UM11153

NTAG SmartSensor getting started: A guide to start developing using an NHS31xx

Rev. 2.3 — 11 March 2023 User manual

Document information

Information	Content
Keywords	NHS31xx, Starter kit, SDK, quick start guide
Abstract	A concise guide describing the NHS31xx SW SDK setup process using the MCUXpresso development environment and its features when using the NXP provided NTAG SmartSensor development boards.

Revision history

Rev	Date	Description
2.3	20230311	Update for SDK 12.6 and MCUXpresso 11.7
2.2	20220124	Update for SDK 12.5 and MCUXpresso 10.2.1
2.1	20200106	Update for SDK 12.4
2.0	20190329	Update for SDK 12
1.7	20180615	Update for SDK 11.2
1.6	20180205	Update for SDK 11.1
1.5	20171011	Update for SDK 11 and LPCXpresso 8.2.2
1.4	20170220	Update for SDK 10
1.3	20170113	Update for SDK 9
1.2	20160905	Update for SDK 8.1
1.1	20160628	Update for SDK 8 and LPCXpresso 8.1.4
1.0	20160122	Update for SDK 6 and LPCXpresso 8.0.0
0.1	20150616	Initial version

1 Introduction

1.1 Document scope

The NHS31xx series is a family of ICs targeting vertical markets such as, but not limited to, smart logistics, industry 4.0, personal healthcare, and therapy compliance. The chips combine low power and flexibility with an Arm Cortex-M0+. Communication is typically done using the built-in NFC interface or wired using I^2C or SPI. All chips have been designed with a low system cost in mind and provide simple interfaces to sensors such as the built-in accurate temperature sensor.

Each NHS product has a specific value proposition and comes with its own use cases and reference design. This document describes how to evaluate our offering: How to set up, compile, and flash the firmware contained in the SDK using the released boards. In this way, a firmware engineer can start developing using an NHS31xx.

Figure 1. Backside of an NTAG SmartSensor board.

Note: *This document is applicable to any setup featuring an NHS31xx, regardless of the demo PCB the IC is mounted on.*

1.2 Supported environments

- The MCUXpresso IDE, specifically versions 11.7 and up, is the only supported environment for firmware development. Older versions are not supported.
- The NHS31xx SDKs are developed and tested under Windows 10 and macOS Monterey.
  The NHS31xx SDKs are known to be working on Linux distributions (such as Arch and Debian/Ubuntu). However, no support is given.

1.3 Contact

- Business: nhs-info@nxp.com
- Technical: nhs-support@nxp.com

2 Tooling

2.1 MCUXpresso IDE

Download the MCUXpresso IDE v11.7 installer for your platform via https://www.nxp.com/pages/:MCUXpresso-IDE#downloads.

Note: *Install the MCUXpresso IDE to the default location. Do not use a folder which requires administrative or elevated rights to write to, such as "Program Files" or "Program Files (x86)" (Windows).*

2.2 SDK

The SDK can be used by importing all the provided projects into a workspace. The MCUXpresso IDE-specific Quickstart panel can be used to import everything easily. For a short description of Quickstart panel, see Section 4.1.1.

Create a workspace or reuse your existing workspace and import the projects contained in the SDK release.

1. In the Quickstart Panel view, click Import projects from file system.
2. Fill in the path (including the filename) to the compressed SDK contents next to Archive or fill in the path to the decompressed SDK contents next to Root directory and click Next.
3. Select the projects found. Best is to import all projects found, including the non-application projects. Each application project depends on the chip library, on one board library, and on multiple modules grouped in the mods container project. To be able to build successfully, all sources must be available in the same workspace.

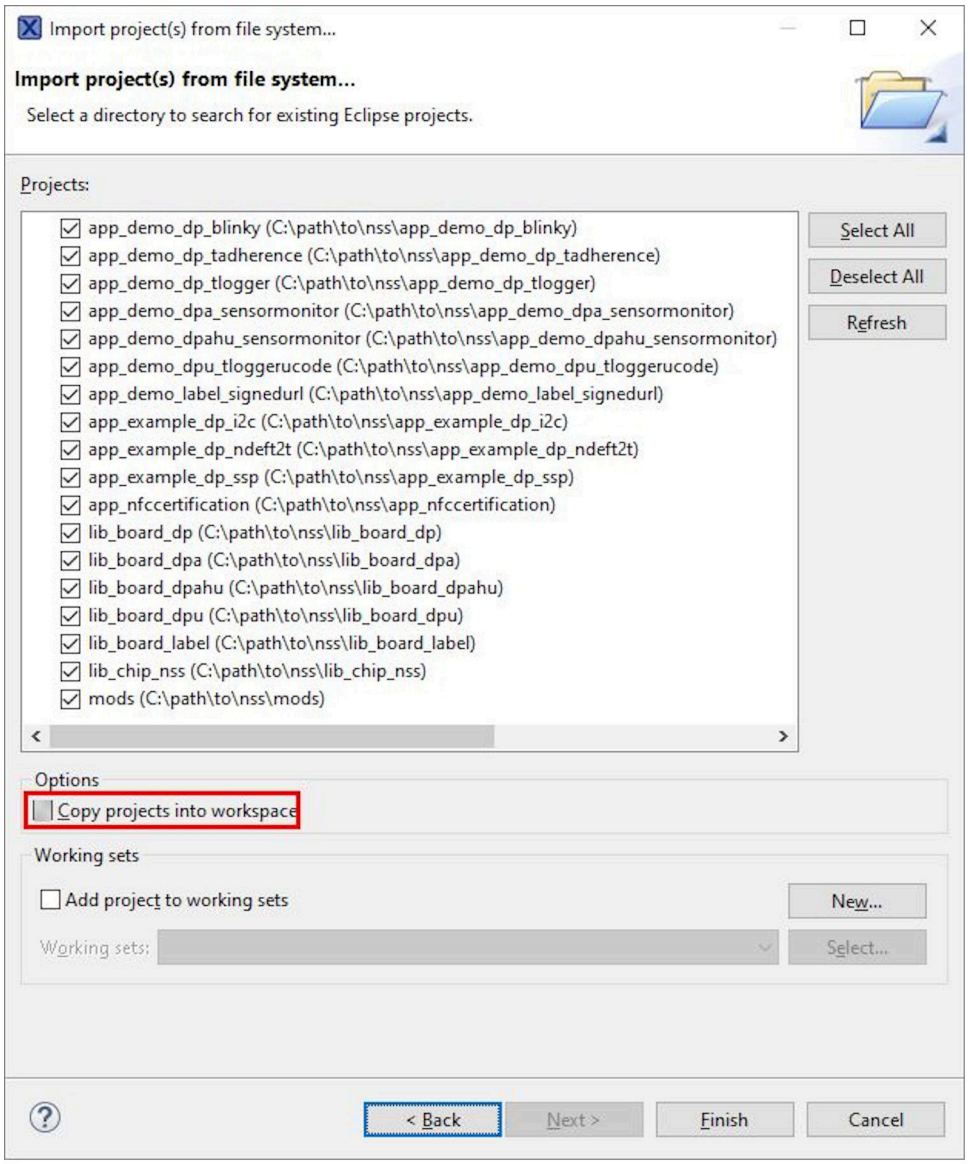


Figure 2. Importing existing projects

Note:

Pay special attention to the option Copy projects into workspace. When importing from a Project archive (zip), it is grayed out as only a copy can be performed. However, when importing from a Project directory (unpacked), you can choose.

