NHS3xxx Getting Started

A guide to start developing using a NHS3xxx

Rev. 1.2 — September 5, 2016 **User manual**

Document information

Info	Content
Keywords	NHS3xxx, Starter kit, SDK, quick start guide
Abstract	A concise guide describing the NHS3xxx SW SDK setup process using the LPCXpresso development environment, the intended development HW and its features.

Revision history

Rev	Date	Description
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

Contact information

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

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

1.1 Document Scope

The NHS3xxx series is a family of ICs targeting the smart pharma/smart health market. They combine low power and flexibility with an ARM Cortex M0+. Communication is done using the NFC interface or wired using I^2C or SPI. All chips have been designed with the lowest system cost in mind and provide simple interfaces to sensors such as temperature.

This document describes how to start with the firmware contained in the SDK to evaluate and start development using an NHS3xxx. Each NHS product has a specific value proposition and comes with its own use cases and reference design.

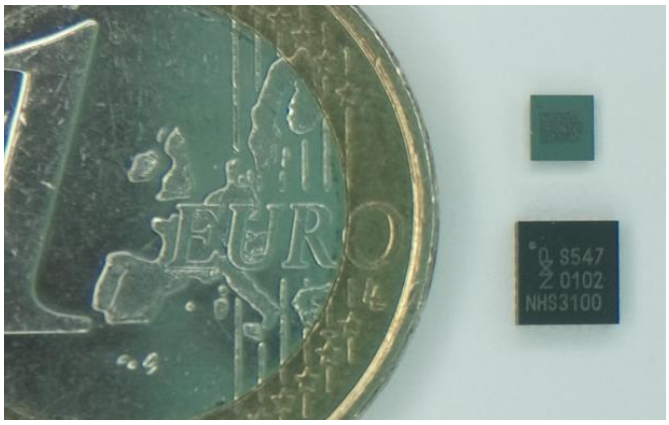

Figure 1: €1 and two NHS3xxx in a CSP and HVQfn24 package

1.2 Supported environments

- LPCXpresso IDE is the only supported environment for firmware development.
- The NHS3xxx SDK has been developed and tested under Windows 7.

1.3 Contact

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

2. Tooling

2.1 LPCXpresso IDE

Before installation of the LPCXpresso IDE, ensure the development PC meets the requirements as listed in

http://www.nxp.com/lpcxpresso/ide-v8-user-guide,
§2.1 Host Computer Hardware Requirements

2.1.1 Installation

- Download the LPCXpresso IDE 8.1.4. A direct download link can be found in the *tools* folder of the SDK contents.
- Install.

More information:

http://www.nxp.com/lpcxpresso/ide-v8-user-guide,
§2.2 Installation

Warning:

Higher versions have not been checked; lower versions will not work with the NHS3xxx plugin (see section 2.2 NHS3xxx plugin, below).

Warning:

Be careful when migrating from one version of the LPCXpresso IDE to another one. A new LPCXpresso IDE version can open workspaces created by earlier releases, but the workspace and the projects it contains may thereafter no longer correctly load into an earlier version. Also ensure all the launch configurations are re-created. More information can be found at
https://www.lpcware.com/content/faq/lpcxpresso/migrating-new-version-LPCXpressoIDE

2.1.2 Registration

Initially the LPCXpresso IDE runs with a default *Unregistered* license. The current license is listed on the *Welcome* page that is by default opened after each restart, and can also be checked via *Help* > *Display License Type*.

Activate with a *Free Edition* or *Pro Edition* license. The *Free Edition* is sufficient for all NHS3xxx development: all the features of the LPCXpresso IDE are available and the debug download limit of 256kB will not be hit, as any NHS3xxx IC has 32kB Flash.

More information:

http://www.nxp.com/lpcxpresso/ide-v8-user-guide,
§2.3 Licensing overview
§2.5 Activating your product (LPCXpresso Free Edition)
§2.6 Activating your product (LPCXpresso Pro Edition)

Note:

The automatic display of the *Welcome* page at startup can be suppressed by clearing the checkbox at *Window* > *Preferences* > *LPCXpresso* > *General* > *Show welcome view*. This option is workspace dependent.

2.1.3 SDK

As a prerequisite, the NHS3xxx plugin must be installed before the SKD can be used. See 2.2 below to install the plugin, and 3 below for an overview of the benefits of the plugin.

With the plugin in place, the SDK can be used by importing all the provided projects into a workspace. The LPCXpresso specific *Quickstart panel* can be used to easily do this. See 2.1.4 below for a short description, and 4.1.1 below for the required steps.

After importing the projects, all firmware content from the SDK becomes available from within the LPCXpresso IDE. For the provided Android APP sources, we recommend using Android Studio.

Note:

A description and guide regarding Android APP development and the Android Studio environment is out of scope of this document.

2.1.4 Quickstart panel

The LPCXpresso IDE is built upon Eclipse and adds a *Quickstart Panel* view. In that view, the panel *Start here* provides quick links to existing functionality that is scattered throughout the various Eclipse menus.

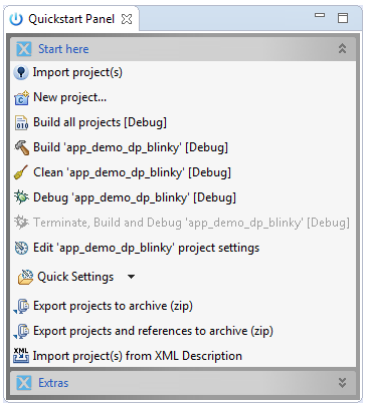

Figure 2: The Quickstart panel groups often used commands for the selected project

Should 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 always be found again via *Windows* > *Show View* > *Other…* > *Quickstart* > *Quickstart Panel*

2.2 NHS3xxx plugin

The LPCXpresso IDE provides support for all commercially available Cortex-M/ARM7/ARM9 LPC MCUs. However, to provide support for the NHS3xxx chips, a dedicated plugin must be installed on top of an existing installation of the LPCXpresso IDE.

With the plugin installed, the code can be compiled for the correct CPU core and corresponding memories; and while debugging, the peripheral registers and the different fields are available for inspection and modification. In addition, the *New Project* wizard is expanded with support for the NHS3xxx SW framework and corresponding chips.

Note:

It is recommended to use the default location to install the LPCXpresso IDE to.

Warning:

Either install the LPCXpresso IDE in a path where you are sure you have sufficient rights; or launch the LPCXpresso executable with administrative rights before installing this plugin. Lack of write rights will fail the installation but this will not be reported and only become apparent when trying to establish a debug session with the NHS3100.

Warning:

Be sure to install the plugin in the correct version of the LPCXpresso IDE. No support can be given when using a not recommended combination of the IDE and plugin versions. Even when plugin installation completes successfully, the plugin's functionality may be fully or partially broken: e.g. building may still succeed while downloading a firmware image fails in unpredictable ways.

