>** To change to slow speed: **keystrokespeed slow<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

NOTE: This value should only be changed if the Host that will receive the result string via keyboard emulation experiences issues due to the typing speed exerted by the reader.

keyboardlayout

Syntax	**keyboardlayout<CR>** **keyboardlayout value<CR>**
Function	Check (no arguments) or change the keyboard layout when the readmode is hid or hidserial (wedge emulation).
Description	Select the keyboard layout for use with the reader when it is configured to emulate a keyboard. This ensures that characters and letters follow the defined layout.
Values	**english_us** English (United States). **portuguese_br** Portuguese (Brazil). **spanish_latam** Spanish (Latim America)
Default	**english_us**
Example	To check the keyboard layout: **keyboardlayout <CR>** **english_us<CR>** To change to brazilian layout: **keyboardlayout portuguese_br<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

reportrssi

Command	**reportrssi<CR>** **reportrssi value<CR>**
Function	Check (no arguments) or change the transmission of the RSSI value from the read tag.
Description	Enables or disables sending the RSSI value of the tag read in the result string.
Values	**on** **off**
Default	**off**
Notes	The RSSI value is reported with the negative sign '-'.
Example	To check the value: **reportrssi<CR>** **off<CR>** To change the value to include the RSSI value in the result string: **reportrssi on<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

reportantenna

Command	**reportantenna<CR>** **reportantenna value<CR>**
Function	Check (no arguments) or change the transmission of the antenna from the read tag.
Description	Enables or disables sending the antenna the tag was read in the result string.
Values	**on** **off**
Default	**off**
Example	To check the value: **reportrssi<CR>** **off<CR>** To change the value to include the RSSI value in the result string: **reportrssi on<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

reportreadcount

Syntax	**reportreadcount<CR>** **reportreadcount value<CR>**
Function	Check (no arguments) or change the transmission of the tag's read quantity.
Description	Enables or disables the transmission of the tag's read quantity during the "rfon" period in the result string.
Values	**on** **off**
Default	**off**
Notes	The maximum value for the read quantity is 255.
Example	To check the value: **reportreadcount<CR>** **off<CR>** To change the value to include the read quantity in the result string: **reportreadcount on<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

summary

Syntax	**summary<CR>**
Function	Lists all the values configured in the reader.
Errors	error - too many args.<CR>

temperature

Syntax	**temperature<CR>**
Function	Check (no arguments) the internal temperature of the reader in Celsius degree.
Errors	error - too many args.<CR>

version

Syntax	**version<CR>**
Function	Get (no arguments) the processor firmware version.
Errors	error - too many args.<CR>

rfidversion

Syntax	**rfidversion<CR>**
Function	Get (no arguments) the RFID firmware version.
Errors	error - too many args.<CR>

rfidmodule

Syntax	**rfidmodule<CR>**
Function	Get (no arguments) the RFID module type.
Errors	error - too many args.<CR>

rfidregion

Syntax	**rfidregion<CR>**
Function	Get (no arguments) the current RFID region.
Errors	error - too many args.<CR>

NOTE: The reader's regional configuration is determined by the firmware.

serialnumber

Syntax	**serialnumber<CR>**
Function	Get (no arguments) the reader serial number.
Errors	error - too many args.<CR>

dumpconfig

Syntax	**dumpconfig<CR>**
Function	Lists all the current configuration of the device in a format that can be copied and pasted via terminal in another device.
Description	This command can compile a list of the present reader configurations, streamlining the process of archiving or duplicating them across other readers with greater efficiency.
Errors	error - too many args.<CR>

enablebeep

Syntax	**enablebeep<CR>** **enablebeep value<CR>**
Function	Check (no arguments) or change the beep with each reading of the tag.
Description	Enables or disables beeping with each tag reading.
Values	**on** **off**
Default	**on**
Example	To check the value: **enablebeep<CR>** **off<CR>** To turn off the tag read alert sound: **enablebeep off<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

enablecapslock

Syntax	**enablecapslock<CR>** **enablecapslock value<CR>**
Function	Check (no arguments) or change the use of the shortcut to change read mode.
Description	Enables or disables the use of the "CapsLock" 5x quick-type key, to switch from Autonomous to Transparent Mode and vice versa.
Values	**on** **off**
Default	**on**
Notes	This feature only works on Windows and Linux operating systems. An alternative to exit transparent mode is to send the command **$ off<CR>** to the reader.
Example	To check the value: **enablecapslock<CR>** **off<CR>** To change the value to disable shortcut use: **enablecapslock on<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

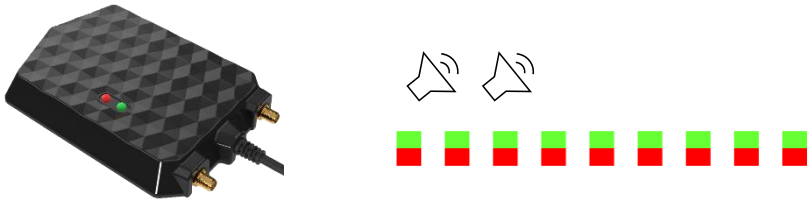
NOTE: For more details on changing the reader's operating mode, please refer to **Switching Autonomous Mode to Transparent Mode and vice versa**.