- *If checked, the files are copied into your MCUXpresso workspace and the original files are left untouched.*
- *If unchecked, the workspace only contains a reference to the location you provided and all changes are done in-place.*

The option is selected by default, but both options yield equal results. Choose whatever best suits your style. Be careful not to mix styles. Application projects using modules from the mods project use relative references. If the application is imported by copying while the mods project is imported by referencing or vice versa, building fails with errors, like `chip.h:35:29: fatal error: startup/startup.h: No such file or directory.`

Note:

The current workspace can be retrieved by hovering over the hyperlinked text in the bottom status bar or by copying the default value from the dialog that pops up after File > Switch Workspace > Other... > Workspace.

After importing the projects, all firmware content from the SDK becomes available from within the IDE.

Note: *A description and guide regarding host tools (mobile and PC-based) and IDEs other than the MCUXpresso IDE is outside the scope of this document.*

2.3 Flash Magic

Flash Magic is a third-party PC tool for programming FLASH-based microcontrollers from NXP Semiconductors using Intel HEX files via a serial protocol. It can be freely used during development or for programming small batches. Using Flash Magic on a production line is also possible, but requires a purchase.

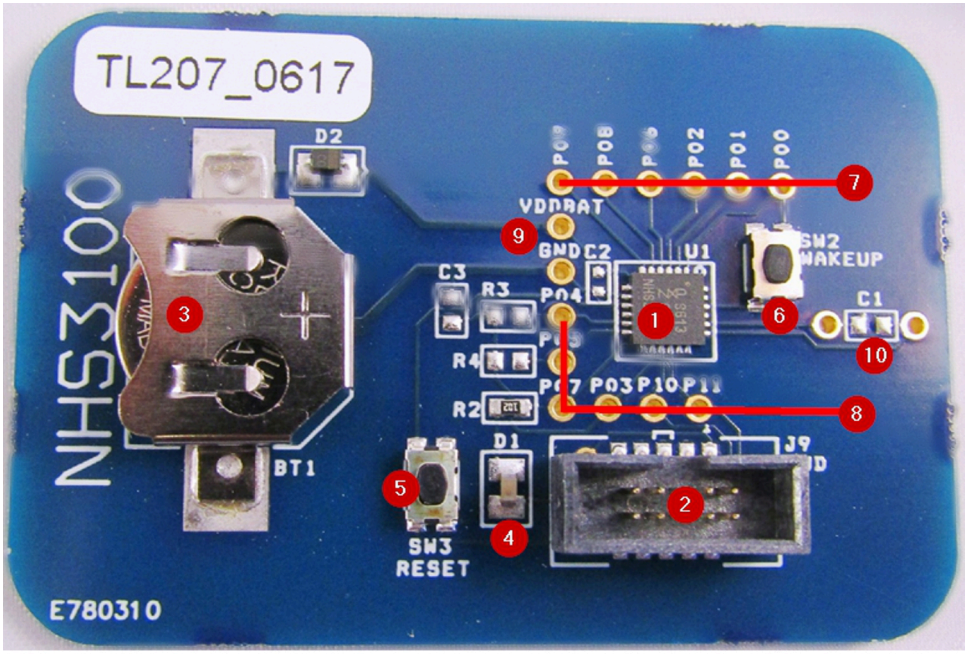
The use of this tool is not enforced, but it helps to program ICs quickly using prebuilt firmware images.

Further information can be found in the application note "Overview of supported methods for firmware flashing on NHS31xx ICs" (AN12328), which is part of the SDK.

3 Boards

3.1 Demo PCB

For each IC, several boards have been developed and released. These boards can be used for demonstration and SW development purposes. Most boards feature the same characteristics. As an example, consider the NHS3100 temperature logger demo PCB (Figure 3).



aaa-031858

Figure 3. NHS3100 temperature logger demo PCB

Components:

- (1) An NHS3100 IC in an HVQFN24 package (U1)
- (2) An SWD connector (J9)
- (3) A coincell holder for standalone operations (BT1)
- (4) One SW controllable LED (D1)
- (5) A tactile switch (SW3) connected to the RESETN pin
- (6) A tactile switch (SW2) connected to the WAKEUP pin
- Through-holes for easy access to:
  - (7) and (8) All PIOs of the IC (P0x)
  - (9) GND and VDDBAT
  - (10) Antenna coil connections LA and LB, connected with the NFC antenna on the back (not visible)

The demo PCB is ready for use upon arrival.

Note: *After powering the demo PCB by inserting a battery or by connecting it to a prepared LPC-Link2 board (see Section 3.2.1), the NHS IC does not become active. Only after briefly providing an NFC field near the antenna or when the RESET button is pushed, the IC wakes up and starts executing the Arm program stored in Flash.*

Note: *The suggested coincell battery type to use is CR1225. It can be ordered internationally at, for example, https://www.newark.com/w/c/batteries-chargers/prl/results?st=CR1225.*

Note: *The Demo PCB can also be powered via the JTAG connector or a nearby NFC field.*

Note: *The diode D2 protects the circuits against reverse battery polarity and charging the battery via the SWD reference. However, there is a chance that the battery itself is shorted, depleting it. Ensure that the battery is inserted with the positive side facing up.*

3.2 Debug probe

3.2.1 LPC-Link2

LPC-Link2 is a legacy standalone debug adapter that can be configured to support various development tools and IDEs with downloadable firmware. It is compatible with the MCUXpresso IDE and the NHS31xx SDK, using the CMSIS-DAP debugging protocol. This board has been traditionally included in the full development kits that are on offer via nxp.com/ntagsmartsensor.

