PRODUCT DOCUMENTATION

PN7160 MINI V1 -- I2C

User Manual

Connection, configuration, software and troubleshooting

ELECHOUSE | 2026-09-08

Online documentation and downloads

Contents

1. Quick Start

ESP32-S3 quick start

Connect the module, adapt the official ELECHOUSE DetectTags example to your pins and supply, then check NFC-A tag detection.

Release 1.0 | 2026-09-08 | Example VCC: 3.3 V | I2C: 400 kHz recommended | Library reference: 3.1.3 | Serial Monitor: 115200 bit/s

Online Quick Start

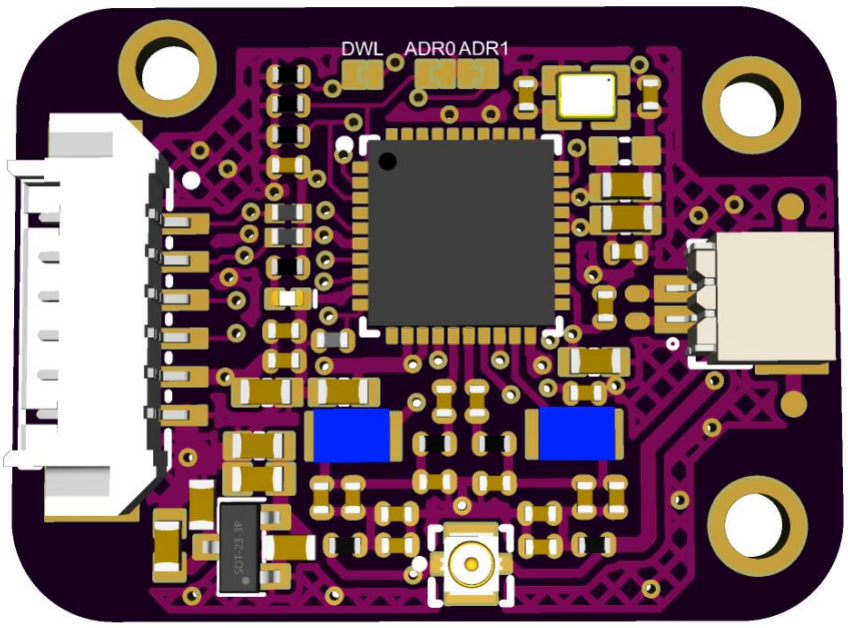
1. What you need

- PN7160 MINI V1 module and one of its included 50 × 40 mm or 25 × 10 mm antennas.
- ESP32-S3 SuperMini, a USB data cable and Arduino IDE with the Espressif ESP32 board package.
- A compatible 1.25 mm six-pin host cable or wiring harness, purchased separately.
- An NFC-A tag, such as a MIFARE card, for the first check.

Use one antenna at a time
Connect one included antenna to the ST1.0 2-pin antenna socket. Its lead is 10 cm long. The IPEX4 socket is an alternative antenna connection; never connect antennas to both sockets simultaneously.

The six-pin host cable and development board are not included. A 100 × 160 mm antenna is available separately; see the datasheet for package contents and reading-distance references.

2. Connect the module



The dot beside the host connector identifies Pin 1 (SDA).

Disconnect power before wiring. Starting at the dot, the connector signals are **SDA, SCL, IRQ, VEN, VCC, GND**.

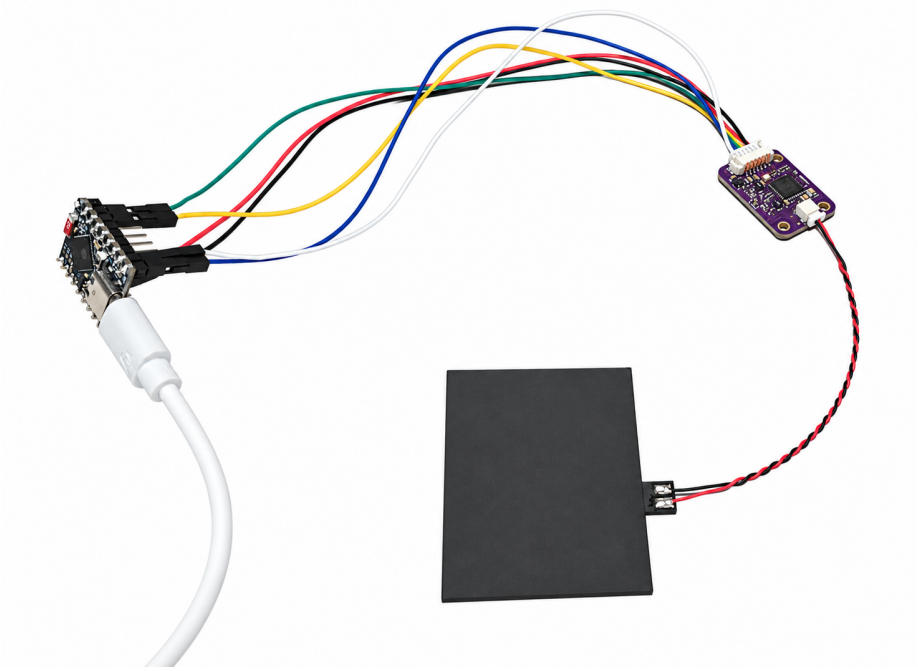
Connect by signal name, not wire color. The apparent left-to-right order is reversed when the mating plug is viewed from the opposite side. Align the keyed connector housing and do not force the plug.