echochar

Syntax	**echochar<CR>** **echochar value<CR>**
Function	Check (no arguments) or turn echo on/off.
Description	Enables or disables the echo of each byte sent by serial communication in Autonomous Mode.
Values	**on** **off**
Default	**on**
Notes	When configured to **off**" typing in terminal software will not appear unless character echoing is set up in the terminal software. It is recommended to turn off the **echochar** when custom developed software will send the ASCII commands to the reader.
Example	To check the value: **echochar<CR>** **on<CR>** To disable the echo of sent bytes: **echochar off<CR>** **ok - parameter has been changed.<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

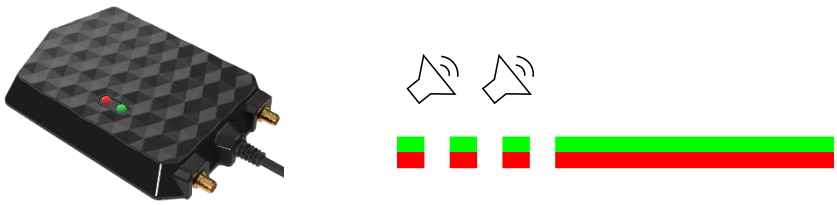
facreset

Syntax	**facreset<CR>**
Function	Returns all values to the factory default.
Errors	error - too many args.<CR>

	Beep, Green and Red LEDs flash rapidly simultaneously for 5 seconds to indicate that it's in the factory reset process.

upgradefirmware

Syntax	**upgradefirmware<CR>**
Function	Starts the reader firmware update process.
Example	To check the commands accepted by the reader: **help<CR>** **command list... <CR>** **upgradefirmware** !!! Bootloader will be executed!!! The reader will be restarted.
Errors	error - too many args.<CR>

	Beep, Green and Red LEDs flash rapidly 3 times and then both LEDs solid to indicate the bootloader execution.

help

Syntax	**echochar<CR>** **echochar value<CR>**
Function	Lists all commands (no arguments) or provides detailed help for the given command.
Description	For detailed help on a particular command just type: "help command<CR>".
Example	To check the commands accepted by the reader: **help<CR>** **command list... <CR>** For detailed help on the tagop command: **help tagop<CR>** **Command:** **tagop arg<CR>** **Description:** **Get (no arguments) or set the read tag operation.<CR>** **Arguments:** **reserved:wp:wl** **Reserved bank.*<CR>** **epc:wp:wl** **EPC bank.*<CR>** **tid:wp:wl** **TID bank.*<CR>** **user:wp:wl** **User bank.*<CR>** **autoid** **Acura Autoid tags.<CR>** **pa** **Artefato SJ5511 tags.<CR>** **none** **No tagop.<CR>** *** wp - word pointer, wl - word length, max 32, membank:0:0 entire bank** **Default:** **none<CR>**
Returns	ok - parameter already has this value.<CR> ok - parameter has been changed.<CR> error - value out of range<CR> error - too many args.<CR>

Switching Autonomous Mode to Transparent Mode and vice-versa

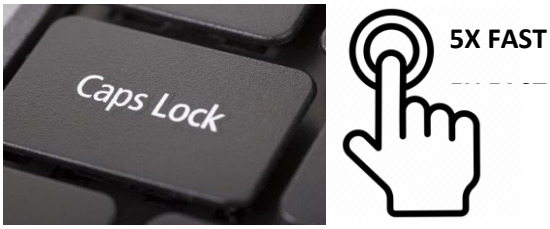
For changing the operation mode of the reader the following options can be used:

A. Use the **readmode command**, examples:
   a. The reader is in Autonomous mode, and the command **readmode transparent<CR>** is sent; this way, the reader enters Transparent mode of operation.
   b. The reader is in Transparent Mode and the command **$ off<CR>** is sent, so the reader enters Autonomous Mode with hid interface.

NOTE: When in Transparent Mode, the reader does not echo the characters typed in the serial terminal.

B. Use the **Caps Lock** key (Windows and Linux only), example:
   a. Close any serial communication with the reader.
   b. To switch between one operating mode and another, simply ***quickly click the Caps Lock key 5 times in sequence*** and check the reader's LEDs to see the change.

NOTE: The command **enablecapslock** enables or disables this functionality, to use it the **enablecapslock** must be on.



ALERT: The Caps Lock shortcut for changing the operating mode only works if and only if the reader's serial port is not connected (not being used by any software).

Read rate of the same tag

To control the read rate it is recommend to use the **tagtimeout** command as explained below.

The read rate of the same tag can be controled through the following commands:

- **gen2session**
- **gen2target**
- **rfon**
- **rfoff**
- **tagtimeout**

The commands mentioned above, except **tagtimeout**, change the parameters of the UHF EPC Gen2 air protocol and influence the communication between the tag and reader.

Combinations of values for the Session and Target of the tag reflect how often a tag that has already responded to an inventory will respond again.

Examples:

A. I would like to read the EPC id of a tag only once, and I don't want the same EPC id to be repeated while the tag is in the reading field. Configure it as follows:
gen2session s2<CR>
gen2target a<CR>

B. I would like to read the same tag again and again while it is in the reading field. Configure it as follows:
gen2session s0<CR>
gen2target ab<CR>

In this case, what will determine the read repetition rate are the **rfon** and **rfoff** commands, meaning the total time will dictate the repetition rate. The highest possible read repetition rate is every 50ms:
rfon 50<CR>
rfoff 0<CR>

The read rate of the same tag can also be tweaked through the **tagtimeout** command. This command acts as a filter for tag reading, unlike the EPC Gen2 commands (Session and Target) mentioned above, which are part of the air protocol definition.