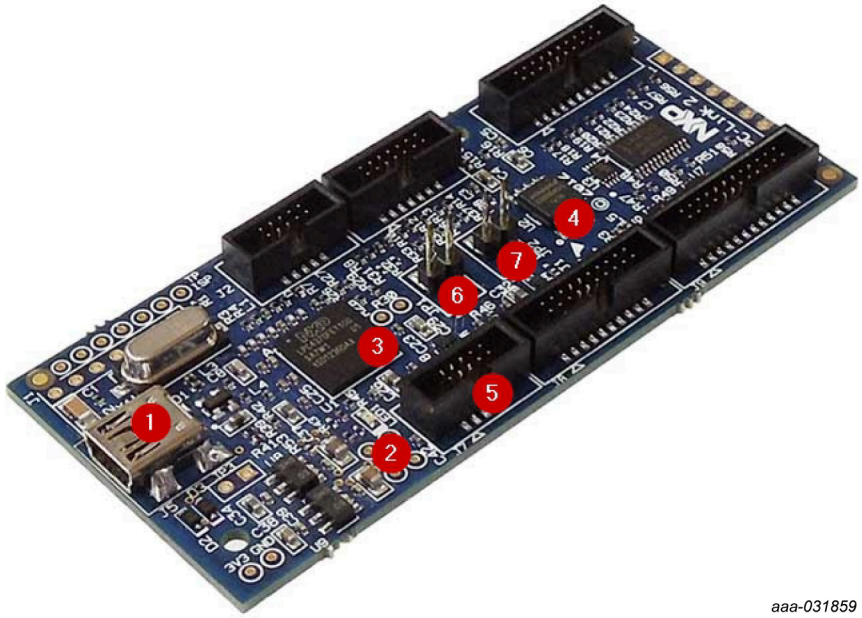


Figure 4. LPC-Link2 board without jumpers over JP1 and JP2

A few components of interest:

- (1) A miniUSB connector
- (2) One LED signifying connection and communication with the PC
- (3) An LPC4370 to execute the debugging protocol firmware
- (4) An SPIFI flash which may be used to store the debugging protocol firmware
- (5) A JTAG connector compatible with the NHS demo PCB
- (6) JP1 to control the use of the SPIFI flash
  When mounted, it selects the SWD firmware in the onboard SPIFI flash. When missing, the MCUXpresso IDE soft-loads the SWD software via DFU.
- (7) JP2 to control if the Device Under Test is powered via the JTAG connector
  When mounted, the LPC-Link2 provides 3.3 V supply voltage on the VTREF pin of the SWD and powers the NHS31xx.

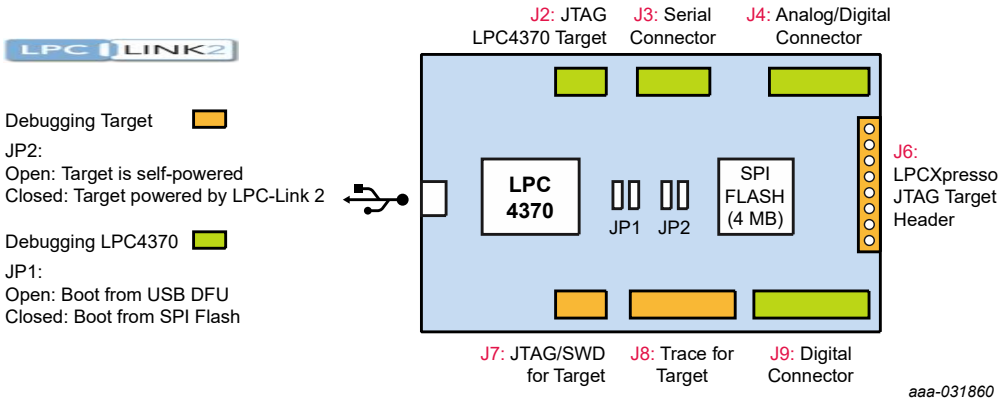


Figure 5. Image taken from https://community.nxp.com/message/630655

More information

http://www.nxp.com/demoboard/OM13054.html

The LPC-Link2 board is ready for use upon arrival.

Warning

On some Windows PCs, an old version of the LPC-Link2 driver (1.0.0.0) is automatically selected, even though the installation procedure was followed correctly. The selection of an old version of the LPC-Link2 driver goes unnoticed until a debug session is attempted. No LPC-Link2 board can be put to use. To fix this issue, uninstall and remove the old driver via the Windows Device Manager. After installing the new device drivers, driver version 2.0.0.0 is reported.

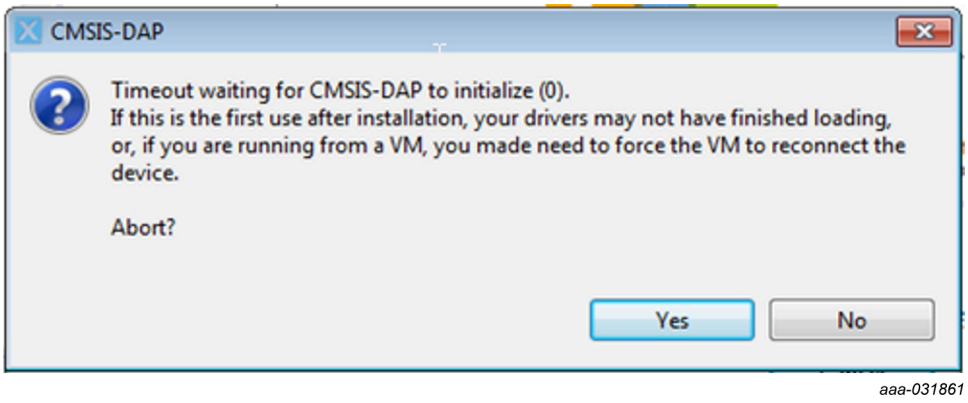


Figure 6. Connection problems because of an outdated Windows driver version

More information

https://community.nxp.com/message/630660

3.2.1.1 JP1

By default, no jumper is placed over JP1.

- JP1 unmounted
  The PC host downloads the CMSIS-DAP debugging protocol firmware[1] to the RAM of the LPC-Link2 board each time it is power cycled.
- JP1 mounted
  Optionally, if the protocol firmware is stored in the SPIFI flash on the LPC-Link2 board, a jumper can be placed over JP1. With it, the LPC-Link2 board is immediately ready for use after powering up. The result is a modest speed increase, which may become important when there is only a very small time window in which the SWD lines are active.

To program the SPIFI flash on the LPC-Link2 board with the CMSIS-DAP debugging protocol firmware, download and install LPCScrypt. Ensure that the jumper is not placed over JP1 and launch the Program LPC-Link2 with the CMSIS-DAP script (under Windows directly available from the Start Menu). Afterward, fit JP1 and power cycle the LPC-Link2 board.

Note: *The use of LPCScrypt is not required. Using it, a faster start-up cycle is gained while establishing a debug connection. When continually switching between debugging and using Flash Magic, remember to remove and refit the jumper over JP1.*

Non-bridged variant

By default, the debugging protocol includes extra bridge channels to provide extra functionality such as SWO Trace capture and UART VCOM port. This functionality is either not available through the LPC-Link2 board or not exposed by the NHS31xx IC.