Leave **ADR0** and **ADR1** open for the default 7-bit I2C address `0x28`.

Module pin	Signal	ESP32-S3 SuperMini
1	SDA	GPIO8
2	SCL	GPIO9
3	IRQ	GPIO4
4	VEN	GPIO5
5	VCC	3V3 for this procedure
6	GND	GND

Supply voltage and signal voltage are different
The module VCC operating range is 3.3-5.5 V, including supply tolerance and ripple. SDA, SCL, IRQ and VEN use 3.3 V logic; do not apply 5 V to them. This procedure uses a 3.3 V supply and its matching software preset.

Reserve at least 300 mA of supply capacity for the module; a 500 mA or higher-rated source provides additional margin. Budget the host board separately. Check that the development board's 3.3 V regulator can supply the combined load. SDA and SCL already have 4.7 kΩ pull-ups to 3.3 V.



Connection illustration. Follow the GPIO table above for the ESP32-S3; the host board and host cable shown are not included.

3. Install the official library

1. Download the ELECHOUSE library 3.1.3 ZIP used by this guide. The main library repository also provides updates and source code.
2. In Arduino IDE, select **Sketch → Include Library → Add .ZIP Library** and select the downloaded ZIP.
3. Open examples/DetectTags/DetectTags.ino, or select **DetectTags** from the library's examples menu.
4. Save a working copy for your project, then make the configuration changes below.

This is the same software library linked from the standard ELECHOUSE PN7160 product. The ZIP is fixed to source revision `a9c0f71` so the setup can be reproduced. Use this guide for the MINI's six-pin wiring and single VCC input. No separate MINI driver or NeoPixel library is required for DetectTags.

4. Configure DetectTags for the MINI

4.1 Set the pins and supply preset

Put this block at the top of your sketch, before the existing library `#include` lines. Remove any conflicting definitions in the sketch.

```
#define PN71XX_USE_SPI 0
#define PN71XX_I2C_SDA 8
#define PN71XX_I2C_SCL 9
#define PN71XX_I2C_IRQ 4
#define PN71XX_I2C_VEN 5
#define PN71XX_I2C_ADDR 0x28
#define PN71XX_I2C_CHIP_MODEL PN7160
#define PN71XX_FIXED_VBAT_3V3 1  // Module VCC = 3.3 V

#include <Wire.h>
// Keep the existing ELECHOUSE includes and the rest of DetectTags.
```

Keep the example's `PN71XX_NFC_INSTANCE(nfc)` and `pn71xxConfigureExampleTransport(nfc)` calls. They apply these settings. Explicit IRQ and VEN definitions are important because the shared example's generic ESP32-S3 defaults differ from this wiring.

Module VCC	Sketch setting	Host signals
3.3 V	`#define PN71XX_FIXED_VBAT_3V3 1`	3.3 V logic
5.0 V	`#define PN71XX_FIXED_VBAT_3V3 0`	3.3 V logic; common ground

The preset is not automatic voltage detection. For a 5 V design, connect module VCC to the 5 V supply and change the macro to `0`; do not change the signal wiring to 5 V. Completely disconnect power before changing the wiring or supply preset, then upload the matching configuration.

4.2 Set the I2C clock

In `setup()`, add the following line **after the initialization error-check block** and before the first waiting-for-card message:

```
Wire.setClock(400000UL);  // 400 kHz for normal operation
```

The library starts the I2C bus during initialization. Setting the clock after successful initialization makes the subsequent tag-reading traffic use 400 kHz. Reapply this line if your application closes and restarts the I2C bus.

5. Upload

1. Select **ESP32S3 Dev Module** in Arduino IDE.
2. For a board using the ESP32-S3 native USB connection, set **USB CDC On Boot** to **Enabled**.
3. Select the board's serial port and upload the sketch.
4. Open Serial Monitor at **115200 bit/s** and reset the host board.

If your board needs manual download mode, hold BOOT, tap RESET, start the upload and release BOOT. Firmware is uploaded to the ESP32-S3; no programming connection to the NFC module is required for this procedure.

6. Verify NFC-A operation

1. Confirm initialization completes and Serial Monitor displays a waiting-for-card message.
2. Hold an NFC-A card close to and parallel with the antenna.
3. Check the reported technology and NFC ID. DetectTags prints results to Serial Monitor; it does not provide the board-specific RGB LED indication.
4. Remove the card and confirm that detection resumes.

For ISO15693 identifier handling, see the API notes. Card data and NDEF contents require the corresponding protocol example; detecting an identifier alone does not read protected application data.

Symptom	First checks
No serial output	USB data cable, selected serial port, USB CDC setting and 115200 bit/s.
Initialization error	Connector orientation, VCC, common ground, GPIO overrides, address and IRQ/VEN connections.
Initializes but detects no card	One antenna connected, correct supply preset, card close to the antenna and no nearby metal.

Software guide | Troubleshooting | Download user manual PDF

2. Hardware Integration

PN7160 MINI V1 -- I2C

Connect power and host signals, select the I2C address, and integrate the antenna and module into your product.