Installation steps:

- *Help > Install New Software…*

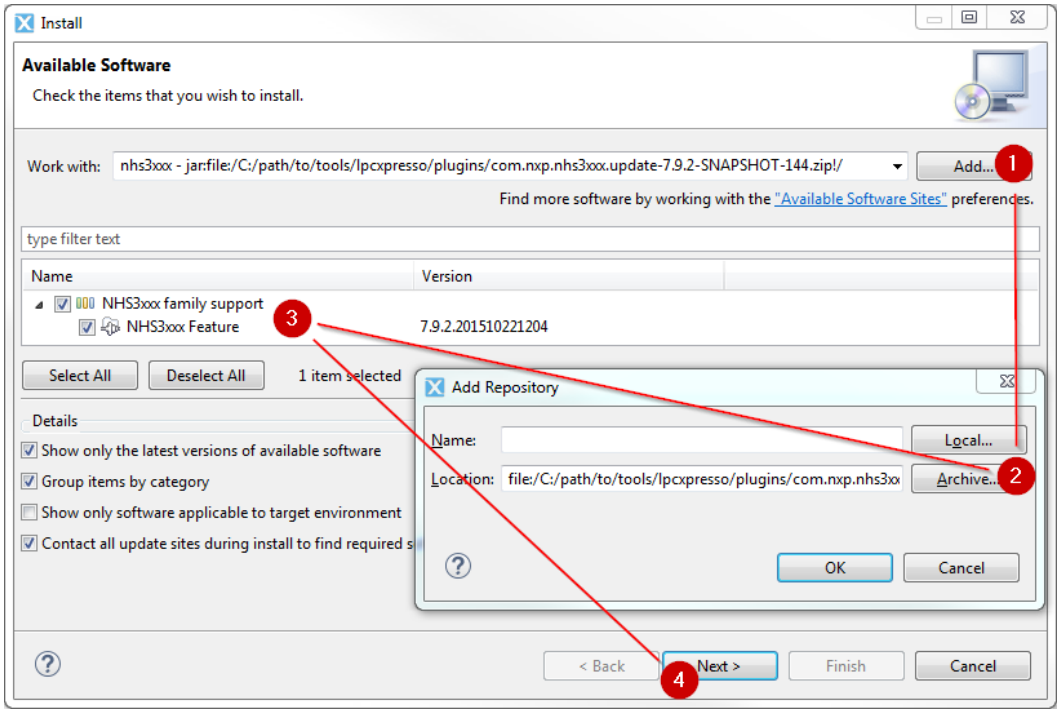

Figure 3: NHS3xxx plugin installation

- Locate the plugin *com.nxp.nhs3xxx.update-xxxxx.zip* in the SDK release under the *tools/lpcxpresso/plugins* folder by following these steps also outlined in Figure 3: *Add…* (1) the *Archive…* (2) as an *Available Software Site*, select the presented *NHS3xxx Feature* (3), and click *Next* (4).
- Finish the installation by accepting the license agreement. A warning dialog and a request to restart the LPCXpresso IDE will pop-up. Click *OK* and *Yes* where appropriate; when the LPCXpresso IDE starts up again, full NHS3xxx support is available in your installation.

Note:

- You may speed up the installation process by unchecking the option *Contact all update sites during install to find required software* – the plugin does not depend on other plugins that must be installed up front.

2.3 Flash Magic

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

The use of this tool is not enforced, but will help to quickly program ICs using pre-built firmware images; and may at times help in recovering ICs which became inaccessible due to a bug in the SW. See the warning under 6.2 below or in the documentation of the SDK: *SW Debug Considerations* in *2000FirmwareDevelopmentDocumentation*.

2.3.1 Installation

- Download the latest version from https://flashmagic.com/.
  Be sure to use version 9.72 or higher.
- Install. When prompted during installation also proceed with the installation of the NXP LPC USB drivers.

After installation, Flash Magic is ready to be launched and used.

Figure 4: Flash Magic as it appears in the Windows Start menu

More information:

http://www.flashmagictool.com/productionsystem.html

2.3.2 Setup

1. First remove JP1 from the LPC-Link2 board – see Figure 5
2. Next connect the Demo PCB with the LPC-Link2 board, and the LPC-Link2 board with the PC.

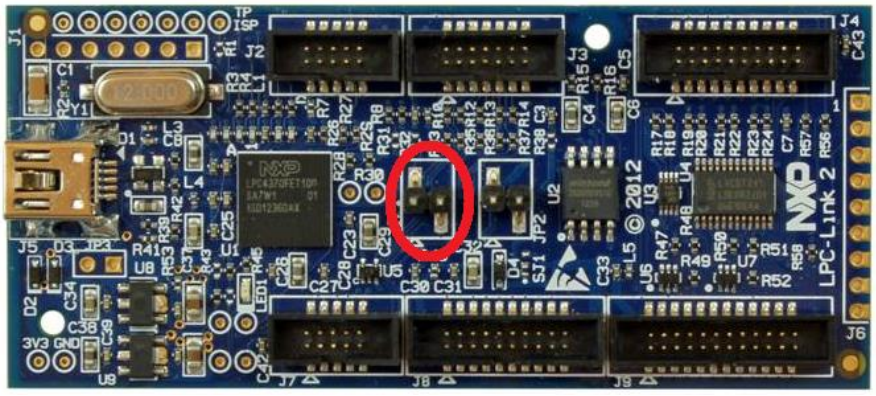

Figure 5: Highlighting the position of JP1

2.3.3 Usage

1. Use the settings as shown in Figure 6.
2. Click Start.

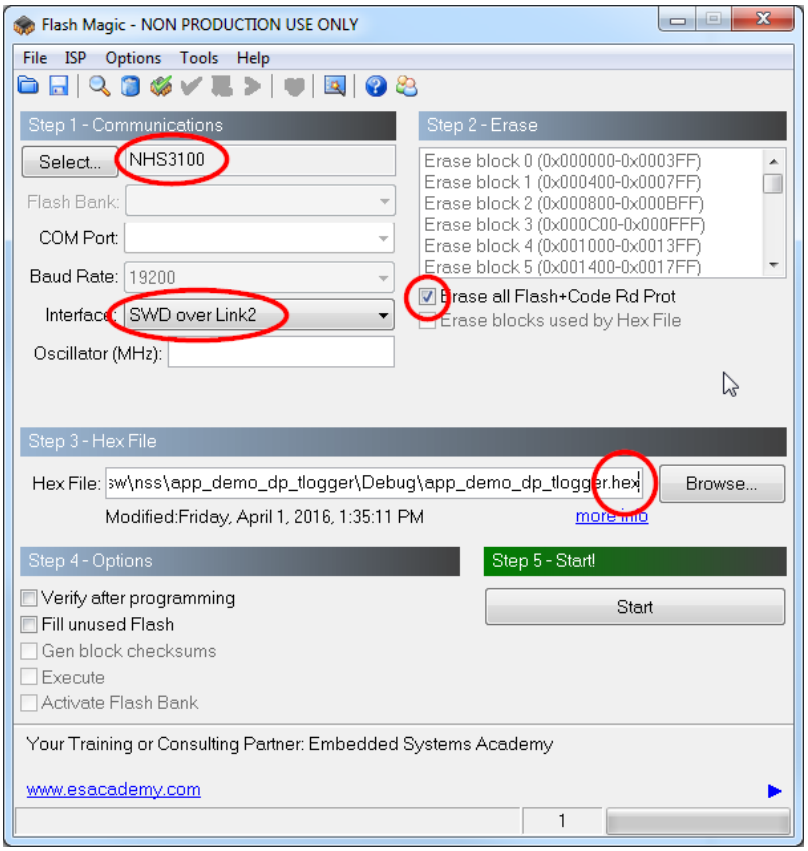

Figure 6: The recommended options within the Flash Magic tool

Note

After successfully flashing the firmware, the demo PCB needs to be reset manually; this is not done by Flash Magic.