To use the non-bridged variant with JP1 unmounted, select CMSIS-DAP (Non-bridged - Debug only) as LPC-Link 2-boot type under Window > Preferences > MCUXpresso IDE > LinkServer Options. This option is applied on all the projects in the same workspace.

To use the non-bridged variant with JP2 mounted, use the CMSIS-DAP script from LPCScrypt with the `NB` argument: `<LPCScrypt install path>\scripts\program_CMSIS NB`

Download

A direct download link can be found in the tools folder of the SDK contents.

1 CMSIS-DAP provides a standardized way to access the debug port of an Arm Cortex microcontroller via USB. In combination with a NHS31xx IC, it provides a connection from a development board to a debugger running on a host computer using serial wire debug (SWD) to the target device.

3.2.1.2 JP2

By default, no jumper is placed over JP2.

- JP2 unmounted
  The debugging target, the NHS31xx demo PCB, must be self-powered.
- JP2 mounted
  The NHS31xx demo PCB is powered by the LPC-Link2 board.

Note: *The battery and the jumper over JP2 can be fitted simultaneously.*

Warning:

Regardless whether JP2 is mounted, when testing the power-off state of the NHS31xx (VDDBAT switched open), the SWD connector must be removed as the SWD pins are driven high by the LPC-Link2 probe, interfering with the PMU of the NHS31xx.

3.2.1.3 Debug setup

Through the LPC-Link2 board, you have full SW debug capabilities on the NHS31xx demo PCB.

- Connect a flat cable with the JTAG connector on the LPC-Link2 board: J7 (1) and the Demo PCB: J9 (2).
- Connect a mini USB cable to the LPC-Link2 board and your PC (3).

Your setup should now look similar to Figure 7.

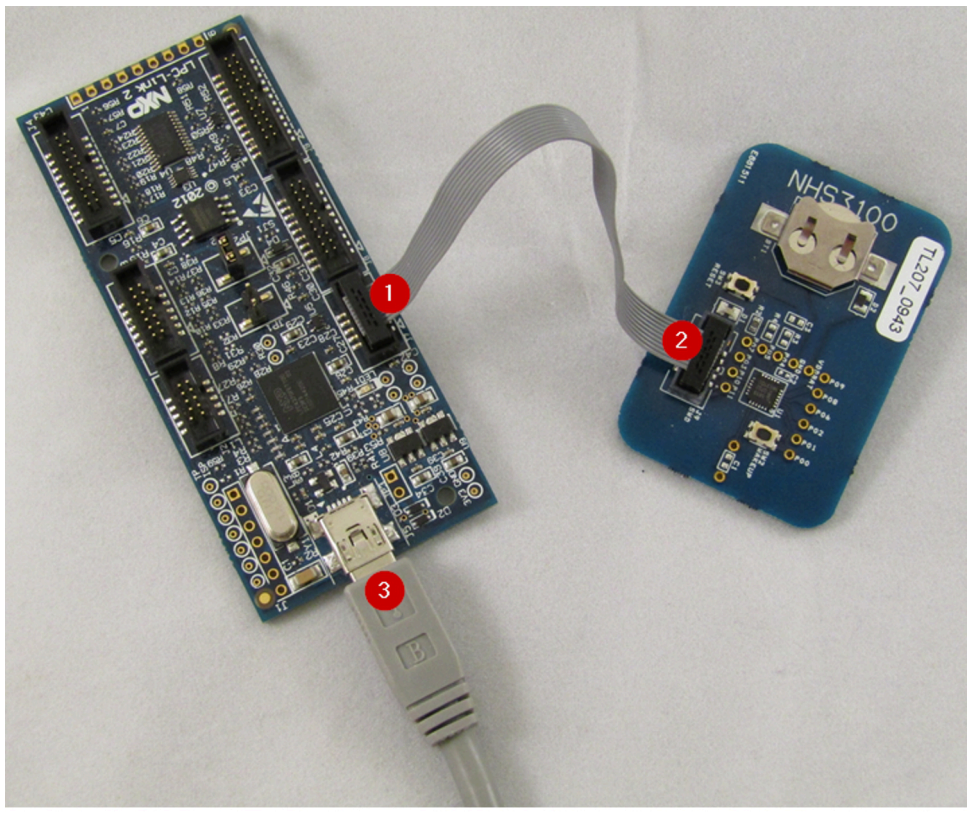


Figure 7. Demo PCB ready for debug using an LPC-Link2

3.2.2 MCU-Link Pro

MCU-Link Pro is the new standard debugger from NXP. It is a fully featured debug probe that can be used with MCUXpresso IDE and any other IDE that supports CMSIS-DAP or J-Link protocols. It is compatible with the MCUXpresso IDE v11.7 and the NHS31xx SDK, using the CMSIS-DAP debugging protocol.

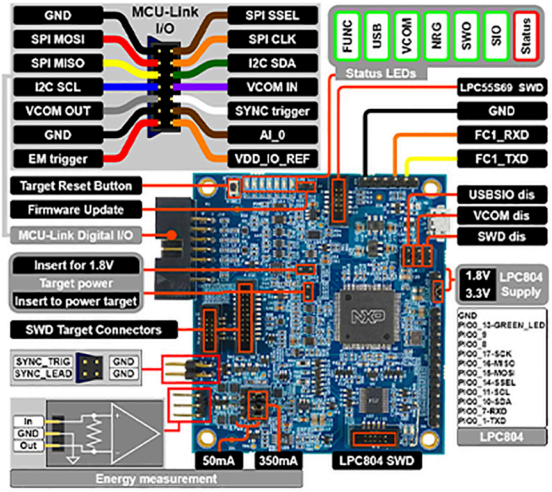


Figure 8. MCU-Link2 Pro board

More information

https://www.nxp.com/pages/:MCU-LINK-PRO

The MCU-Link Pro board is ready for use upon arrival.

Notes

- The tracking of reference voltage circuit of MCU-LINK connects VDDBAT of the NHS31xx to 3.3 V via a 30.1 kΩ resistor. The current injected (about 50 µA at VDDBAT = 1.8 V) can prevent that the NHS31xx power-cycles when the onboard battery is disconnected.
  In this case, power-cycle the NHS31xx by disconnecting the SWD cable.
- Fit jumper J6 on the MCU-LINK Pro board to supply 3.3 V to the NHS31xx board. It eliminates the requirement for a battery.

3.2.3 J-Link BASE

J-Link BASE is a USB-powered JTAG debug probe, fully supporting all NHS31xx boards. It can be used in all major IDEs including Eclipse and the MCUXpresso IDE.

Figure 9. J-Link Base

To connect the J-Link BASE with a NHS31xx board, a 9-Pin Cortex-M adapter is required.

