

XE212 SDK

User Guide

XE212 SDK

User Guide

Author: Acconeer AB

Version: a2-v1.2.0

Acconeer AB Nov 03, 2026

© 2026 by Acconeer AB - All rights reserved

Contents

1 Introduction 4
2 Build the SDK 5
3 Install Python & Packages 6
3.1 Linux 6
3.1.1 Install Dependencies 6
3.1.2 Install UV 6
3.2 Windows 6
3.2.1 Install UV 6
4 Using the XE212 7
4.1 Upgrade the XE212 Firmware 7
4.2 Flash an Application to the NVM 7
4.3 Reset the A212 7
4.4 Erase the NVM 7
4.5 Start an Application over DFU 7
4.6 Debug the A212 with GDB 8
4.6.1 Connect the Debug Pins 8
4.6.2 Start the GDB Server 8
4.6.3 Load and Debug an Application 8
4.6.4 Attach to a Running Application 8
4.6.5 Monitor Commands 9
4.6.6 Troubleshooting 9
4.7 Use the A212 with an External Host 9
4.7.1 Disable the NVM 9
4.7.2 Enable the NVM 9
5 Disclaimer 10

© 2026 by Acconeer AB - All rights reserved

Acconeer SDK Documentation Overview

To better understand which SDK document to use, a summary of the documents is shown in the table below. More general information about Pulsed Coherent Radar, PCR, can be found at docs.acconeer.com.

Name	Description	When to use
RSS API documentation (html)		
rss_api	The complete C API documentation.	• RSS application implementation • Understanding RSS API functions
User guides (PDF)		
A212 Sensor FW	Describes how to develop SW on the A212	• Developing SW inside the A212 sensor
A212 SDK VS Code	Describes how to build the A212 SDK and XE212 SDK, and use the XE212 EVK, in a VS Code dev container	• Working with A212 SDK or XE212 SDK in VS Code
XE212 SDK	Describes how to develop SW on the A212 using the XE212 EVK	• Working with XE212 SDK
Reference Manuals (PDF)		
A212 Technical	Complete information on A212 memory and peripherals	• Advanced development of SW inside the A212 sensor • Development of driver SW inside the A212 sensor
Application Notes (PDF)		
DFU Protocol	Defines the protocol used to communicate with the A212, either with UART or SPI in DFU mode	• Development of bootstrap software for external host
NVM Image Format	Defines the format of the image data to stored in external NVM	• Development of A212 software that loads from external NVM
Exploration Server Protocol	Defines the request/response protocol between clients and the Exploration Server	• Development of client software for the exploration server
Readme (txt)		
README	Various target specific information and links	• After SDK download
BUILDINFO	Compiler information	• To understand pre-compiled code

Table 1: SDK document overview.

© 2026 by Acconeer AB - All rights reserved

1 Introduction

The XE212 EVK is an evaluation kit designed for the following purposes:

- Serving as a platform for on-sensor development of applications running on the A212
- Working with the Acconeer Exploration Tool to evaluate use cases and radar performance

To use the XE212 EVK with the Acconeer Exploration Tool, flash it with the exploration server software. Refer to the Acconeer Exploration Tool documentation for details.

For on-sensor development, `xe212tool.py` (see Section 4) loads, flashes, starts and debugs applications on the A212.

© 2026 by Acconeer AB - All rights reserved

2 Build the SDK

The applications in the SDK are built with Visual Studio Code (VS Code) in a *dev container*, which contains the toolchain and everything else that is needed. Nothing except Docker and VS Code has to be installed on the computer.

How to set up Docker and VS Code on Linux and Windows, and how to build, load and flash applications from VS Code, is described in the **A212 SDK VS Code User Guide** in the `doc/` folder of the SDK.

© 2026 by Acconeer AB - All rights reserved

3 Install Python & Packages

The tools that communicate with the A212 on the XE212 EVK are written in Python. They are run with `uv`, which installs Python and the packages the tools need. For more information, see the uv documentation at `https://docs.astral.sh/uv/`.

3.1 Linux

3.1.1 Install Dependencies

Ubuntu is used as the reference distribution in this guide. Other Linux distributions may use different package managers, so installation commands may need to be adapted accordingly.

```
$> sudo apt update
$> sudo apt install libusb-1.0-0
```

3.1.2 Install UV

Download and run the installation script:

```
$> wget -qO- https://astral.sh/uv/install.sh | sh
```

3.2 Windows

3.2.1 Install UV

Install `uv` with `winget` in PowerShell:

```
PS> winget install --id=astral-sh.uv -e
```

© 2026 by Acconeer AB - All rights reserved

4 Using the XE212

`xe212tool.py` is a command-line tool for the XE212 EVK. It flashes and erases the NVM, resets the A212, starts applications over DFU, serves GDB for debugging, and upgrades and configures the XE212 EVK.