Note

When the ARM core could not be halted, Flash Magic will alert you with an unrelated error message – see Figure 7. Either the Demo PCB is not attached or not powered, either the firmware image actively disables SWD access.

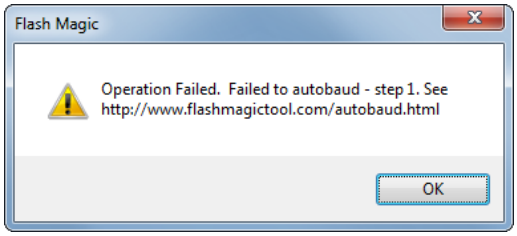

Figure 7: Pop-up error dialog when the ARM core could not be accessed.

3. Boards

3.1 Demo PCB

The Demo PCB is an IC-specific board that can be used for both the demonstration and the SW development of an NHS3xxx IC.

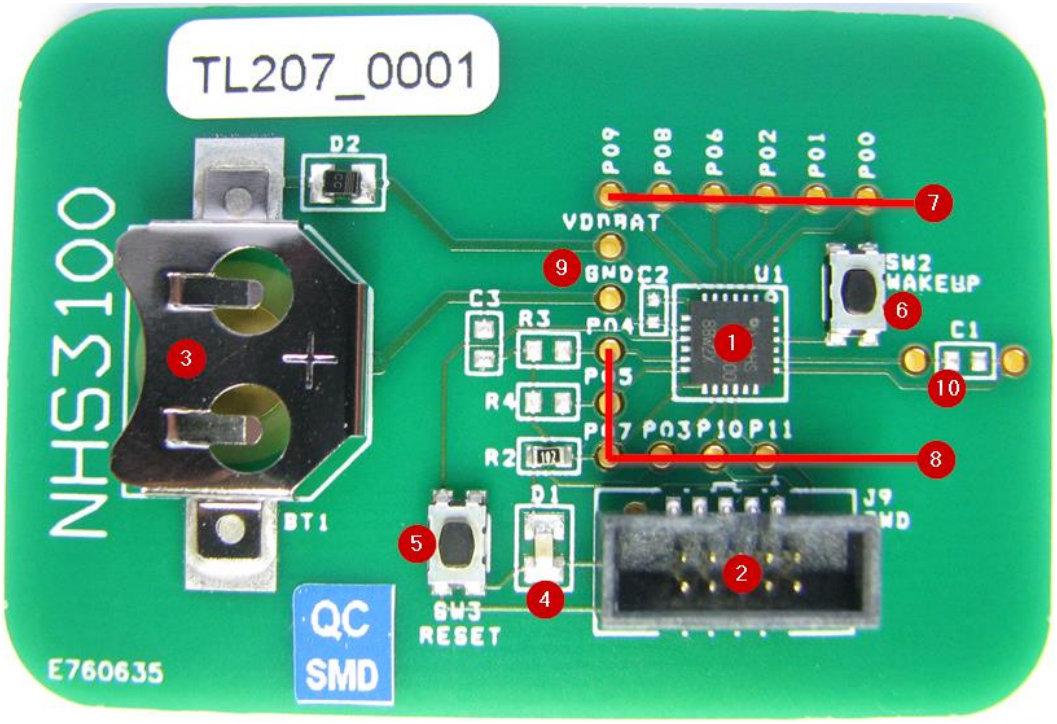

Figure 8: An NHS3100 demo PCB

It contains:

1. An NHS3100 IC in a HVQFN24 package (U1).
2. An SWD connector (J9).
3. A coin cell holder for stand-alone 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.

Plus a set of through-holes to easily access:

7. All PIOs of the IC (P0x).
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 pictured).

Note:

The suggested coin cell battery type to use is CR1225. It can be ordered internationally at e.g. http://www.newark.com/webapp/wcs/stores/servlet/Search?st=CR1225&categoryId=800000005170

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 but there remains a high chance the battery itself will be shorted, depleting the battery. Be sure to insert the battery with the positive side facing up.

3.2 LPC-Link2

LPC-Link2 is a stand-alone debug adapter that can be configured to support various development tools and IDEs by means of downloadable firmware. It is compatible with the LPCXpresso IDE and the NHS3xxx SDK using the CMSIS-DAP debugging protocol.

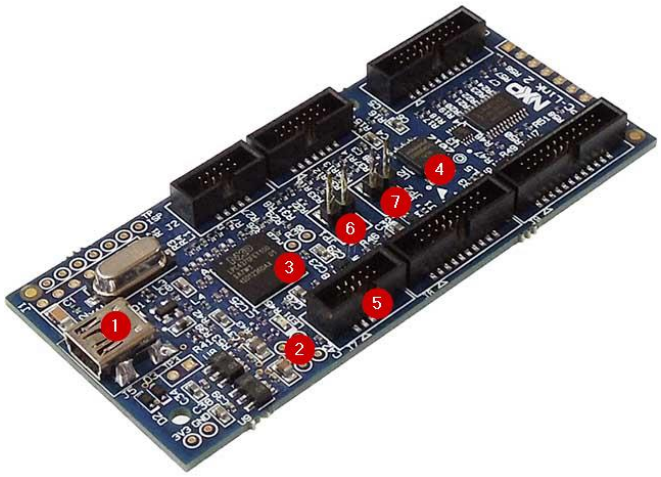

Figure 9: An LPC-Link2 board without jumpers over JP1 and JP2

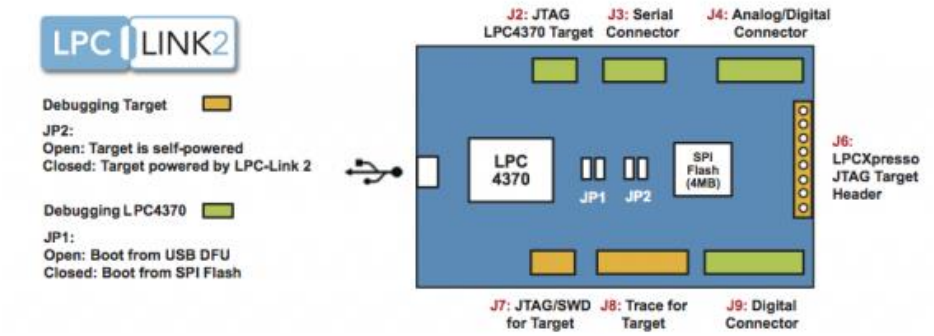

Figure 10: Image taken from https://community.nxp.com/message/630655

A few components of interest:

1. A mini USB 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: selects the SWD firmware in the onboard SPIFI flash.
   When missing: the LPCXpresso IDE will softload the SWD software via DFU.
7. JP2 to control whether the *Device Under Test* is powered via the JTAG connector.
   When mounted: the LPC-Link2 provides 3.3V supply voltage on the VTREF pin of the SWD and powers the NHS3100.

More information:

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

http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/lpc-cortex-m-mcus/software-tools/lpcxpresso-boards:LPCXPRESSO-BOARDS

Note:

4. Project setup

4.1 SW

An NHS3xxx project can be developed like any other project within the LPCXpresso IDE: the LPCXpresso IDE and the accompanying NHS3xxx plugin abstract away most specificities. A few topics deserve special mention.