Notes

- To power the NHS31xx board, a SEGGER target-supply-adapter can be used, eliminating the requirement for batteries. To route the 3.3 V from the target-supply-adapter LDO to VDDBAT of the NHS31xx, a wire must be mounted on the target-supply-adapter 20p target connector, connecting pin 1 (VTREF) to pin 19 (supply)

More information

https://www.segger.com/products/debug-probes/j-link/models/j-link-base

https://www.segger.com/products/debug-probes/j-link/accessories/adapters/9-pin-cortex-m-adapter/

https://www.segger.com/products/debug-probes/j-link/accessories/adapters/target-supply-adapter/

4 Using the IDE

Documentation about using the MCUXpresso environment can be found under Help > Help Contents. If you are new the MCUXpresso IDE or unfamiliar with an Eclipse-derived IDE, read this documentation.

4.1 Project Setup

4.1.1 Quickstart panel

The MCUXpresso IDE is built on Eclipse and adds a Quickstart Panel view. In that view, the panel Start provides quick links to existing functionality scattered over the various Eclipse menus.

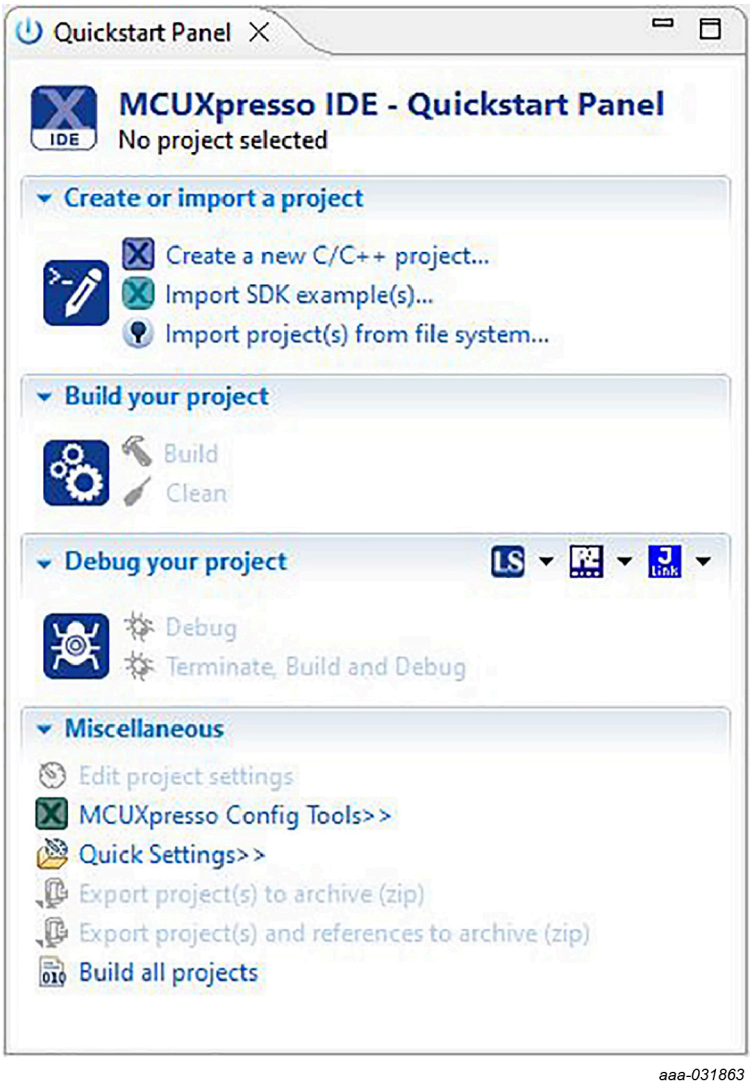


Figure 10. Quickstart panel groups often used project-related commands

If you feel more comfortable with the Eclipse way of offering, you can close this view, saving up some screen space for other views. The view can be found again via Windows > Show View > Other… > Quickstart > Quickstart Panel.

4.1.2 New project wizard

The new project wizard is the preferred way to create projects quickly.

1. In the Quickstart Panel view, click Create a new C/C++ project…
2. Filter the available boards by selecting NHS31xx as Target under Preinstalled MCUs.
3. Select the board that features the type of the IC in use.

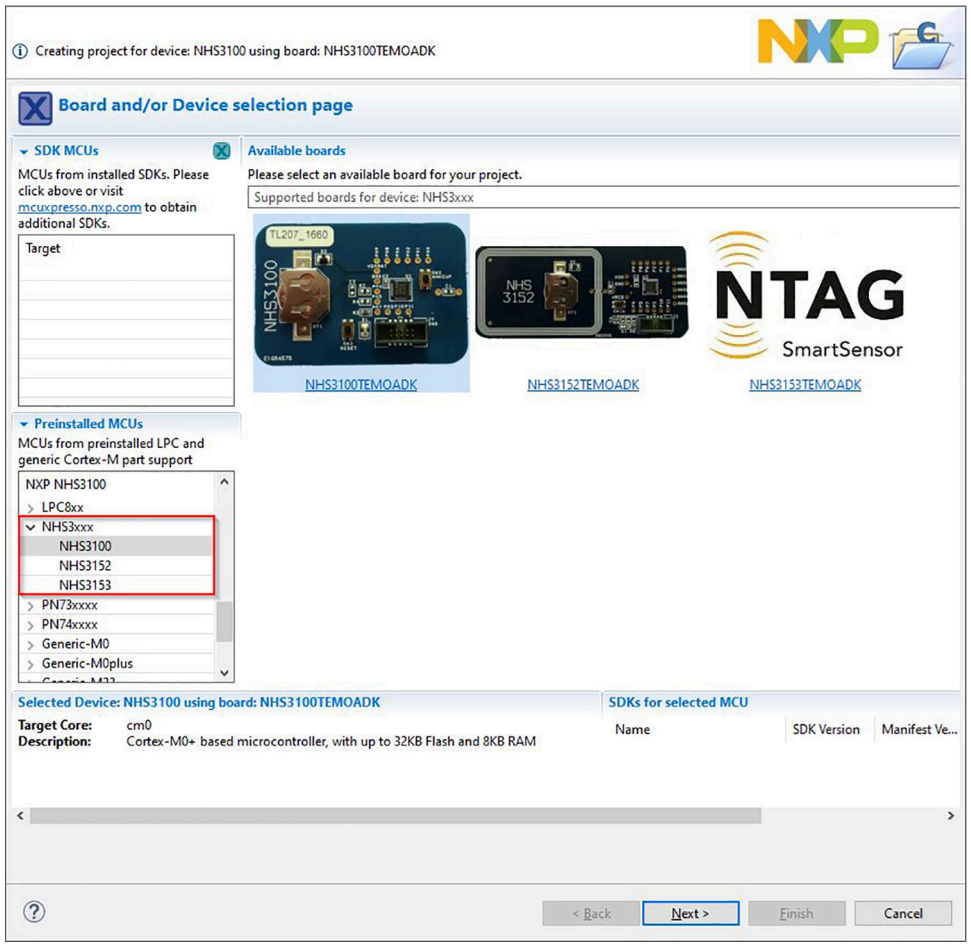


Figure 11. SDK Wizard

1. Select LPCOpen – C Project.
2. Choose a project name and select the desired location.
3. You can accept or change the default options presented in the screens that follow as you see fit, except for the chip and board library. Choose lib_chip_nss as the referenced LPCOpen Chip Library Project. Select or type in the correct board - such as lib_board_dp - as the referenced LPCOpen Chip Board Project.