Release 1.0 | 2026-09-08 | Mini V1 | I2C

Online Hardware Integration

1. Module connections

HOST PRODUCT	
3.3-5.5 V supply Stable VCC source and common ground	**Host MCU** 3.3 V I2C, IRQ input and VEN output
↓	
Six-pin host connector 5 VCC → 6 GND ↔ 1 SDA ↔ 2 SCL → 3 IRQ ← 4 VEN →	
↓	
PN7160 MINI V1 MODULE	
Single power input Onboard power conditioning; no separate antenna power connection	
↓	
NFC controller and 3.3 V host interface I2C/NCI communication, interrupt notification and enable/reset control	
ADR0 address pad | ADR1 address pad | DWL maintenance pad	
↓ select one antenna connection	
Antenna connector 1 - IPEX4 Shielded coaxial lead; cable-length sensitive	**Antenna connector 2 - 1.0 mm, 2-pin** Preferred low-loss connection
One external antenna at a time	

Functional connection diagram. Signal directions are relative to the module.

Connect VCC to a 3.3-5.5 V supply and GND to the host ground. Connect SDA and SCL to the host I2C bus, IRQ to a host input and VEN to a host output. Match the signal labels on the module; do not identify pins by cable color.

The dot beside the six-pin connector marks Pin 1 (SDA). See the connector-orientation images and pin table before connecting. A six-pin host cable is not included.

2. Power design

2.1 Supply and wiring

Item	Integration requirement
Input voltage	3.3-5.5 V at the module VCC/GND connection, including source tolerance and ripple.
Supply capacity	At least 300 mA available to the module. A 500 mA or higher-rated source provides additional margin; budget other system loads separately.
Signal levels	3.3 V host logic on SDA, SCL, IRQ and VEN, regardless of the VCC supply voltage.
Power routing	Keep VCC and GND connections short and use a stable, low-impedance supply. Check the voltage at the module connector during operation.
Decoupling	Local decoupling is built in. Any additional host-side bulk capacitance depends on the supply, cable and other loads; no universal value is specified.

Check voltage and polarity before connection
Do not exceed 5.5 V on VCC or apply 5 V to the signal pins. Startup and peak current are not specified; the recommended supply capacity is not a peak-current rating.

For approximate operating-current observations with the supplied antennas, see the datasheet current reference.

2.2 Software power configuration

Select the library power configuration for the actual supply before configuring RF operation. The following examples cover 3.3 V and 5.0 V supplies.

Supply	Arduino configuration	How to apply
3.3 V	`setPn7160FixedVbat3V3(true)`	Call before `configureSettings()`. This is used in the supplied ESP32-S3 quick-start example.
5.0 V	`setPn7160FixedVbat3V3(false)`	Call before `configureSettings()` to select the normal external-5 V configuration.

Fully disconnect and restore module power after changing the supply configuration. For other supply designs within 3.3-5.5 V, confirm the appropriate software configuration with support; see the Software Guide.

3. I2C interface

- Use 3.3 V host logic and connect the host and module grounds.
- The module includes 4.7 kΩ pull-ups on SDA and SCL to the 3.3 V logic supply. Include them when checking the combined pull-up resistance if the host also has pull-ups.
- Keep SDA and SCL wiring short and close to a ground connection. Check timing at the actual cable length and bus capacitance.
- Use **400 kHz** for normal integration. Standard-mode at 100 kHz is also supported.
- Connect IRQ to an interrupt-capable host input for data-ready notification.

Logic-level compatibility

Use the digital signal-level table to check the host interface. SDA and SCL require a HIGH input of at least 70% of the module logic supply and a LOW input no higher than 30%. At a 3.3 V logic supply these are 2.31 V and 0.99 V, respectively. Do not substitute the external VCC voltage in these calculations.

VEN has separate thresholds: HIGH from 1.1 V, LOW up to 0.4 V; its input must not exceed the actual module VCC. Continue to use a 3.3 V host output for normal connection. IRQ drives a host input; its output-level limits apply at a source or sink current below 3 mA.

These DC limits follow the PN7160 electrical specification, Rev. 4.2, Tables 48, 49 and 59. They do not replace timing checks with the final wiring and pull-ups.