The **tagtimeout** command ranges from 0 to 300000 ms, up to 5 minutes. During this time, the same tag EPC id will not be reported by the reader again, even if it remains in the reading field.

Example:

A. I would like to read the tag and only be able to read it again after 20 seconds. Configure the reader as follows:
tagtimeout 20000<CR>
gen2session s0<CR>
gen2target ab<CR>

An important point to note is that, depending on the Session and Target configurations in the reader, the tag response time can override the time defined in **tagtimeout**.

Example:

A. I would like to configure the tagtimeout to 1 second, but I will keep the UHF Gen2 configuration as Session 2.
tagtimeout 1000<CR>
gen2session s2<CR>
gen2target a<CR>

WARNING: The tag read rate can be much faster than the rate at which results are sent through keyboard emulation (HID). In other words, if a high reading rate is chosen and the chosen interface is keyboard emulation, the reader will store the reading result and continue sending it even after the tag is removed from the reading field.

Use cases (Autonomous Mode)

Below are some examples of using and applying the AcuPad-50 family readers with the details of the configuration commands.

Checkout – GS1/EAN standard

For applications that use UHF EPC Gen2 tags encoded according to the *GS1 EPC Tag Data Standards (TDS) 1.11* standard, the AcuPad-50 family readers can be used and supports the following tag encodings:

- **sgtin-96**
- **sgtin-198**

Is an encoding scheme that stands for Serialized Global Trade Identification Number and is sometimes called a serialized GTIN. The 96 in the name refers to the number of bits in the final EPC binary form.

If you need alphanumeric characters in the serial numbers, SGTIN-198 must be used. This means that you must use RFID tags with higher capacity in the EPC memory, than if you use the SGTIN-96.

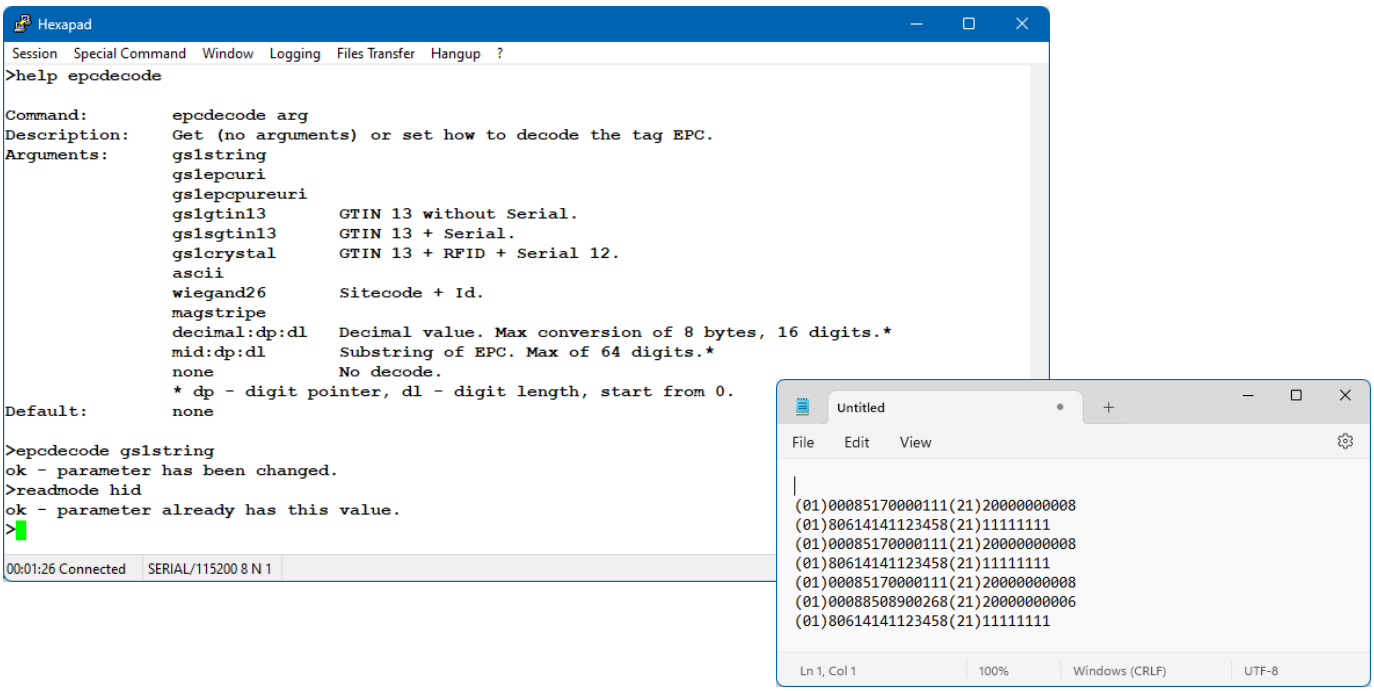
The results of reading the encoded tag can be sent in the following formats:

- **GS1 element string**, example "(01)03608449920322(21)1540341";
- **EPC Tag URI**, example "urn:epc:tag:sgtin-96:1.360844.0992032.1540341";
- **EPC Pure Identity URI**, example "urn:epc:id:sgtin:360844.0992032.1540341";
- **GTIN13 without the serial number**, example "3608449920322";
- **GTIN13 + serial number**, example "3608449920322 1540341";

Read tag encoded in SGTIN-96/198 and show result in GS1 element string format

Configure the reader as follows:

epcdecode gs1string<CR>
(to decode sgtin-96 or sgtin-198 and send in GS1 element string format)