4.1.1 Importing

Create a new workspace or re-use your existing workspace, and import the projects contained in the SDK release.

- In the *Quickstart Panel* view, click *Import project(s)*.
- Either 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*.
- Select the projects found. Usually all projects found are imported – as is done in Figure 11, but *lib_board_dp*, *lib_chip_nss* and *mods* are mandatory to be able to build successfully; import others as you see fit.

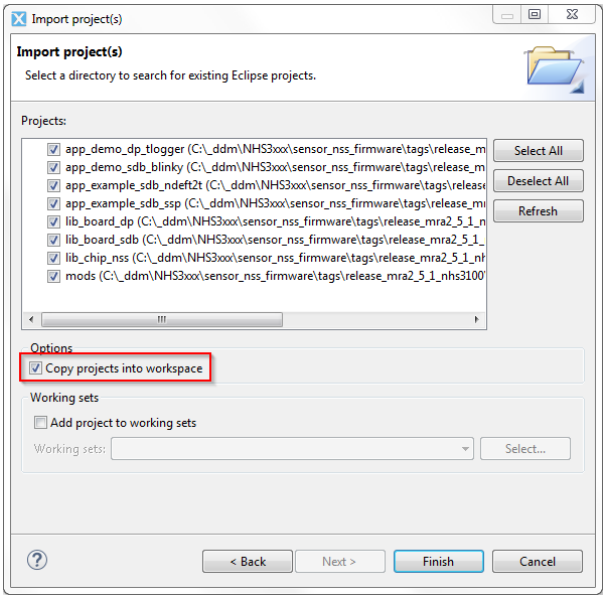

Figure 11: Importing existing projects

Note:

Pay special attention to the option *Copy projects into workspace*. It is grayed out when importing from a *Project archive (zip)*, but is selected by default when importing from a *Project directory (unpacked)*.

- If checked, the files are copied into your LPCXpresso 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.

Both options yield equal results: choose whatever best suits your style. But be careful when mixing styles: application projects using modules from the *mods* project use relative references, and if the application is imported by copying while the mods project is imported by referencing – or vice versa, building will fail with errors similar to `chip.h:35:29: fatal error: startup/startup.h: No such file or directory.`
For best results it is advised to stick to one choice.

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*

4.1.2 Building

Building can be done in various ways: via the menu *Project*, the *Quickstart Panel* view, `<CTRL>+B` shortcut key combination or the project's context menu. No additional steps are necessary: after a successful import, all imported projects should build without issues.
First select a project, after which in the *Quickstart Panel* view the *Build* command will become active, noting the selected project and the active build configuration between brackets. Now the build command can be issued: e.g. just click the now active command *Build*.

Note:

Very rarely, running make may result in an error similar to this:

```
0 [main] us 0 init_cheap: VirtualAlloc pointer is null, Win32 error 487
AllocationBase 0x0, BaseAddress 0x71110000, RegionSize 0x350000, State
0x10000
\msys\bin\make.exe: *** Couldn't reserve space for cygwin's heap, Win32
error 0
```

This is an occasional issue with the MSYS binaries that come along with an LPCXpresso installation. If this happens, you can change the DLL base address of the DLL `msys-1.0.dll` which can be found under `<install_dir>\lpcxpresso\msys\bin`. Replace the file with the file that can be downloaded from https://community.nxp.com/message/630728 – there is no need to close or restart the LPCXpresso IDE.

This does not fix the problem as it only moves the DLL base address. This means that users who do not encounter this problem, may start seeing this problem after replacing the DLL as described above.
Do not apply this workaround when you do not encounter this problem.

4.1.3 Mods

Code that does not belong in the *Chip* library nor the *Board* library, but that is still usable for a multitude of projects is grouped under the *Mods* project. The *Mods* LPCXpresso project contains SW modules that can be re-used across several other projects while supporting diversity (using `#define` precompilation flags). These modules may be included at any layer (chip, board, application) and the respective code is compiled by the project they are included in. The *Mods* project is thus a simple container project with no compilation settings enabled.

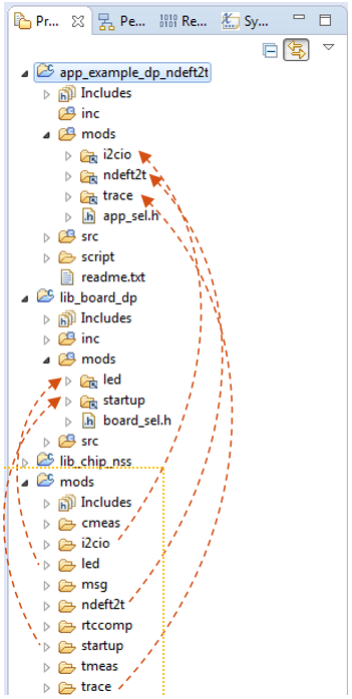

Figure 12: Code re-use by linking modules in other projects

Using this approach, the SDK can offer additional reusable blocks of code while still adhering to LPCXpresso's way of working.

An application that needs the functionality provided by one such reusable module, must create a link to it – see also Figure 12. In the LPCXpresso IDE, in the *Project Explorer view*:

- 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.
- A window pops up. Ensure that *Link to files and folders* and *Create link locations relative to*: *PROJECT_LOC* are selected – see Figure 13.

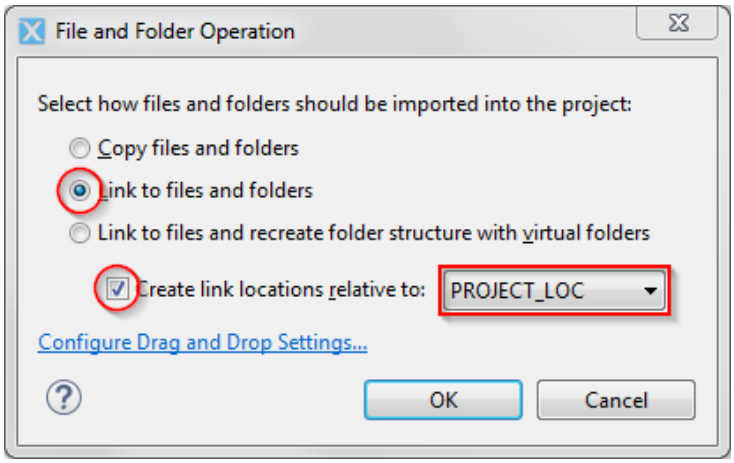

Figure 13: Creating a link to a re-usable module

The linked module is given a link icon overlay and its files are directly accessible within the application project.
The module will now compile together with the rest of the application. A simple include statement `#include "modx/modx.h"` will now give you access to the module's API.

After including one or more modules this way, check the documentation of each module to see which diversity settings to use. Each module provides reasonable defaults which you can override by means of adding `#define` precompilation flags in

`app_sel.h`. By tweaking the module, you can include or exclude functionality or 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 *Link to alternate location (Linked Folder)*. In the field below you can build up the path to the module you want to reuse: use the *Browse…* and *Variables…* buttons to achieve this in a reusable way.

4.2 HW

The following topics describe the actions that must or may be done to prepare the boards for development.