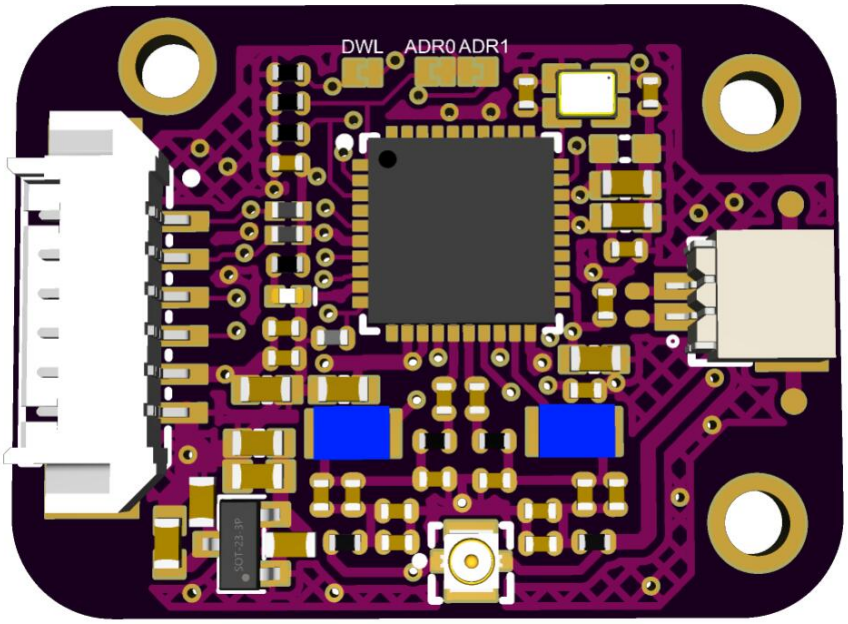
4. IRQ, VEN and DWL

Signal	Function	Connection / use
IRQ	Active-high data-ready output	Connect to a host input. Read pending NCI data until IRQ is released.
VEN	Enable/reset input	Drive low for reset and high for operation. The ELECHOUSE library handles the reset sequence during initialization.
DWL	Firmware-maintenance pad	Leave unconnected during normal use. Use only with a supported firmware-maintenance procedure.

5. I2C address selection

Use the solder pads marked **ADR0** and **ADR1** on the module. An open pad selects logic 0; bridging the two halves of a pad with solder selects logic 1. Both pads are open as supplied.

ADR1	ADR0	7-bit I2C address
Open (0)	Open (0)	**0x28 - factory default**
Open (0)	Soldered (1)	0x29
Soldered (1)	Open (0)	0x2A
Soldered (1)	Soldered (1)	0x2B



Use the ADR0 and ADR1 markings on the module.

1. Disconnect all power before soldering.
2. Bridge only the selected ADR pad or pads. Leave the DWL maintenance pad unchanged.
3. Check for unintended solder bridges.
4. Set the host software to the corresponding 7-bit address.
5. Reconnect power and initialize the module.

To restore the default address, remove the solder bridges from both ADR pads with power disconnected, then set the host address to 0x28 and power up again.

6. RF and antenna integration

Antenna connection	Connector	Selection guidance
Connector 2 - preferred	ST1.0, 1.0 mm-pitch, 2-pin wire-to-board	Use the matching two-wire antenna assembly for low connection loss.
Connector 1 - alternative	IPEX4	Use a matching shielded coaxial lead. Longer leads reduce reading distance; check performance with the final cable length.

Connect only one antenna
The two antenna connectors are alternatives. Never connect antennas to both at the same time.

6.1 Antenna options and reading-distance reference

The standard package contains both 50 × 40 mm and 25 × 10 mm antennas. The 100 × 160 mm antenna is available separately and is not included in the standard package. Antenna lead length is 10 cm.

Maximum measured reading distances for the production configuration - reference values.

Antenna size	Package / availability	S70	ISO/IEC 15693
25 × 10 mm	Included as standard	2.9 cm	6.5 cm
50 × 40 mm	Included as standard	5.0 cm	14.0 cm
100 × 160 mm	Optional; sold separately	5.6 cm	12.0 cm

These are reference observations, not guaranteed minimum distances. Results depend on the card or tag, supply, orientation, antenna connection and surrounding materials. Other antenna sizes are available by customization; contact service@elechouse.com with your installation space and application requirements.

Test conditions: 5 V supply, ST1.0 connection (connector 2), and a 10 cm antenna lead. Hold the card parallel to the antenna and determine the maximum reading distance. S70 results use an 85 × 55 mm MIFARE S70 PVC card; ISO/IEC 15693 results use an 85 × 55 mm ICODE SLIX2 card.

6.2 Installation guidance

- Use a supplied or supported antenna assembly. Avoid changing the lead length without checking reading performance.
- Keep metal, batteries, displays, ground planes and cable loops away from the active antenna area.
- Check reading performance in the final enclosure with its mounting hardware and nearby wiring in place.
- Keep antenna connections mechanically secure and leave room for the plug and cable bend.
- Disconnect power before changing antennas and follow ESD handling precautions.

7. Mechanical integration

Item	Value / integration guidance
PCB outline	26.937 × 20.066 mm nominal; standard routed-profile tolerance reference: ±0.20 mm.
PCB thickness	1.60 mm nominal; standard fabrication tolerance reference: ±0.16 mm (±10%), or 1.44-1.76 mm.
Maximum module thickness	5.5 mm. Allow additional space for plugs and cables.
Mounting holes	3 plated through-holes, Ø2.11 ± 0.10 mm.

Use the STEP model and DXF drawing for nominal mounting-hole positions and enclosure design. Allow for the outline and thickness tolerances above, together with space for the selected plugs, cable bends and insertion/removal.

Download DXF | Download STEP

8. Environment and ESD

Operate the module from -25°C to +85°C in a non-condensing environment. For storage, aim for +5°C to +30°C and 40-60% RH in ESD-protective packaging. These are recommended warehouse conditions, not extreme storage ratings.

- Handle the bare module with ESD precautions and protect it from moisture, conductive contamination and corrosive gases.
- Provide suitable host-side protection for connectors exposed outside the enclosure.
- Keep fast digital signals and switching-power circuits away from the antenna.
- Check emissions, immunity and RF performance in the completed host product; module integration does not certify the end product.

No module-level ESD rating is specified. Use ESD handling precautions and assess protection requirements for the completed host product.