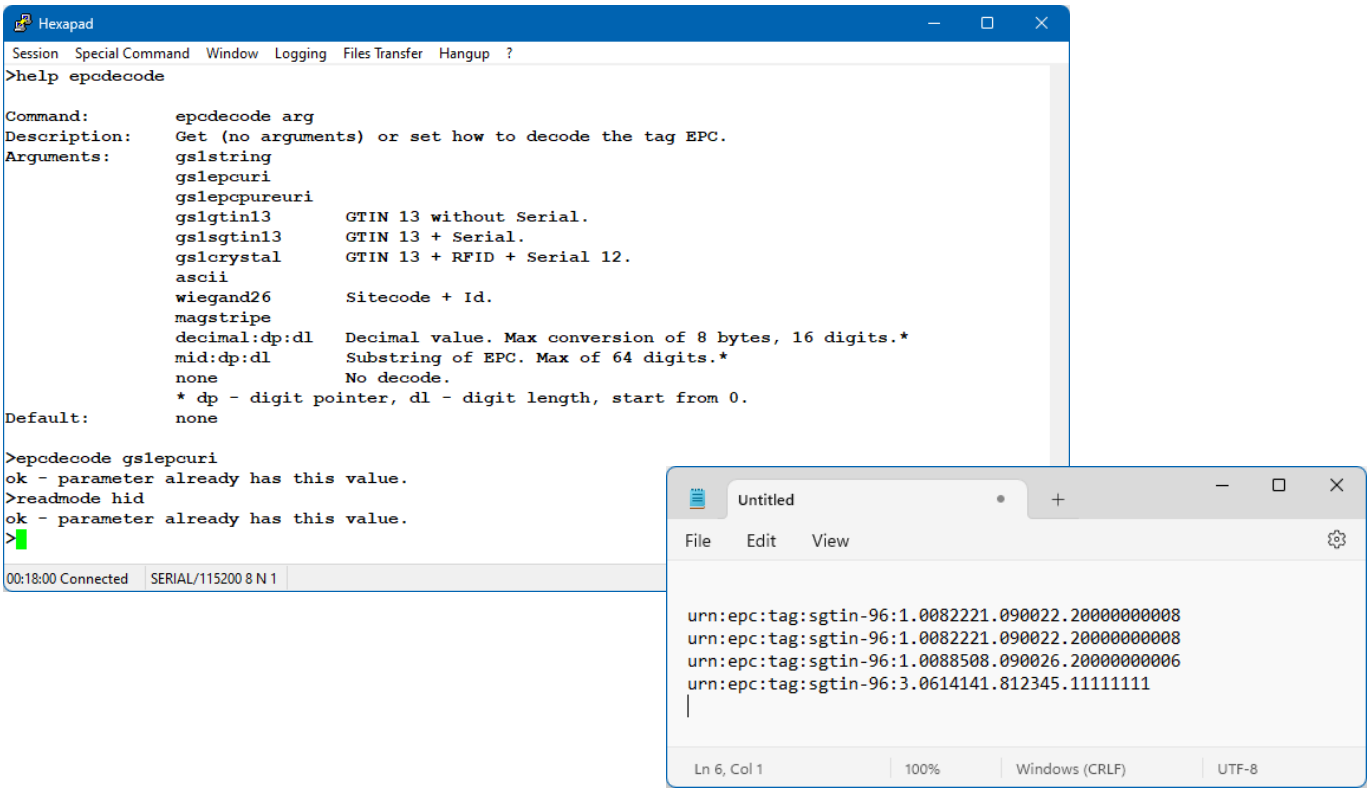


Read tag encoded in SGTIN-96/198 and show result in EPC Tag URI format

Configure the reader as follows:

epcdecode gs1epcuri<CR>

(to decode sgtin-96 or sgtin-198 and send in EPC Tag URI format)



For the other formats just use the **epcdecode** command with the equivalent format that can be checked in **Configuration commands > epcdecode**.

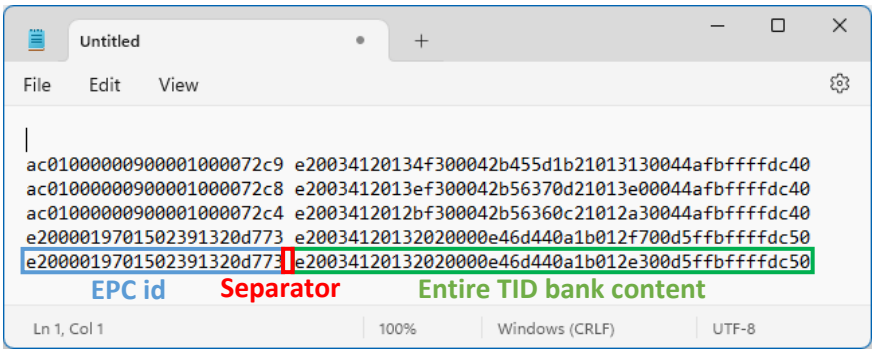
WARNING: The keyboard/language layout configured in the Operating System can change the ":" character of the result string of the epcdecode type gs1epcuri and gs1pureepcuri.