Note: Only one program can access the XE212 USB interface at a time. Close the Acconeer Exploration Tool, and any other program that communicates with the board, before using `xe212tool.py`, and vice versa. Flashing or upgrading while another program holds the USB interface can fail and leave the board in bootloader mode. If that happens, close all programs that access the board and run the upgrade or flashing again.

Commands that start an application on the A212 can stream its debug log, from `UART1` on `PIN2`, when `--log-uart` is given. The log uses 115200 baud, 8 data bits, no parity and one stop bit (8N1), and is streamed until Ctrl+C is pressed.

With more than one XE212 EVK connected, select one with `--serial` **`<serial number>`**. The `list` command shows the connected XE212 EVKs and their serial numbers.

4.1 Upgrade the XE212 Firmware

`xe212tool.py` requires the XE212 EVK firmware of the same SDK version. The `upgrade` command checks the firmware version and upgrades it if needed.

```
$> uv run script/xe212tool.py upgrade
```

4.2 Flash an Application to the NVM

The `nvm` command flashes an application to the NVM on the XE212 EVK and resets the A212, which then starts it. The A212 boots from the NVM in two stages, so the NVM bootloader is flashed together with the application. The `mx25r6435f_nvm_qspi_boot.elf` bootloader loads the application from the NVM on the XE212 EVK.

Command Line: `uv run script/xe212tool.py nvm` **`<bootloader.elf> <application.elf>`** [`--log-uart`]

Add `--log-uart` to stream the UART log after the reset.

Example:

```
$> uv run script/xe212tool.py nvm out/mx25r6435f_nvm_qspi_boot.elf out/example_diagnostics.elf --log-uart
```

4.3 Reset the A212

The `reset` command resets the A212, which then starts the application in the NVM again.

Command Line: `uv run script/xe212tool.py reset` [`--log-uart`]

Add `--log-uart` to stream the UART log after the reset.

Example:

```
$> uv run script/xe212tool.py reset --log-uart
```

4.4 Erase the NVM

The `nvm_erase` command erases the application in the NVM, so that the A212 no longer starts it.

```
$> uv run script/xe212tool.py nvm_erase
```

4.5 Start an Application over DFU

The `dfu` command loads an application from the computer to the A212 RAM over DFU and starts it, without using the NVM.

Command Line: `uv run script/xe212tool.py dfu` **`<application.elf>`** [`--dfu-interface spi|uart`] [`--log-uart`]

The application is loaded over SPI unless `--dfu-interface uart` is given. Add `--log-uart` to stream the UART log after the application is started.

Example:

```
$> uv run script/xe212tool.py dfu out/example_diagnostics.elf --log-uart
```

© 2026 by Acconeer AB - All rights reserved

4.6 Debug the A212 with GDB

The `gdb_server` command serves GDB for the A212 APS core over the cJTAG debug interface of the A212. Debug with a GDB that supports RISC-V, for example `gdb-multiarch` from the Ubuntu package of the same name. The dev container of the SDK includes `gdb-multiarch`.

4.6.1 Connect the Debug Pins

The XE212 drives cJTAG on the A212 pins `PIN5` and `PIN8`. Before the GDB server is started, connect them with wires to the A212 debug pins on the XE212 EVK board:

From	To
PIN5	TMSC
PIN8	TCKC

The application must not use `PIN5` or `PIN8` while it is debugged.

4.6.2 Start the GDB Server

Command Line: `uv run script/xe212tool.py gdb_server [--attach-to-running-target] [--log-uart] [--port <port>]`

The GDB server resets the A212 and halts it at the boot ROM reset vector, so that nothing runs until GDB continues. Add `--attach-to-running-target` (`-a`) to leave the A212 running instead, for example to debug an application started with the `dfu` or `nvm` command. Add `--log-uart` to also stream the UART log. The GDB server listens on TCP port 3333, or on the port given with `--port`.

The GDB server runs until Ctrl+C is pressed, and GDB can connect and disconnect as often as needed meanwhile. The GDB server holds the XE212 USB interface, so stop it before other `xe212tool.py` commands are used.