9. First-start checklist

1. Check the module version and identify the connector signals from the module labels.
2. With power disconnected, connect one antenna and the six host signals.
3. Check VCC polarity, common ground and 3.3 V host logic.
4. Match the software I2C address to the ADR0/ADR1 pad settings; use 0x28 for the factory default. Set the host I2C clock to the recommended 400 kHz.
5. Select the software power configuration for the actual supply voltage.
6. Apply power and check VCC at the module connector and the SDA/SCL idle levels.
7. Run the supplied example, confirm initialization and read the firmware version.
8. Test the required card types using the selected antenna.
9. Repeat the reading test after installation in the final enclosure.

User Manual PDF | Product datasheet | Quick Start

3. Software Development

Software development guide

Use the official ELECHOUSE PN7160 software library for Arduino and ESP32. Configure the MINI's pins and supply, then build on the standard tag-reading examples.

Release 1.0 | 2026-09-08 | Library reference: 3.1.3 | I2C / NCI | Example host: ESP32-S3

Online Software Development

1. Software resources

The MINI uses the same ELECHOUSE Arduino library linked from the standard PN7160 product. No separate MINI-specific library is required. Use the wiring in this documentation for the MINI board.

Platform	Software path	Integration note
ESP32 / ESP32-S3 / ESP32-C3	ELECHOUSE Arduino library and Arduino-ESP32	Set the GPIOs explicitly; this guide uses an ESP32-S3 with SDA8, SCL9, IRQ4 and VEN5.
Other Arduino-compatible hosts	ELECHOUSE Arduino library	Verify RAM capacity, I2C support and GPIO behavior on the selected board. Use 3.3 V logic or suitable level translation.
Raspberry Pi / Linux	NXP linux_libnfc-nci	Separate middleware installation and board-specific configuration are required; Arduino macros do not configure the Linux stack.

For Linux middleware installation background, use the ELECHOUSE PN7160 I2C Linux guide. Its physical wiring describes a different module; use this MINI's six-pin pinout instead and confirm a power configuration appropriate to your middleware and module supply before enabling RF.

2. Install and select an example

1. Install the board package for your host in Arduino IDE.
2. Download the ELECHOUSE library 3.1.3 ZIP and install it with **Sketch → Include Library → Add .ZIP Library**.
3. Open DetectTags for the first NFC-A check.
4. Apply the MINI pin, power and clock settings, then select the board and upload.

The download above is fixed to source revision `a9c0f71`, library version 3.1.3. Check the installed version in `library.properties` and review changes in the main repository before updating a production application.

3. Configure the pins and power

3.1 Standard library examples

Define the following before the library's shared example-transport header. These are ESP32-S3 GPIO numbers, not module connector pin numbers.

```
#define PN71XX_USE_SPI 0
#define PN71XX_I2C_SDA 8
#define PN71XX_I2C_SCL 9
#define PN71XX_I2C_IRQ 4
#define PN71XX_I2C_VEN 5
#define PN71XX_I2C_ADDR 0x28
#define PN71XX_I2C_CHIP_MODEL PN7160
#define PN71XX_FIXED_VBAT_3V3 1  // Module VCC = 3.3 V

#include <Electroniccats_PN7150.h>
#include <Electroniccats_PN71xx_ExampleTransport.h>
```

Keep the example's `pn71xxConfigureExampleTransport(nfc)` call. It applies these choices before initialization. Leave ADR0 and ADR1 open for address `0x28`; change the address macro if you solder the address pads for another address.

3.2 Power presets

Module VCC	Shared example macro	Direct API
3.3 V	`PN71XX_FIXED_VBAT_3V3 1`	`nfc.setPn7160FixedVbat3V3(true)`
5.0 V	`PN71XX_FIXED_VBAT_3V3 0`	`nfc.setPn7160FixedVbat3V3(false)`

Select the preset for the actual supply
The module does not automatically select the power preset. Apply the correct setting before `configureSettings()`. When using `begin()`, select the preset before calling `begin()`, because that method applies the settings internally. Power down before changing the supply or configuration.

The specified input range is 3.3-5.5 V at VCC, including tolerance and ripple. The examples above cover nominal 3.3 V and 5.0 V designs. This is independent of the 3.3 V signal interface: do not connect 5 V to SDA, SCL, IRQ or VEN.

3.3 Custom applications

```
Electroniccats_PN7150 nfc(4, 5, 0x28, PN7160);
//                        IRQ, VEN, address, device

nfc.setI2CPins(8, 9);  // SDA, SCL
```

The library retains the header and class name `Electroniccats_PN7150`. Explicitly pass `PN7160` in the constructor; the shorter three-argument constructor selects a different controller path.

4. Initialization and I2C clock

Use **400 kHz** for normal operation. A 100 kHz bus is also supported. The library initializes `Wire` inside `connectNCI()`; set the operating clock after the connection succeeds.

```
nfc.setI2CPins(8, 9);
nfc.setPn7160FixedVbat3V3(true);  // VCC = 3.3 V; false for 5.0 V

if (nfc.connectNCI()) {
  // Stop here and report the connection error.
  return;
}
Wire.setClock(400000UL);

if (nfc.configureSettings()) return;
if (nfc.configMode()) return;
if (nfc.startDiscovery()) return;
// Initialization is now complete.
```

This is an initialization fragment; include `Wire.h` and the library header, define the controller instance, and prevent the application loop from accessing NFC after an initialization failure. The API summary includes a complete basic sketch.