4.2.1 Demo PCB

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 4.2.2.2 JP2 below – the NHS IC will *not* become active. Only after briefly providing an NFC field near the antenna, or by pushing the *RESET* button (*SW3*), the IC wakes up and starts executing the ARM program stored in Flash.

4.2.2 LPC-Link2

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. This goes unnoticed until a debug session is attempted: no LPC-Link2 board can put to use.

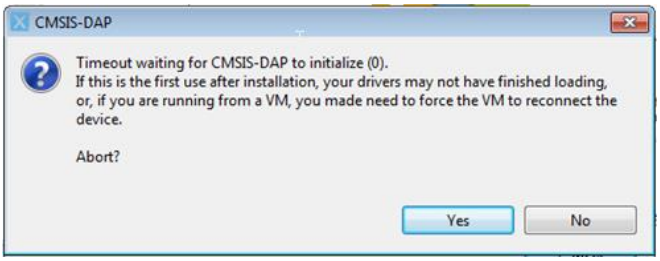

Figure 14: An outdated Windows driver version causes connection problems.

To fix this, uninstall and remove the old driver via the Windows Device Manager. After installing the new device drivers, Driver version 2.0.0.0 should be reported.

More information

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

4.2.2.1 JP1

By default, no jumper is placed over JP1.

- JP1 unmounted:
  The PC host will download the CMSIS-DAP debugging protocol firmware 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. This results in 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 CMSIS-DAP* script (under Windows directly available from the *Start Menu*). Afterwards, fit JP1 and power cycle the LPC-Link2 board.

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 NHS3xxx 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* > *LPCXpresso* > *Redlink / LinkServer*. This option is workspace dependent

To use the non-bridged variant with JP2 mounted: use the *CMSIS-DAP* script from LPCScrypt with the `NB` argument:

```
<LPCScrypt install path>\scripts\programCMSIS NB
```

Download

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

More information

LPC-Link2 Support: https://community.nxp.com/message/630655

LPC-Link2 Debug Probe firmware programming:
https://community.nxp.com/message/630661

LPC-Link2 Debug Probe Script Options: Section *7.2* in
http://cache.nxp.com/files/microcontrollers/doc/support_info/Debug_Probe_Firmware_Programming.pdf

Configuring which LPC-Link2 firmware image to soft-load:
https://community.nxp.com/thread/388968

LPC-Link2 fails to enumerate with CMSIS-DAP firmware:
https://community.nxp.com/thread/389044

4.2.2.2 JP2

By default, no jumper is placed over JP2.

- JP2 unmounted:
  The debugging target – the NHS3xxx demo PCB – must be self-powered.
- JP2 mounted:
  The NHS3xxx demo PCB is powered by the LPC-Link2 board.

Note:

Both the battery and the jumper over JP2 may be fitted simultaneously.

Warning:

Regardless whether JP2 is mounted: when testing the power-off state of the NHS3xxx (VDDBAT switched open) the SWD connector should be removed as the SWD pins will be driven high by the LPC-Link2 probe, interfering with the PMU of the NHS3xxx.

4.2.3 HW Setup

Through the LPC-Link2 board you have full SW debug capabilities on the NHS3xxx 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 15.

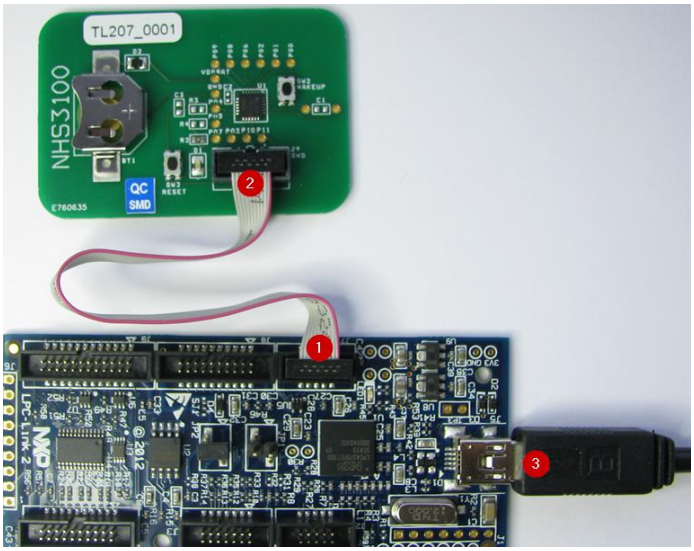

Figure 15: A Demo PCB ready for debug using an LPC-Link2

5. NHS3xxx plugin

The plugin expands the LPCXpresso IDE with several NHS-specific features. A short overview is given:

5.1.1 Chip support

The chip for which a project is built can be checked or modified. For this, go to *Project* > *Properties* > *C/C++ Build* > *MCU Settings* – see also Figure 16. With the plugin installed, the targets NHS3100, NHS3152 and NHS3153 have now become available.

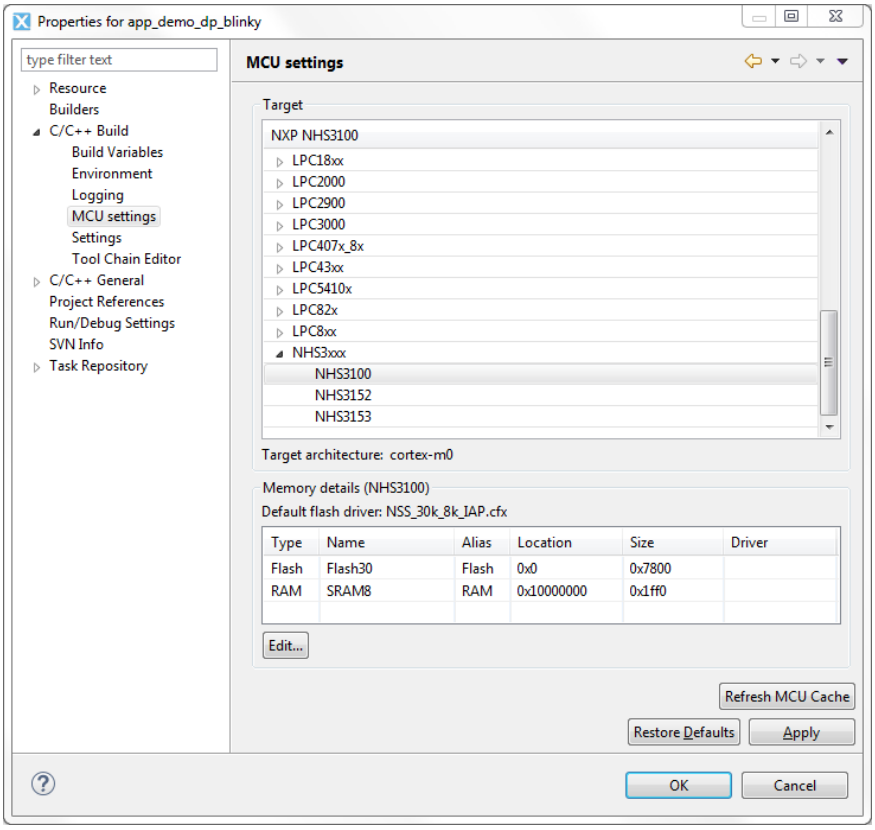

Figure 16: After installing the NHS3xxx plugin, support for several NHS ICs become available

Note:

After updating your installation with a newer version of the NHS3xxx plugin, explicitly refresh LPCXpresso's list of supported chips. This can be done via a project's context menu > *Tools* > *Refresh MCU cache*.