Read TID memory bank and EPC id

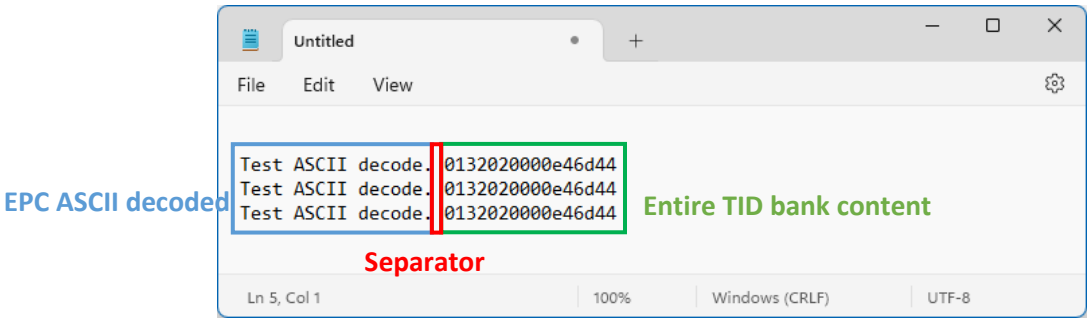
With the AcuPad-50 family readers it is possible to read any memory bank of the EPC Gen2 tag, (reserved, epc, tid and user memory), just configure by the **tagop** command in which database is desired to read along with the initial position and the size of the data (in word) that you want to read.

The syntax of the **tagop** command is explained in details in section **tagop**. Examples:

1. Read the entire TID bank of the tag plus the EPC id in hexa:
   tagop tid:0:0<CR>
   epcdecode none<CR>
   Example of reading two tags with different sizes from the TID bank:



2. Read from the TID bank 4 words skipping the first 2 words and decoding the EPC id by the ASCII table:
   tagop tid:2:4<CR>
   epcdecode ascii<CR>
   Example of reading two different tags:



In addition to TID bank, RESERVED, USER, and EPC banks can also be read in conjunction with the EPC tag id.

NOTE: The count unit in the EPC Gen2 standard is **word**, which is equivalent to 2 bytes or 16 bits.

NOTE: For the RESERVED bank, if the tag has an access password other than 0x00000000, the tag's password must be configured in the reader using the **gen2accesspassword** command.

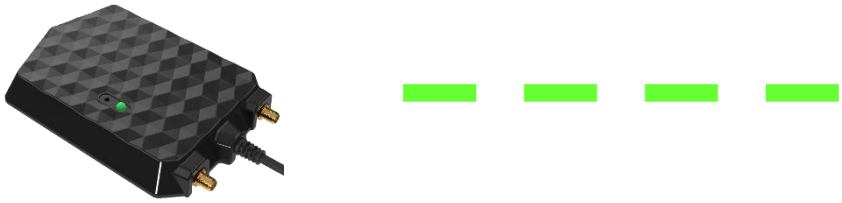
NOTE: More details of commands in **Configuration commands**.

Section – Transparent Mode

3.2 Reader in Transparent Mode

The reader in Transparent Mode starts to respond through the serial to a software developed using the Mercury API, which is a library package for C#/.Net, Java and C languages, with this development package it is possible to develop software that can have full control of the reader for reading and writing UHF EPC Gen2 tags (ISO 18000-6C) and UHF EPC Gen2 V2 (ISO 18000-63).

Visual feedback from the reader when in Transparent Mode:

	Green LED flashes slowly and red LED turned off.

The default operation mode of the reader is the Autonomous Mode, to change to Transparent Mode please refer:

Switching Autonomous Mode to Transparent and vice versa;
readmode

The reader's red LED and buzzer are controlled by the user's software through the manipulation of two available GPIOs in MercuryAPI. For more details, refer to **Controlling the LED and buzzer**.

Communication between the user software and the reader is conducted through a serial port created by the operating system when the reader is connected via USB. To identify this port, please refer to section **Turning on the reader**.

Software programming and development

For software development, it is necessary to use the MercuryAPI sdk.

The latest and recommended version for development can be found at:

https://drive.google.com/drive/folders/1AOXUCQPtyAfcvyUeZf2gsHyLh3vfAEsf?usp=drive_link

In the API file package, there are numerous examples and source code in all supported languages. This development package has no additional cost and is open-source.

The complete API documentation is also available at the same link.

Demo Software – Universal Reader Assistant (URA)

The reader in Transparent Mode can be tested using Jadak's Universal Reader Assistant software. The software download and its manual are available at the links below:

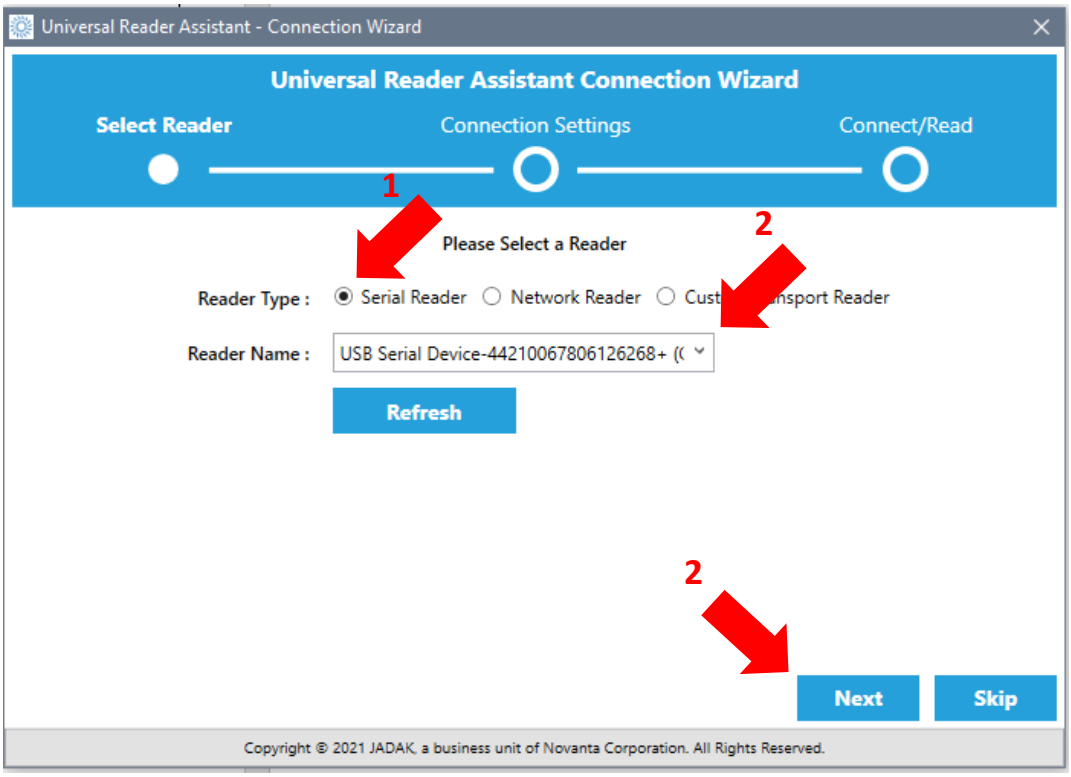
https://drive.google.com/drive/folders/1SsEdqQSkHQI_n-WqaQb7Kx8oi2INgzj2?usp=drive_link

Instructions for Demo Software

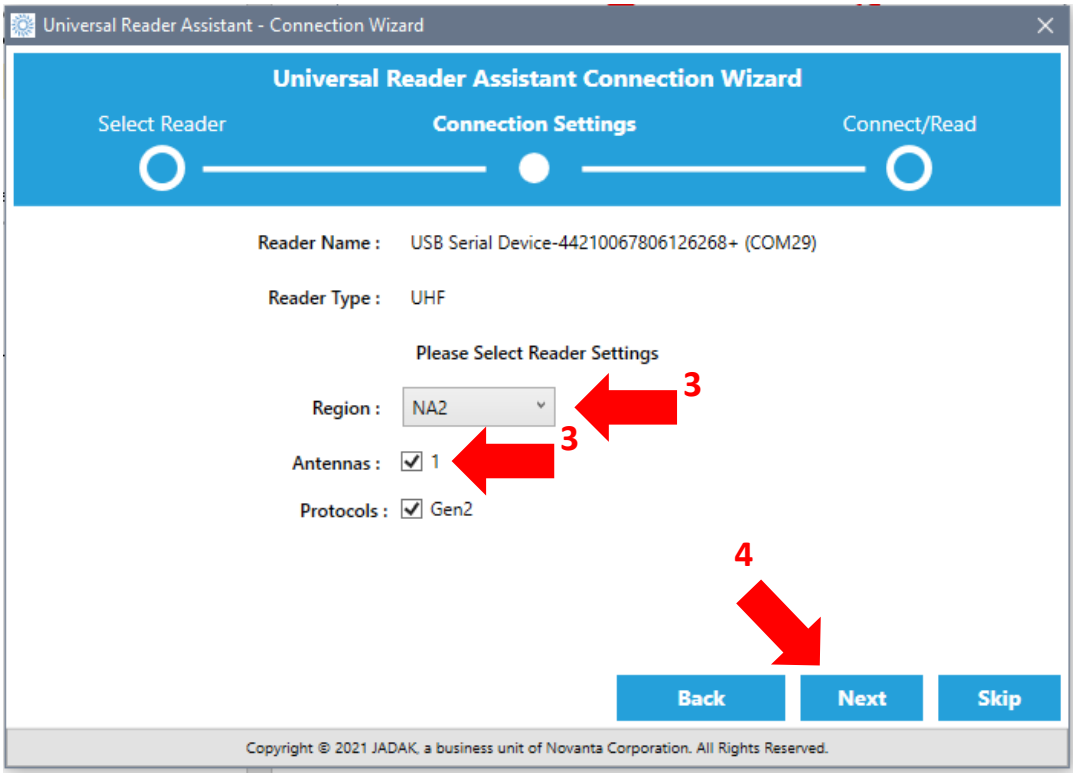
Below, there are instructions for using the Universal Reader Assistant demo software:

NOTE: Ensure that the reader is set to Transparent Mode. Please refer to **Reader in Transparent Mode**. "