When using the existing DetectTags helper, add `Wire.setClock(400000UL)` after its initialization error-check block. If recovery code closes and restarts `Wire`, set the clock again after the new connection.

5. Choose the example for your task

Repository example	Purpose	Application consideration
DetectTags	Discover tags and report identifiers.	Start with NFC-A; see identifier length handling for NFC-V.
NDEFReadMessage	Read an NDEF message.	Use a supported tag containing valid NDEF data.
NDEFSendMessage	Write an NDEF message.	Use a writable test tag; writing replaces existing content.
MifareClassic_read_block	Read a MIFARE Classic block.	Valid sector authentication keys are required.
ISO15693_read_block	Read ISO15693 memory blocks.	Check the card's block layout and supported commands.

Other examples are listed in the repository examples directory. Apply the same MINI transport and supply settings to each example. Example availability does not replace testing with the intended card and application.

6. Application handling and recovery

- Use a finite `isTagDetected(timeoutMs)` timeout if your application must service other tasks.
- Read `remoteDevice` only after successful detection, and use the identifier accessor for the detected technology.
- `waitForTagRemoval()` blocks while checking tag presence. Plan for this behavior in applications with timing or user-interface requirements.
- Check `reset()`: it returns `true` when discovery restarts successfully, unlike the initialization methods that use zero for success.
- If recovery fails, stop tag operations, verify power and wiring, then reset the controller through VEN and repeat initialization with bounded retries.
- Use the no-argument `configureSettings()` call; see the library 3.1.3 compatibility note.

Before deploying an application, check cold startup, repeated card presentation and removal, and recovery using the final supply, antenna, cable routing and enclosure. For support, record the library version, host board, GPIO mapping, VCC, preset and the first reported error.

API summary | Troubleshooting | Download user manual PDF

4. API Reference

API reference summary

The core calls for connecting the module and detecting NFC tags, with links to the official library's detailed reference.

Release 1.0 | 2026-09-08 | Library reference: 3.1.3 | Device selector: PN7160 | Full API: API.md on GitHub

Online API Reference

The full library API reference covers multiple controller and transport configurations. The settings below apply to PN7160 MINI V1 over I2C and the library 3.1.3 source ZIP, revision `a9c0f71`. For the standard repository example, start with Quick Start.

1. Constructor and transport

```
#include <Wire.h>
#include <Electroniccats_PN7150.h>

Electroniccats_PN7150 nfc(4, 5, 0x28, PN7160);
//                      IRQ, VEN, address, device
```

The library's class name is `Electroniccats_PN7150`, including when used with PN7160. Pass the explicit `PN7160` selector; do not omit the fourth constructor argument.

`nfc.setI2CPins(8, 9)` selects SDA8 and SCL9 on ESP32-S3. Call it before `connectNCI()`. The address is 7-bit; the factory-open ADR0 and ADR1 pads select `0x28`.

2. Initialization

Call	Purpose and order	Return behavior in 3.1.3
`setPn7160FixedVbat3V3(bool)`	Select the module supply preset before applying settings: `true` for 3.3 V, `false` for 5.0 V.	No return value.
`connectNCI()`	Initialize the bus, reset the controller and establish NCI communication.	0 = success; 1 = failure.
`configureSettings()`	Apply settings after connecting and selecting the supply preset.	0 = success; 1 = failure.
`configMode()`	Configure the selected operating mode; reader/writer is the normal starting mode.	0 = success; 1 = failure.
`startDiscovery()`	Begin NFC discovery after mode configuration.	0 = success; 1 = failure.
`begin()`	Shortcut that calls the four initialization stages above. Set the supply preset before using it.	0 = success; 1 = failure.

Set `Wire.setClock(400000UL)` after `connectNCI()` or `begin()` succeeds. These methods initialize the I2C bus. Set the clock again if your recovery code closes and restarts the bus.

Library 3.1.3: use the no-argument settings call
Use `configureSettings()`. Do not use the custom-UID overload `configureSettings(uid, length)` with this version: it has a known buffer-handling issue that can read beyond the provided UID data. The normal initialization path shown here uses the no-argument call.

The software power preset does not detect the applied voltage. VCC supports 3.3-5.5 V, while the module's host signals use 3.3 V logic. See supply configuration before changing power sources.

3. Tag detection and recovery

Call or object	Use	Important behavior
`isTagDetected(timeoutMs)`	Wait for tag activation, for example `isTagDetected(250)`.	`true` means a tag was detected; `false` does not itself distinguish timeout from a communication error.
`remoteDevice`	Read the detected technology, protocol and identifier.	Access only after successful detection; use the correct accessor for that technology.
`waitForTagRemoval()`	Perform presence checks until the tag is no longer present.	Blocking operation; no return value.
`reset()`	Reconfigure and restart the discovery flow.	`true` = success; `false` = failure. This differs from the initialization return convention.
`stopDiscovery()`	Request that discovery stops.	Version 3.1.3 returns 0 without reporting controller acknowledgement; do not treat this return as proof of a completed state change.
`getFirmwareVersion()`	Read the firmware information retained from initialization.	Useful for application logs and support requests.
`closeCommunication()`	Close the communication path before shutdown or reconfiguration.	Reinitialize the bus and controller before resuming NFC operations.

If discovery recovery fails, stop tag operations, verify the supply and bus, and perform a controller reset through VEN before reconnecting. Use bounded retries rather than an uncontrolled reset loop.