Example:

```
$> uv run script/xe212tool.py gdb_server --log-uart
```

4.6.3 Load and Debug an Application

Load the application to RAM, start it from the boot ROM and run to `main`:

```
$> gdb-multiarch out/example_diagnostics.elf
(gdb) target remote :3333
(gdb) monitor reset halt
(gdb) load
(gdb) monitor reset debug
(gdb) maintenance flush register-cache
(gdb) break main
(gdb) continue
```

`monitor reset debug` resets the A212 and halts it in the boot ROM. When the A212 continues, the boot ROM boots as usual, but starts the loaded application instead of the one in the NVM. The application starts at `pc`, which `load` sets to its start address. GDB does not know that `monitor reset debug` changed the registers, so flush its register cache after it.

When GDB detaches or quits, the breakpoints are removed and the application runs on.

4.6.4 Attach to a Running Application

Start the GDB server with `--attach-to-running-target`, and connect GDB with `target extended-remote`, which does not halt the A212:

```
$> uv run script/xe212tool.py gdb_server --attach-to-running-target --log-uart
```

```
$> gdb-multiarch out/example_diagnostics.elf
(gdb) target extended-remote :3333
(gdb) attach 1
(gdb) bt
(gdb) detach
```

`attach 1` halts the A212 where it is, and `detach` lets it run on with GDB still connected.

© 2026 by Acconeer AB - All rights reserved

4.6.5 Monitor Commands

The GDB server has the following monitor commands. `monitor help` lists them in GDB.

Command	Description
`monitor reset [halt]`	Reset the A212 and halt at the boot ROM reset vector.
`monitor reset run`	Reset the A212 and let the boot ROM boot normally.
`monitor reset debug`	Reset the A212 and halt at the boot ROM reset vector, with the boot ROM set to start the loaded application.

After `monitor reset debug`, every reset starts the loaded application, until the application, `monitor reset halt`, `monitor reset run` or a power-on reset clears it.

4.6.6 Troubleshooting

`Did not get expected JTAG ID` means that the A212 does not answer on cJTAG. Check that `PIN5` is connected to `TMSC` and `PIN8` to `TCKC`, and that the application does not use them.

4.7 Use the A212 with an External Host

When an external host communicates with the A212 over SPI, the onboard NVM must be disabled. The A212 and the onboard NVM share the SPI lines, and the XE212 EVK cannot switch them between the A212 and the NVM, so an enabled NVM disturbs the communication.

The NVM setting is stored in the XE212 EVK, so it **persists across power cycles** and only has to be changed once.

Note: Make sure that the SPI voltage levels of the external host are compatible with the XE212 EVK (3.3 V) before connecting it, to avoid damaging the hardware.

4.7.1 Disable the NVM

The `settings write` command with `--nvm-disable` disables the onboard NVM, so that it does not disturb the SPI communication of the external host.

```
$> uv run script/xe212tool.py settings write --nvm-disable
```

4.7.2 Enable the NVM

The `settings write` command with `--nvm-enable` enables the onboard NVM again, so that the A212 can boot from it.

```
$> uv run script/xe212tool.py settings write --nvm-enable
```

© 2026 by Acconeer AB - All rights reserved

5 Disclaimer

The information herein is believed to be correct as of the date issued. Acconeer AB (“Acconeer”) will not be responsible for damages of any nature resulting from the use or reliance upon the information contained herein. Acconeer makes no warranties, expressed or implied, of merchantability or fitness for a particular purpose or course of performance or usage of trade. Therefore, it is the user’s responsibility to thoroughly test the product in their particular application to determine its performance, efficacy and safety. Users should obtain the latest relevant information before placing orders

Unless Acconeer has explicitly designated an individual Acconeer product as meeting the requirement of a particular industry standard, Acconeer is not responsible for any failure to meet such industry standard requirements.

Unless explicitly stated herein this document Acconeer has not performed any regulatory conformity test. It is the user’s responsibility to assure that necessary regulatory conditions are met and approvals have been obtained when using the product. Regardless of whether the product has passed any conformity test, this document does not constitute any regulatory approval of the user’s product or application using Acconeer’s product.

Nothing contained herein is to be considered as permission or a recommendation to infringe any patent or any other intellectual property right. No license, express or implied, to any intellectual property right is granted by Acconeer herein.

Acconeer reserves the right to at any time correct, change, amend, enhance, modify, and improve this document and/or Acconeer products without notice.

This document supersedes and replaces all information supplied prior to the publication hereof.

© 2026 by Acconeer AB - All rights reserved