Note:

The MCUXpresso IDE v11.7 uses the GNU Arm Embedded Toolchain 10.3-2021.10, which means that the default compiler language dialect for C code is gnu18 (GNU dialect of ISO C17).

4.1.3 Import project

Any existing project can be imported. The procedure is the same as described in Section 2.2.

4.1.4 Mods

Code that does not belong in the Chip library or the Board library, but that can still be used for a multitude of projects, is grouped under the mods project. The mods project is a simple container project with no compilation settings enabled. The mods project contains SW modules that can be reused across several other projects while supporting diversity (using `#define` precompilation flags). These modules may be included at any layer (chip, board, application) and the project they are included in compiles the respective code.

An application that requires the functionality one of the reusable modules provides, must create a link to it (see Figure 12).

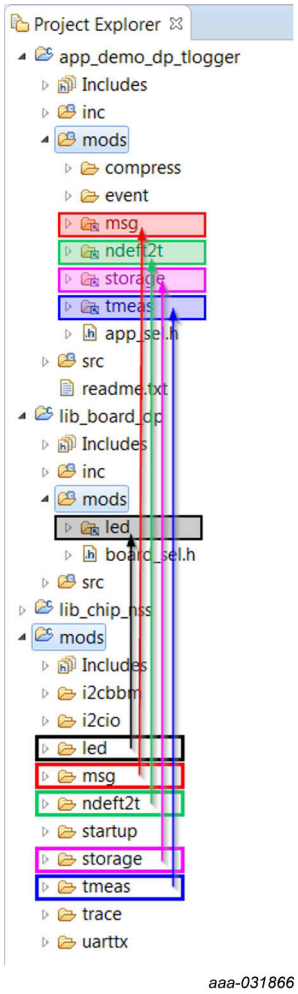


Figure 12. Code reuse by linking modules in other projects

In Project Explore view of the IDE:

1. Control-drag the desired reusable module, a child of the mods project, to the mods folder of the application project. Before releasing the mouse button, be sure that the `CTRL` key is down.

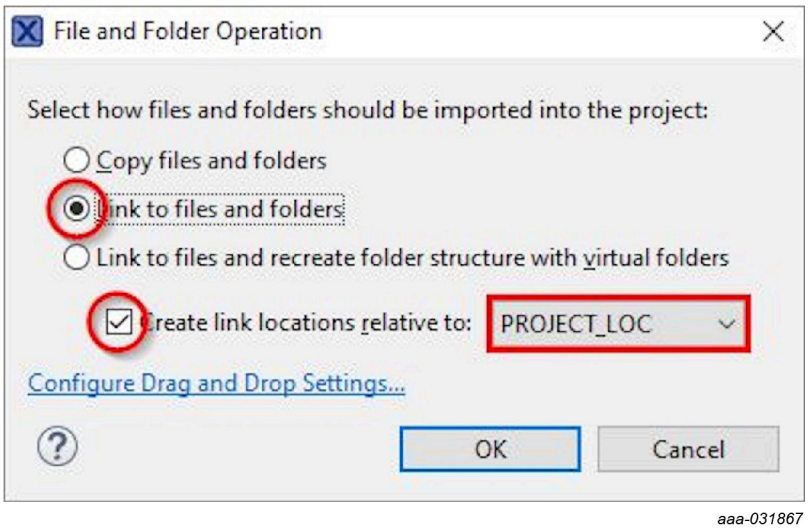


Figure 13. Creating a link to a reusable module

2. A window pops up. Ensure that link to files and folders and Create link locations relative to PROJECT_LOC are selected (see Figure 13).

The linked module is given a link icon overlay. Its files are directly accessible within the application project. The module now compiles together with the rest of the application. A simple include statement `#include "modx/modx.h"` now gives you access to the API of the module.

After including one or more modules, check the documentation of each module to see which diversity settings to use. Each module provides reasonable defaults which you can override by adding `#define` precompilation flags in `app_sel.h`. By tweaking the module, you can include or exclude functionality and reduce the required code size.

Note: *Alternatively, you can create a link via File > New > Other… > General > Folder. After clicking Next, select the mods folder of your project, click Advanced, and select a link to an alternate location (Linked Folder). In the field below, you can build up the path to the module you want to reuse. To build the path in a reusable way, use the Browse… and Variables… buttons.*

4.2 Building

Building can be done in various ways:

- Using the Project menu
- Using the Quickstart Panel view
- Using the shortcut key combination `<CTRL>+B` (which builds all projects using all build configurations)
- Using the context menu of the project

No additional steps are necessary. After a successful import, all imported projects build without issues. First select a project, after which the Build command becomes active in the Quickstart Panel view, noting the selected project and the active build configuration between brackets.

4.3 Flashing

By default, starting a debug session (see Section 4.4) also automatically programs the flash. However, you can also flash any `.axf` file, `.elf`, or `.bin` file without starting a debug session:

1. Make a hardware connection as outlined in Section 3.2.1.3.

2. Within the IDE, select a compatible project. Some of the project settings are implicitly reused. Ensure that a project is selected that reuses the same MCU settings as the `.axf` file you want to flash.
3. Click the GUI Flash Tool icon in the toolbar.

aaa-031868

Figure 14. GUI Flash Tool icon in the toolbar

4. Select an available attached probe: an LPC-Link2 CMSIS-DAP, LPC-Link2 flashed with J-Link firmware, or J-Link probe.
5. In the dialog that pops up, verify these settings (see Figure 15).
   - (1) Flash driver: NSS_30k_8k_IAP.cfx
   - (2) Select file: The `.axf` or `.bin` application file to flash. The Intel hex format - with a `.hex` extension - is not recognized.
   - (3) Base address: 0