4. Read the correct identifier length

Detected technology	Data accessor	Length
NFC-A	`remoteDevice.getNFCID()`	`remoteDevice.getNFCIDLen()`
NFC-V / ISO15693	`remoteDevice.getID()`	8 bytes, after checking that the activated technology is NFC-V.

These accessors return pointers. Do not use `sizeof(pointer)` as the identifier length. In library 3.1.3, the NFC-V display branch in DetectTags uses this pattern, so its printed identifier may be incomplete. For a working copy of that example, print all eight NFC-V identifier bytes and keep the byte order consistent with your application.

```
if (nfc.remoteDevice.getModeTech() == nfc.tech.PASSIVE_NFCV) {
  const unsigned char* id = nfc.remoteDevice.getID();
  for (uint8_t i = 0; i < 8; ++i) {
    if (id[i] < 0x10) Serial.print('0');
    Serial.print(id[i], HEX);
    if (i < 7) Serial.print(':');
  }
  Serial.println();
}
```

The snippet prints the identifier in the order returned by the library. A displayed identifier is not an authentication result or a complete read of the card's protected data.

5. Basic ESP32-S3 sketch

This basic NFC-A identifier example uses SDA8, SCL9, IRQ4, VEN5, address `0x28` and a 3.3 V module supply. For 5.0 V VCC, change the preset argument to `false`; the GPIOs remain 3.3 V logic.

```
#include <Wire.h>
#include <Electroniccats_PN7150.h>

Electroniccats_PN7150 nfc(4, 5, 0x28, PN7160);
bool readerReady = false;

void setup() {
  Serial.begin(115200);
  delay(1000);
  nfc.setI2CPins(8, 9);
  nfc.setPn7160FixedVbat3V3(true);  // Module VCC = 3.3 V

  if (nfc.connectNCI()) {
    Serial.println("NCI connection failed");
    return;
  }
  Wire.setClock(400000UL);

  if (nfc.configureSettings() || nfc.configMode() || nfc.startDiscovery()) {
    Serial.println("NFC initialization failed");
    return;
  }
  readerReady = true;
  Serial.println("Present an NFC-A tag");
}

void loop() {
  if (!readerReady) {
    delay(100);
    return;
  }
  if (!nfc.isTagDetected(250)) return;

  if (nfc.remoteDevice.getModeTech() == nfc.tech.PASSIVE_NFCA) {
    const unsigned char* id = nfc.remoteDevice.getNFCID();
    const uint8_t length = nfc.remoteDevice.getNFCIDLen();
    Serial.print("NFC-A ID: ");
    for (uint8_t i = 0; i < length; ++i) {
      if (id[i] < 0x10) Serial.print('0');
      Serial.print(id[i], HEX);
      if (i + 1 < length) Serial.print(':');
    }
    Serial.println();
  }

  nfc.waitForTagRemoval();
  readerReady = nfc.reset();
  if (!readerReady) Serial.println("Discovery restart failed");
}
```

If initialization or discovery restart fails, this sketch stops NFC activity. Correct the wiring or supply issue and restart the host; add application-specific recovery and user feedback before deployment.

6. Further APIs

For NDEF and card memory operations, follow the corresponding repository example and the full API reference. Use writable test cards for write operations, and check authentication, memory bounds and card-specific commands.

Low-level `readerTagCmd()` does not take a reply-buffer capacity argument. Allocate and validate buffers for the intended command and response; do not pass unchecked application data to the low-level interface.

Full library API | Software guide | Download user manual PDF

5. Troubleshooting and FAQ

PN7160 MINI V1 -- I2C

Start with power and wiring, then isolate NCI communication, controller configuration, RF field generation and tag interoperability.

Release 1.0 | 2026-09-08 | Example host: ESP32-S3 | Default address: 0x28 | Quick-start VCC: 3.3 V

Online Troubleshooting and FAQ

Fastest diagnostic path
Record exactly which stage fails: power-up, `connectNCI()`, `configureSettings()`, `configMode()`, `startDiscovery()`, tag activation or application data exchange.

1. Symptom matrix

Symptom	Likely causes	Checks and action
No power indication	No VCC, reversed connector, open ground	Measure VCC at the six-pin host connector and confirm ground continuity. The dot beside the connector marks Pin 1 (SDA); verify the signal order.
I2C bus held low	SDA/SCL swapped, connector reversed, pull-up conflict, module in reset, damaged host pin	Disconnect other bus devices. Check that SDA/SCL return to approximately 3.3 V when idle. Each signal already has a 4.7 kΩ pull-up; account for any additional host-side pull-ups.
No response at 0x28	Address pads changed, VEN low, bad wiring, startup timing	For the default address, leave both ADR0 and ADR1 open (0). If either pad is soldered (1), use the corresponding address from I2C address selection. Pulse VEN, wait for startup and run the PN7160 NCI initialization sequence.
`connectNCI()` fails	Power/reset/I2C/IRQ problem or stale controller state	Call `Wire.end()`, drive VEN low, restart Wire, drive VEN high, wait, then retry at a limited rate.
`configureSettings()` fails	Wrong controller selection, wrong power preset or NCI state	Use the four-argument constructor with `PN7160`. Confirm that the power preset matches VCC.
Initialization passes but no tag is detected	No antenna, wrong RF connector, both antenna paths connected, antenna mismatch, power preset mismatch, metal detuning	Connect exactly one supplied or supported antenna, preferably through antenna connector 2 (1.0 mm 2-pin); remove nearby metal and test with a known NFC-A card.
Read distance is short or orientation-sensitive	Small antenna, tag size/type, metal, cable loop, enclosure or antenna mismatch	Test in free space, keep the tag parallel and compare against the read-distance reference for the same antenna and card type. Retest in the final assembly.
Tag is detected and immediately lost	Supply droop, RF detuning, application reset loop, protocol-specific presence check	Log supply and IRQ, distinguish RF loss from software reset, and capture the complete serial sequence.
One card works but another does not	Different NFC technology/protocol or antenna coupling	Log technology and protocol. Test NFC-A, ISO-DEP and ISO15693 separately with known samples.
Tags are read but no status LED changes	The standard DetectTags example reports through Serial Monitor	Check Serial Monitor at 115200 bit/s. This example does not control the host board's status LED.