5.1.2 Register support

The NHS plugin provides access to all peripheral registers, together with the fields and their descriptions can be directly accessed.

As Figure 17 also clarifies:

- Start a debug session (1).
- If not already open, open the *Peripheral* view (2) via *Window* > *Show View* > *Other…* > *Debug* > *Peripherals+*
- Select the HW block(s) you want register access to (3).
- Expand the registers of interest, and hover the mouse to get the field's description as a tooltip (4). Writeable fields can be updated via the *Memory* view as well.

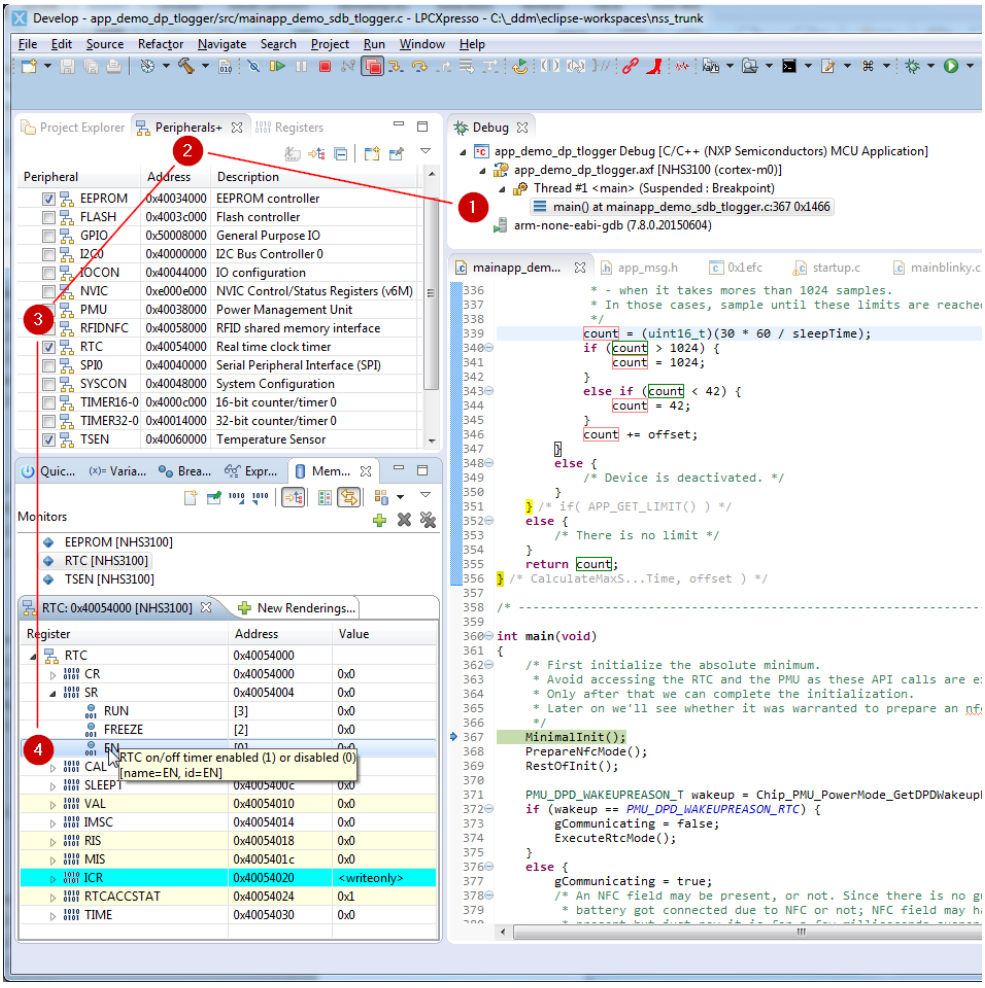

Figure 17: The NHS3xxx plugin adds support for all peripheral registers

Note:

The *Peripherals* and *Memory* views are only populated when a debug session is running.

Warning:

Viewing the registers may influence the program's behavior: e.g. the Data register *DR* in the SSP HW block *SPI0* will be read each time the view is updated, which will pop a value from the RX FIFO buffer. This value is then displayed within the LPCXpresso IDE but lost to the program under debug.

5.1.3 New project wizard

The New Project Wizard is the preferred way to quickly create new projects.

- In the *Quickstart Panel* view, click *New project…*
- Select *NHS3xxx* > *NHS3xxx* > *LPCOpen – C Project* – see Figure 18.

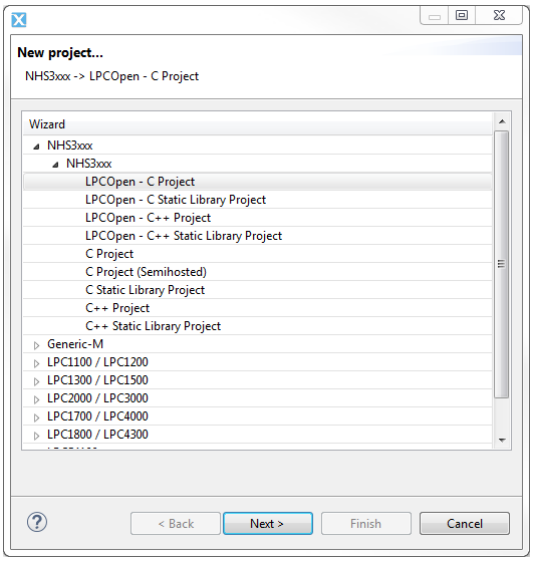

Figure 18: LPCOpen - C project is currently the sole supported project type

- Choose a project name and select the desired location.
- Select the correct NHS3xxx IC from the list – see Figure 19.
- You can accept the default options presented in the screens that follow, except for the chip and board library: specifically choose *lib_chip_nss* as the referenced *LPCOpen Chip Library Project* and *lib_board_dp* as the referenced *LPCOpen Chip Board Project*.

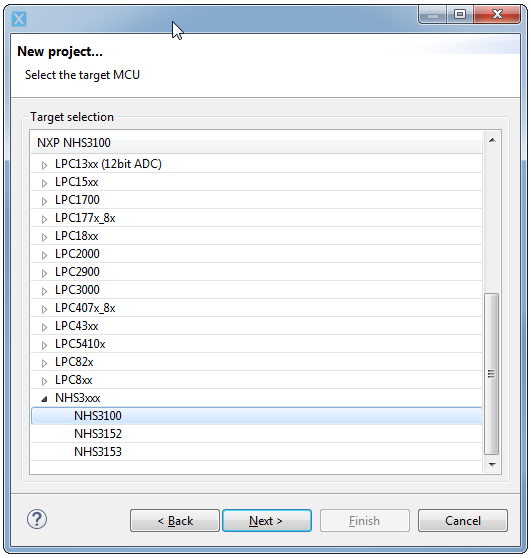

Figure 19: The NHS3xxx plugin expands the list of wizards to choose from

Note:

LPCXpresso IDE 8.1.4 is using the *GNU Tools for ARM Embedded Processors* v5.2.1, which means that the *Default Compiler language dialect* is *gnu11*.

More information:

https://gcc.gnu.org/gcc-5/

Note:

The project type currently supported is *LPCOpen – C Project*.