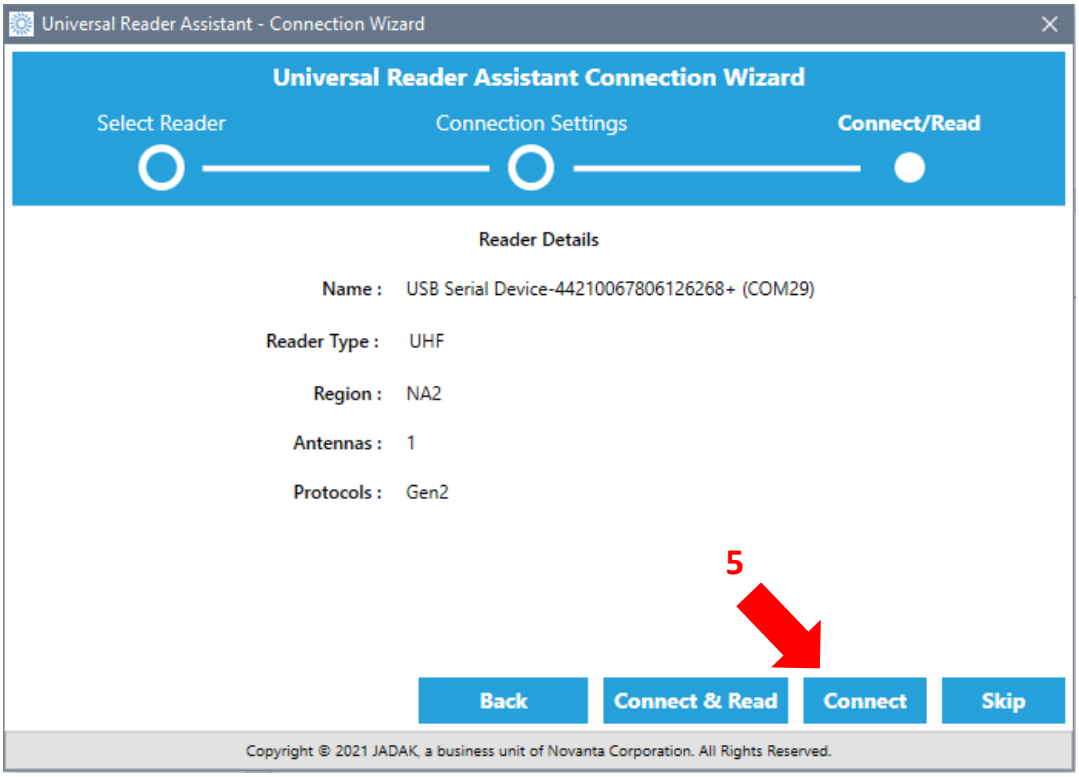
1. Launch the Universal Reader Assistant.
   The software will automatically identify the device connected to your computer
2. Proceed to the next step by clicking on 'Next'.



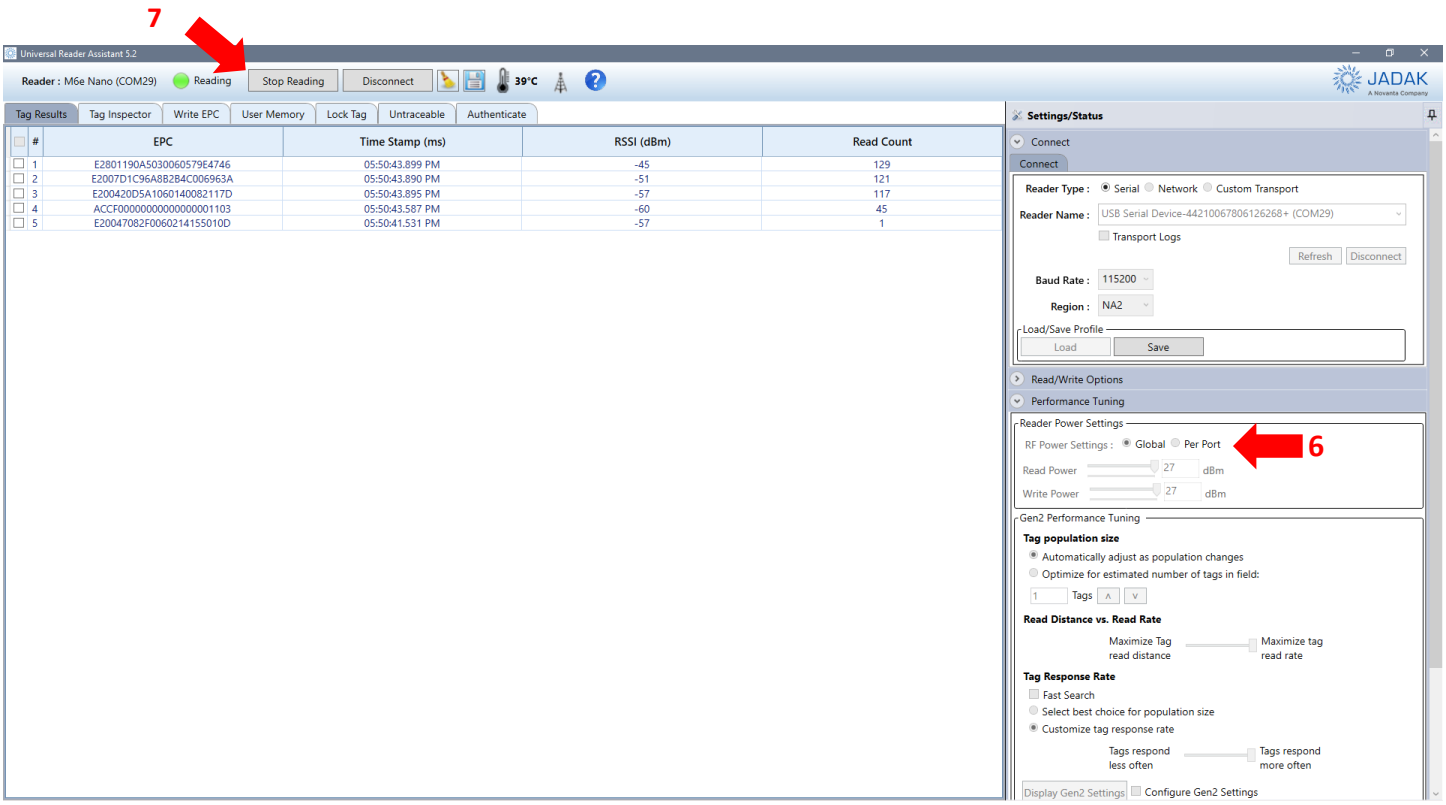
3. Select the Region and the Antenna.
4. Proceed to the next step by clicking on 'Next'.