Symptom	Likely causes	Checks and action
Works after reset but not after cold power-up	Supply sequencing, insufficient startup wait or host pin state during ramp	Log the failing software stage, measure VCC before and after startup with the available meter, verify VEN/IRQ logic states and check the startup delay.

2. 3.3 V setup checklist

- VCC connected to a stable 3.3 V source.
- Common ground between host and module.
- SDA GPIO8, SCL GPIO9, IRQ GPIO4 and VEN GPIO5 on the supplied ESP32-S3 example.
- 3.3 V logic on SDA, SCL, IRQ and VEN; do not apply 5 V to these signals.
- 7-bit I2C address 0x28 with both ADR0 and ADR1 solder pads open (0).
- Recommended I2C clock: 400 kHz. Apply the same setting after restarting the bus.
- Four-argument PN7160 constructor.
- `setPn7160FixedVbat3V3(true)` before `configureSettings()`.
- A known-good antenna and NFC-A/MIFARE test card.

Use one antenna path
Antenna connector 2 (1.0 mm 2-pin, preferred) and antenna connector 1 (IPEX4) are alternatives. Never connect antennas to both simultaneously. Longer IPEX4 coaxial leads can reduce read distance.

The standard package includes 50 × 40 mm and 25 × 10 mm antennas with 10 cm leads; a six-pin host cable is not included. A 100 × 160 mm antenna is available separately. The read-distance reference covers all three antenna sizes with the production module configuration.

3. Frequently asked questions

Which I2C clock should I use?

Use 400 kHz for normal integration. Standard-mode at 100 kHz is also supported and can help diagnose wiring or timing issues. Keep SDA/SCL wiring short and apply the same clock setting after restarting the bus. See the I2C integration requirements.

Can I use a generic I2C scanner?

It can be a wiring clue, but it is not the final functional test. PN7160 communication is IRQ/NCI driven and controller state affects when data is available. Use a complete NCI connection and firmware-version read to confirm operation.

Do I need external SDA/SCL pull-ups?

Normally no. SDA and SCL each have a built-in 4.7 kΩ pull-up to 3.3 V. Calculate the combined resistance when the host or another bus device also contains pull-ups. Do not pull these signals up to 5 V.

How do I change the I2C address?

The board labels are ADR0 and ADR1. An open pad selects 0; bridging its two contacts with solder selects 1. Both pads are open by default, selecting 7-bit address `0x28`. Disconnect power before soldering, then update the host software and power-cycle the module. See the address table and soldering instructions.

Can I power the Mini V1 from 5 V?

Yes. The VCC input operating range is 3.3-5.5 V at the module connector, including supply tolerance and ripple; do not exceed 5.5 V. The signal pins remain 3.3 V logic. Select the software power preset that matches the actual supply. Reserve at least 300 mA of source capacity for the module at its connector; a 500 mA or higher-rated source provides additional margin. See the power-supply requirements. The quick-start guide uses the 3.3 V configuration.

Why does the example class say PN7150?

The library retains its upstream class/header name for compatibility. Pass `PN7160` as the explicit device selector.

Can the read-range number be guaranteed?

The published read distances are maximum measured reference values at 5 V, using antenna connector 2 (ST1.0, 1.0 mm 2-pin), a 10 cm antenna lead and a card held parallel to the antenna. The test cards are an 85 × 55 mm ICODE SLIX2 ISO/IEC 15693 card and an 85 × 55 mm MIFARE S70 PVC card. These distances are not a guaranteed minimum for every card or installation. Actual distance depends on the antenna, card/tag, orientation, supply, enclosure and nearby metal. Check performance in your final assembly.

How should I protect the module against ESD?

Handle the bare module with ESD precautions and keep it in ESD-protective packaging when not in use. No module-level ESD rating is specified. Provide protection appropriate to the completed host product, particularly for exposed connectors.

4. Information to include in a support request

- Module front/back photos and hardware marking.
- Host board and operating-system/core version.
- Exact wiring table and measured VCC.
- Antenna size, lead length, RF connector and installation photo.
- ADR0/ADR1 pad state and configured address.
- Configured I2C clock and host cable length.
- Library version/commit and complete sketch.
- Full serial log from power-up through failure.
- Tag/card type and whether it works with another reader.
- Whether the failure occurs on cold power-up, reset or both.

User Manual PDF | Return to quick start | Hardware guide