6. LPCXpresso Usage

Documentation about the usage of the LPCXpresso environment itself can be found under *Help* > *Help Contents*. It is highly recommended when you are new to Eclipse or the LPCXpresso IDE.

More information:

https://www.lpcware.com/system/files/LPCXpresso_IDE_Flyer.pdf

http://www.nxp.com/lpcxpresso/ide-v8-user-guide,
*§*3 LPCXpresso IDE Overview; and beyond.

6.1 Flashing

By default, starting a debug session – see 6.2 below – will also automatically program the flash. But you can also flash any `.axf` file (or `.elf` or `.bin` file) without starting a debug session:

- Make a HW connection as outlined in 4.2.3 HW Setup above.
- Within the LPCXpresso IDE, select a compatible project. The *Program Flash* icon will only become accessible after selecting a project, since some of the project settings are implicitly re-used: you must therefore select a project that re-uses the same MCU settings as the `.axf` file you want to flash.
- Click the *Program Flash* icon in the toolbar – see Figure 20.

Figure 20: Marking the location of the Program Flash button

- In the dialog that pops up, verify these settings – see also Figure 21:
  1. *Flash driver*: *NSS_30k_8k_IAP.cfx*
     This file was copied to the LPCXpresso installation directory under *<install path>/lpcxpresso/bin/Flash* during the installation of the NHS3xxx plugin – see section 2.2, above, and is already correctly filled in here if the selected project matches your MCU.
  2. *Select file*: the `.axf` or `.bin` application file to flash.

3. *Base address*: *0*

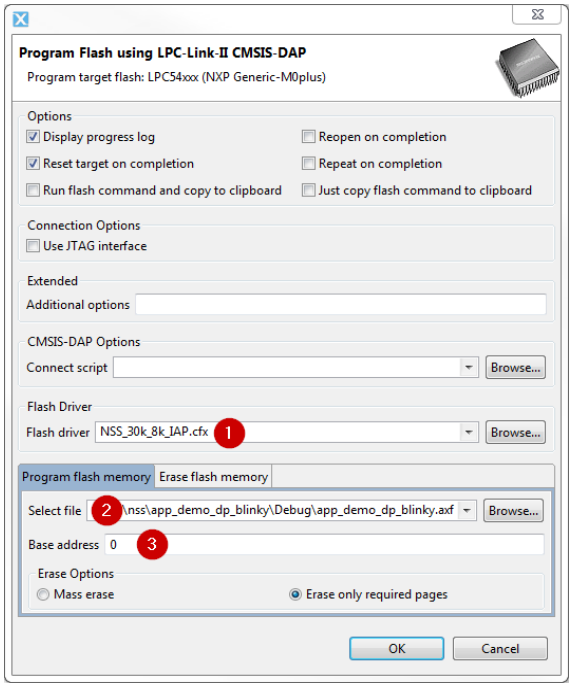

Figure 21: The Program Flash dialog

After clicking *OK*, the flash is programmed. At the end, a dialog pops up with the log and the end result.

6.2 Debugging

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

- Make a proper HW connection as outlined in 4.2.3 HW Setup above.
- Within the LPCXpresso IDE, you can start a debug session in various ways: via the menu *Run*, the *Quickstart Panel* view, shortcut key combination or the project's context menu.

Note:

Some application projects (like for example, the *blinky* application) are configured for a generic MCU as they are fit for various NHS3xxx ICs. In order to have access to the correct register map (as described in section 0, above), select your specific IC model via *Project* > *Properties* > *MCU settings* before launching the debug session.

Warning:

When the current image running in the NHS3xxx IC has put the IC in the *deep power-down* mode or in the *power-off* mode, the SWD lines have become inaccessible. Continuing or starting a debug session is then impossible.
One possible way is to toggle the *RESETN* pin which resets the IC, and then quickly start a debug session in the LPCXpresso IDE. Depending on the code and the moment it decided to go to the *deep power-down* mode, this may also be too impractical.
Another way may then be 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 FlashMagic tool: since it 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 a SW bug does not brick the device by permanently disabling the SWD lines, it may be a good idea to add debug code that implements a short timeout before going to the *deep power-down* mode or disconnecting the battery, also ensuring that the corresponding PIOs have been configured correctly for SWD functionality beforehand.

Note

To learn more about the lower 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.

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

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.

Warning:

On some Windows PCs the errors *no matching debug configuration found for NXP/NHS3100/cortex-m0*, or *"monitor" command not supported by this target* may pop up. See the dialogs given by LPCXpresso below. This occurs when the installation of the plugin was not successful. Re-install the NHS3xxx plugin using administrative rights: see also the warning under 2.2 NHS3xxx plugin.

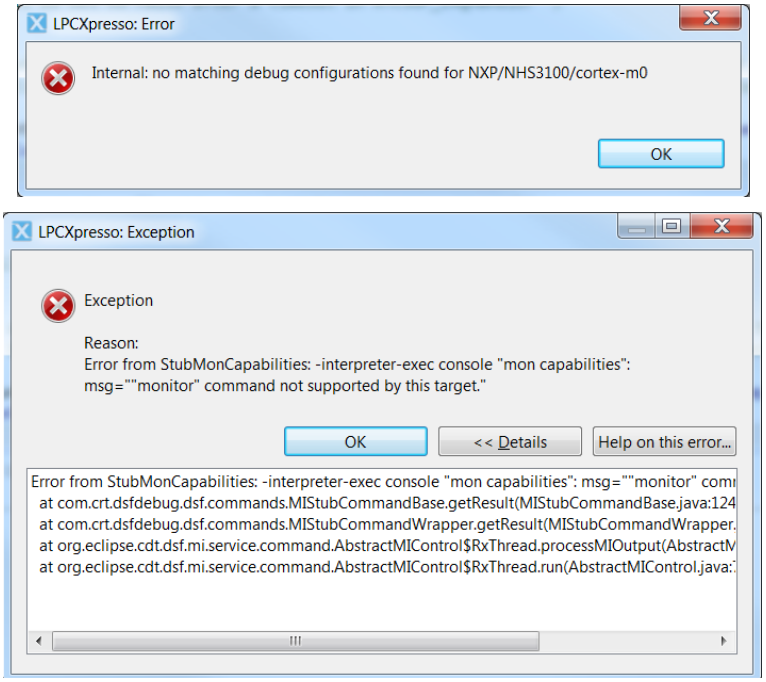

Figure 22: Debug initialization error (2)

6.3 Example

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

The full application code is:

```
#include "board.h"

int main(void)
{
    Board_Init();

    /* Optional feature: send the ARM clock to PIO0_1 */
    Chip_IOCON_SetPinConfig(NSS_IOCON, IOCON_PIO0_1, IOCON_FUNC_1);
    Chip_Clock_Clkout_SetClockSource(CLOCK_CLKOUTSOURCE_SYSTEM);

    /* Blink with a period of 250ms+250ms, or 2Hz */
    while (1) {
        LED_Toggle(LED_RED);
        Chip_Clock_System_BusyWait_ms(250);
    }

    return 0;

  }
```

To debug the code, use the *Debug 'app_demo_dp_blinky' [Debug]* command in the *Quickstart panel*. This will create a default launch configuration and – using that configuration – flash the correct image and start 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, change the Program Counter. Every debugging feature supported by ARM is available.