5. Proceed to the next step by clicking on “Connect”.



6. Select the **Reader Power** in Performance Tuning.
7. Click on **Read** to start the data readings.



Controlling the LED and Beep

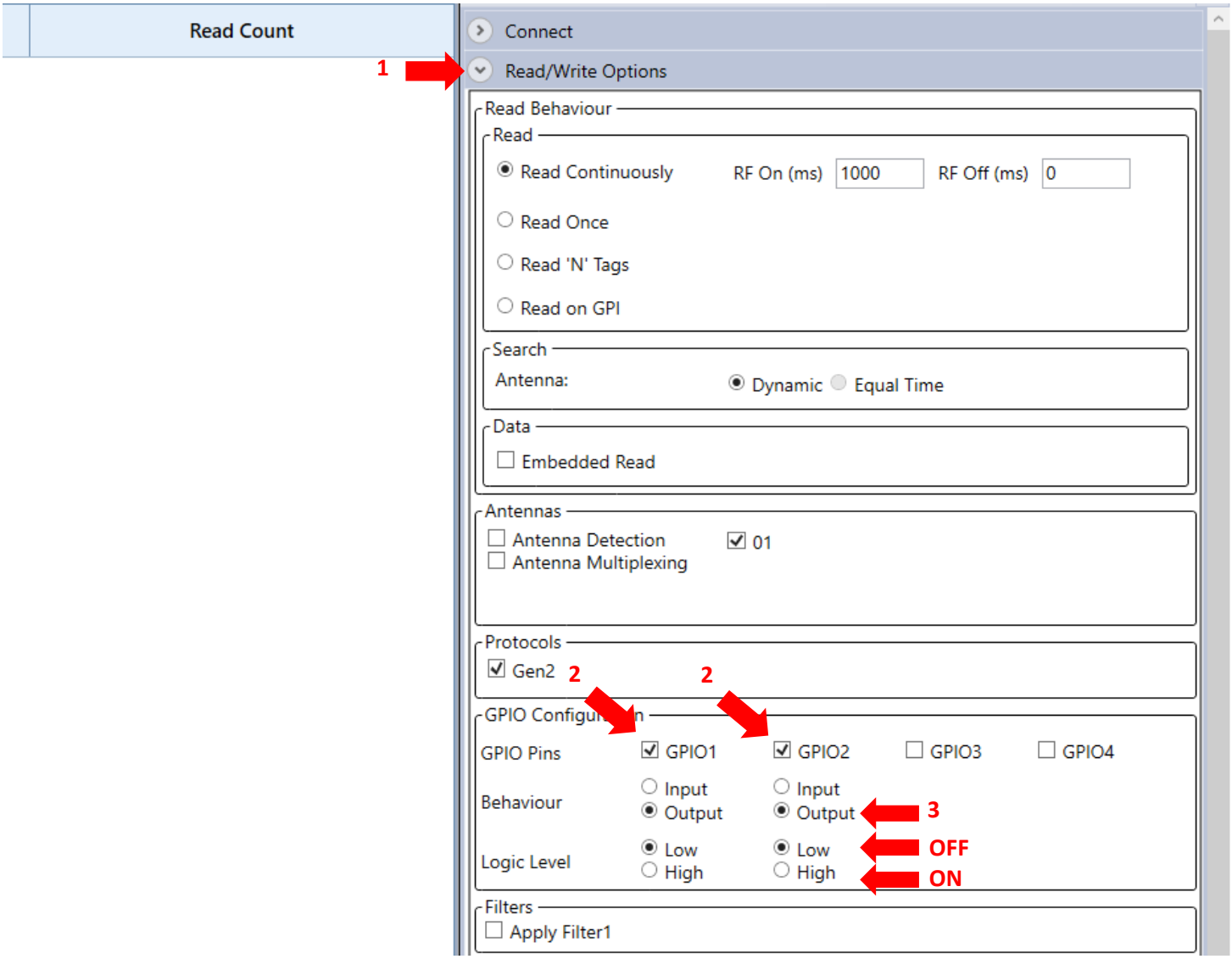
In Transparent Mode, control of the red LED and the beep should be managed by the user's software using the MercuryAPI.

Below is the usage map for the respective GPIOs:

Output	***Type feedback***	***Drive (on)***
GPIO1	Red LED	High
GPIO2	Beep	High

Below is the image demonstrating the use of the URA for controlling the red LED and beep of the reader in Transparent Mode:

1. Choose the "Read/Write Options" in order to access the configuration menu, which includes the "GPIO Configuration" settings.
2. Select GPIO1 and GPIO2, representing the LED and the buzzer, respectively.
3. Choose "Behavior" and set it to "Output".
4. The "Logic Level," when set to "Low," causes the LED or buzzer to turn off, while setting it to "High" turns them on.



NOTE: In the MercuryAPI file package for all languages, there is an example for manipulating the reader's GPIO called "**gpiocommands**."

4 Revision history

Date	Description	Revision
October 2023	Initial release.	A.0