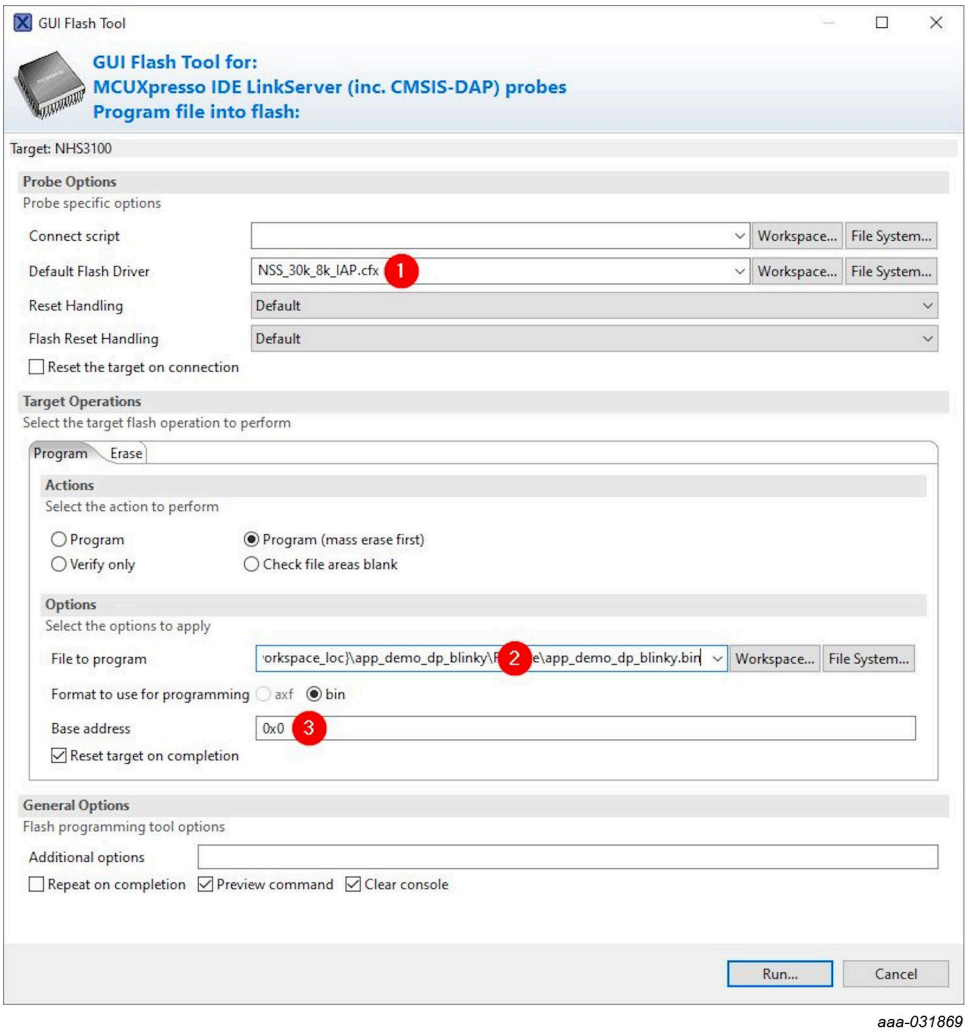


Figure 15. Program Flash dialog

After clicking Run..., the flash is programmed. At the end, a dialog pops up with the final result.

4.4 Debugging

You can use the full debugging features the IDE offers. By default, starting a debug session also automatically programs the flash.

1. Make a proper HW connection as outlined in Section 3.
2. Within the IDE, you can start a debug session in various ways:
   - Via the Run menu
   - Via the Quickstart Panel view
   - Using a shortcut key combination
   - Using the context menu of the project

Note: *The IDE only allows one debugging session at a time. Even if the connection was broken, the session within the IDE must still be explicitly terminated before a new session can be successfully created.*

Warning:

When the firmware running in the NHS31xx IC decides to enter the deep power-down mode or the power-off mode, the SWD lines become inaccessible. Continuing or starting a debug session is then impossible. One possible way is to toggle the RESETN pin. When the IC resets, start a debug session in the IDE, before the code that changes the power mode has a chance to execute. Since this action is time-critical, this approach may be too impractical. Another option is to provide an NFC field near the NFC antenna. Depending on the code that has been flashed, you may then have sufficient time to launch a debugging session. A last way is to use the Flash Magic tool. Since it is much faster than the MCUXpresso IDE and triggers a RESET pulse prior to attempting to set up a debug session, it increases the chance to be able to halt the Arm core before the SWD lines become inaccessible.

To ensure that a software bug does not break the device by permanently disabling the SWD lines, add debug code that implements a means to allow SWD access before entering the deep power-down mode or disconnecting the battery. Also ensure that the corresponding PIOs have been configured correctly for SWD functionality beforehand. Examples of this approach have been implemented in the demo firmware app_demo_dp_tlogger:

- When using the define `APP_MAINTAIN_SWD_CONNECTION`, the deep power-down and power-off modes are entirely avoided while retaining the entire functionality.
- When the board has a provision for it (recognized using `BOARD_ENABLE_WAKEUP`), the application checks if the wake-up pin is pulled low at start-up (in `ResetISR`). It waits for as long as that pin is pulled low, giving ample time to connect via SWD and start a debugging session.
- When using the Debug build configuration, write an NDEF message on page 6 onwards, containing a MIME record with `08h 00h` as payload. It corresponds to a msg command with ID `MSG_ID_PREPAREDEBUG` and configures the SWD lines correctly, then pauses execution. The IDE then has sufficient time to set up a debug connection and either reflash the board or start debugging the running instance. See the msg module documentation, available in the SDK under *<SDK>/docs/firmware.html*.
- If three conditions are met, the application waits for 3 seconds during deinitialization (`DeInit`):
  - NFC, RTC, or the WAKE-UP pin did not start the IC.
  - No NDEF message exchange took place.
  - No measurement is due, that is, the next mode is the power-off mode.

  In practice, it means that to give you a reasonable time to establish an SWD session, the IC must be hard-reset.