After resuming the application you can pause it again by e.g. adding a breakpoint inside the `while(1)` loop – just double-click the desired line in the left gutter of the file editor view, where it will halt execution immediately. See Figure 24.

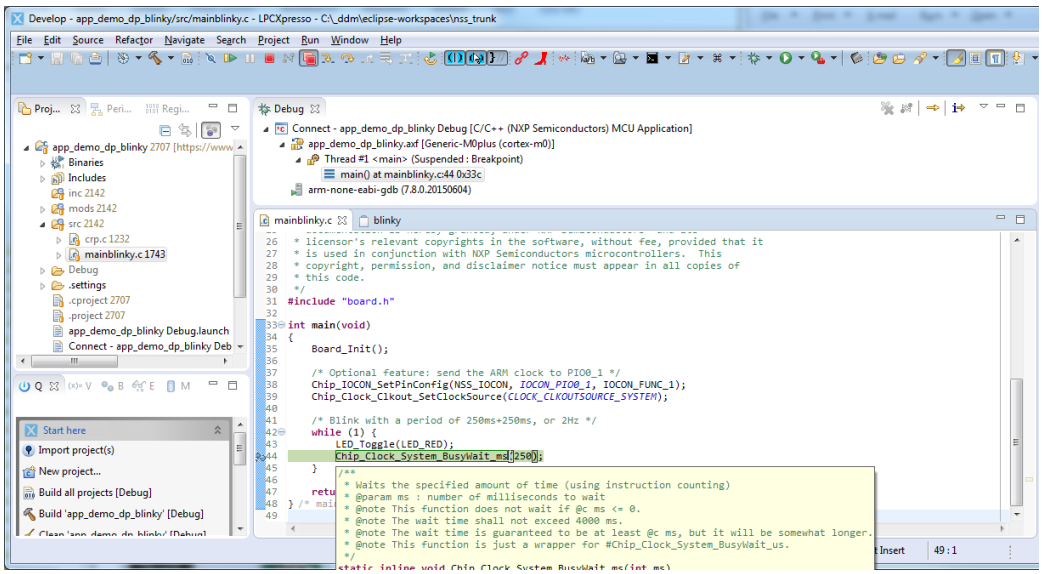

Figure 24: A debug session with ARM halted due to a breakpoint

You can stop a debugging session, which will leave the application running and afterwards launch a new debug session by creating a new launch configuration where you instruct the GDB debugger in the LPCXpresso IDE to only connect; for this 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 25. This will prevent the download of a new firmware image in the ARM and let you investigate a running instance.

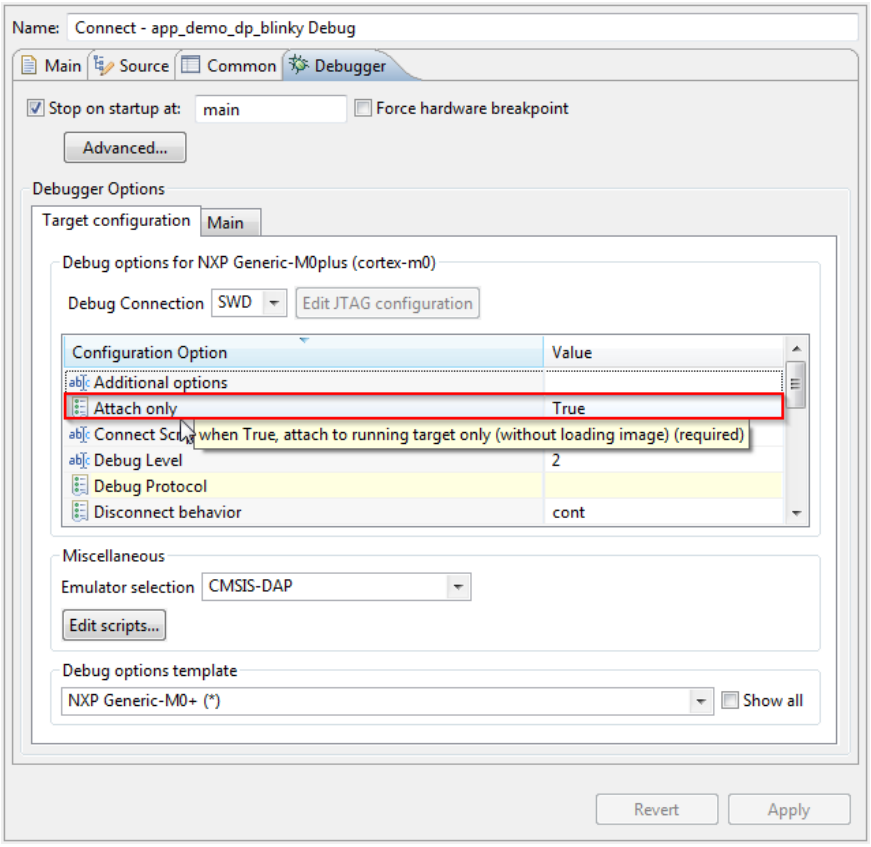

Figure 25: Creating a connect-only launch configuration

7. Conclusion

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

Further and deeper information can be found in:

- the datasheet,
- the schematics of each respective board,
- the SW documentation – near the code itself, or as generated pdf or HTML pages.

All these documents are also part of the Application Development Kit.

Also be sure to check the user manuals for the demo application(s) created for each specific IC. They give a higher level idea of the potential that can be reached with our ICs.

Further releases of the HW, the SW and the Application Development Kit will provide further improvements to HW, SW and documentation alike. To correctly set our priorities, we highly value receiving feedback.
All questions, both business and technical are greatly appreciated.

8. Legal information

8.1 Definitions

Draft — The document is a draft version only. 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 herein and shall have no liability for the consequences of use of such information.

8.2 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.

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 accepts 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.

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.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

8.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

8.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

8.5 Trademarks

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

<Name> — is a trademark of NXP B.V.

9. Contents

1. Introduction 3
1.1 Document Scope 3
1.2 Supported environments 3
1.3 Contact 3
2. Tooling 4
2.1 LPCXpresso IDE 4
2.1.1 Installation 4
2.1.2 Registration 4
2.1.3 SDK 5
2.1.4 Quickstart panel 5
2.2 NHS3xxx plugin 5
2.3 Flash Magic 7
2.3.1 Installation 7
2.3.2 Setup 7
2.3.3 Usage 8
3. Boards 10
3.1 Demo PCB 10
3.2 LPC-Link2 11
4. Project setup 13
4.1 SW 13
4.1.1 Importing 13
4.1.2 Building 14
4.1.3 Mods 15
4.2 HW 16
4.2.1 Demo PCB 16
4.2.2 LPC-Link2 16
4.2.2.1 JP1 17
4.2.2.2 JP2 18
4.2.3 HW Setup 18
5. NHS3xxx plugin 19
5.1.1 Chip support 19
5.1.2 Register support 20
5.1.3 New project wizard 21
6. LPCXpresso Usage 23
6.1 Flashing 23
6.2 Debugging 24
6.3 Example 26
7. Conclusion 28
8. Legal information 29
8.1 Definitions 29
8.2 Disclaimers 29
8.3 Licenses 29
8.4 Patents 29
8.5 Trademarks 29
9. Contents 30

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: September 5, 2016
Document identifier: 1000UM_NHS3xxx_GettingStarted