Note: *To learn more about the low-power modes, check the PMU documentation in the SDK (check for PMU_NSS CHIP: NSS Power Management Unit driver). There, you can also find information about maintaining state information that persists even when restarting after leaving the deep power-down mode.*

Note: *To learn more about waking up from the deep power-down mode in a timely manner, check Example 4 in the RTC documentation in the SDK, under RTC_NSS CHIP: NSS Real-Time Clock driver.*

Note: *To learn more about the problems and possible workarounds that deep power-down mode and power-off mode can give during debugging, check SW Debug Considerations in the documentation in the SDK.*

4.5 Example

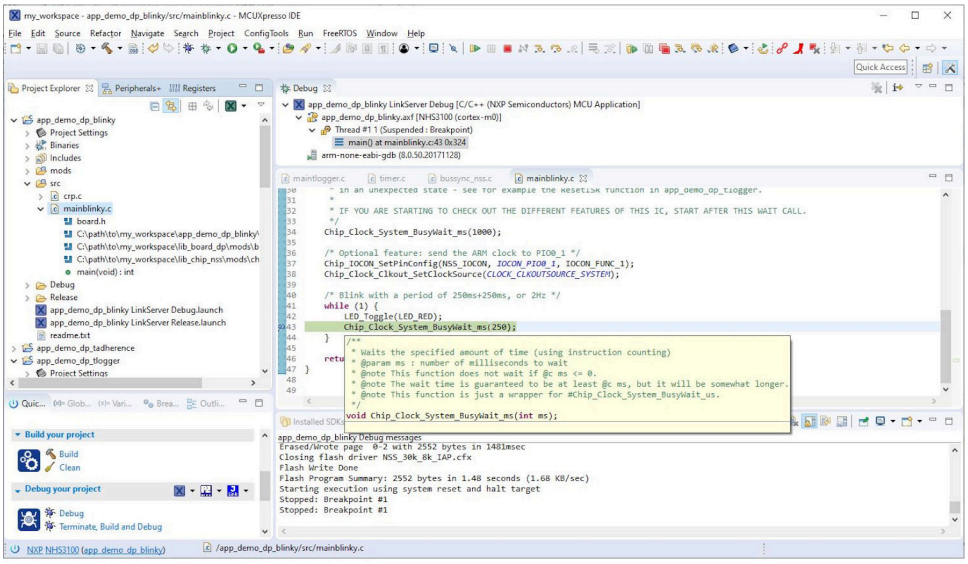
The blinky application is a very basic Hello World application. As an example, we debug the code and perform some basic debugging steps.

The full application code is:

```
#include "board.h"

int main(void)
{
    Board_Init();

    while (1) {
        LED_Toggle(LED_ALL);
        Chip_Clock_System_BusyWait_ms(250); /* Blink with a period of 2Hz */
    }

    return 0;
}
```

To debug the code, use the Debug ‘app_demo_dp_blinky’ [Debug] command in the Quickstart panel. It creates a default launch configuration (if necessary). Using that configuration flashes the correct image and starts a debugging session with the Arm core halted on the first instruction in main. You can step into and over each function, inspect and change memory contents, set and change conditional and data breakpoints, and change the program counter. Every debugging feature Arm supports is available.

After resuming the application you can pause it again by, for example, adding a breakpoint inside the `while(1)` loop. Double-click the desired line in the left gutter of the file editor view, where it halts the execution immediately (see Figure 16).



aaa-031872

Figure 16. Debug session with Arm halted due to a breakpoint

You can stop a debugging session, which leaves the application running. Afterward you can launch a new debug session by creating a launch configuration where you instruct the GDB debugger in the IDE to connect only. For launching a new configuration, go to Run > Debug Configurations… Copy the existing debug launch configuration and, under the Debugger tab, change the Configuration Option Attach only to True (see Figure 17). The connect-only launch configuration prevents that a new firmware image is downloaded in the Arm and lets you investigate a running instance.

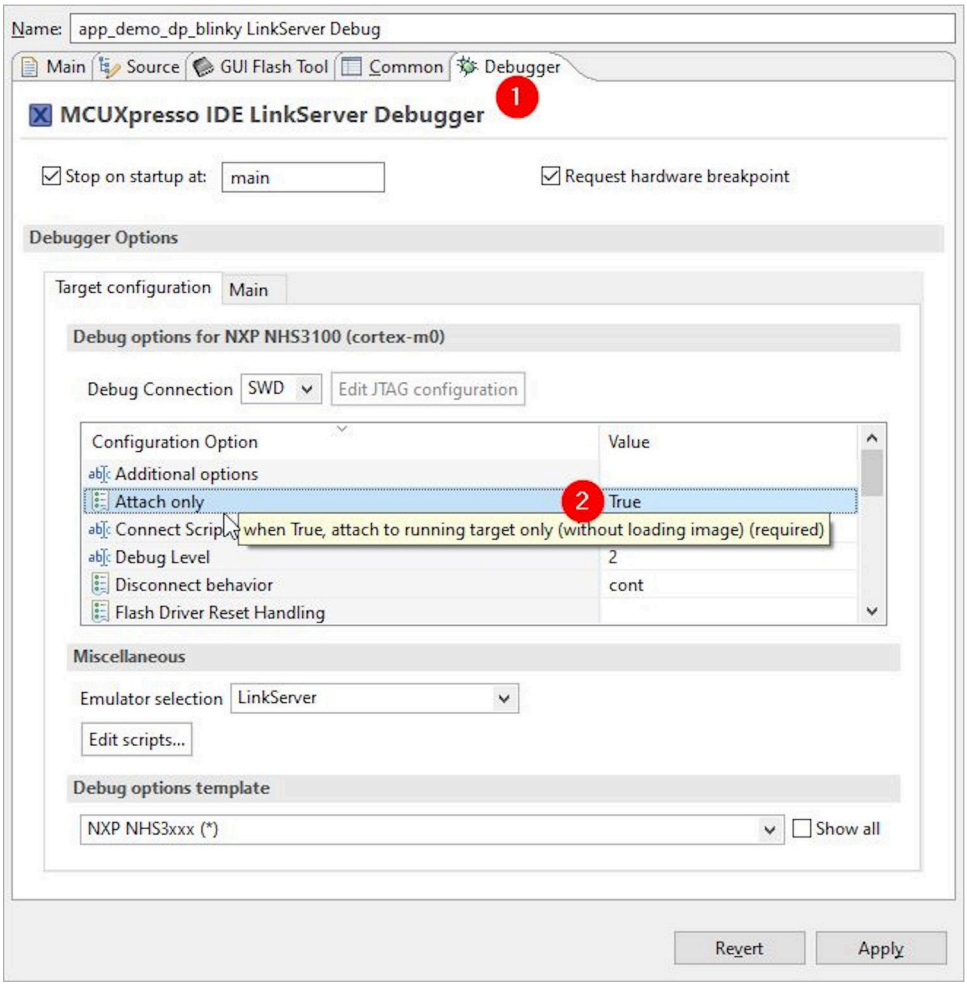


Figure 17. Creating a connect-only launch configuration

5 Conclusion

In this document, a basic overview has been given to unlock the full potential of the HW with an emphasis on the tooling and the SW.

More detailed information can be found in:

- The data sheet
- The schematics of each respective board
- The SW documentation – near the code itself, or as generated HTML pages

All these documents are part of the software development kit.

Ensure that the user manuals for the demo applications created for each specific IC are checked. They give a higher-level idea of the potential that can be reached with our ICs.

6 Legal information

6.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

6.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

NTAG SmartSensor getting started:
A guide to start developing using an NHS31xx

Contents

For more information, please visit: http://www.nxp.com

Date of release: 11 March 2023
Document identifier: UM11153