LPCXpresso

Getting started with NXP LPCXpresso

Rev. 10 — 7 April 2011 **User guide**

Document information

Info	Content
Keywords	LPCXpresso, LPC1100, LPC1200, LPC1300, LPC1700, LPC2000, LPC3000
Abstract	LPCXpresso is a new, low-cost development platform available from NXP. This document is a brief overview on how to get started with LPCXpresso.

Revision history

Rev	Date	Description
10	20110407	• Added Fig 43 and Fig 44.
9	20110301	• Updated keywords and supported products throughout • Updated Section 6.1 • Updated Section 6.2.6 • Updated Section 6.6.2 • Removed Fig 20 • Added Fig 40 and Fig 41
8	20110211	• Updated Section 3.2 • Added Fig 37, Fig 38, Fig 39, Fig 42
7	20100915	• Updated Section 3.1
6	20100712	• Added 7.1 Schematics for LPCXpresso LPC1768 target side • Added 6.1 Installing Eclipse plugins
5	20100604	• Added new products supported to Introduction section: LPC2929, LPC3250 • Removed 6.1.6 Download performance • Updated 6.4.2 Optimization section
4	20100419	• Updated Fig 45
3	20100315	• Updated Fig 36 • Added Fig 37
2	20100311	• Updated Section 3.1 • Added Section 6.2.6
1	20100111	• Initial version

Contact information

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

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

1. Introduction

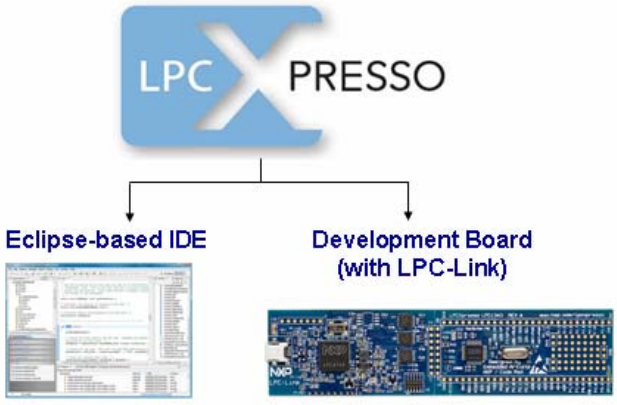
LPCXpresso is a new, low-cost development platform available from NXP. The software consists of an enhanced, Eclipse-based IDE, a GNU C compiler, linker, libraries, and an enhanced GDB debugger. The hardware consists of the LPCXpresso development board which has an LPC-Link debug interface and an NXP LPC ARM-based microcontroller target. LPCXpresso is an end-to-end solution enabling embedded engineers to develop their applications from initial evaluation to final production.

The LPCXpresso IDE, powered by Code Red Technologies (www.code-red-tech.com/lpcxpresso/), is based on the popular Eclipse development platform and includes several LPC-specific enhancements. It is an industry-standard GNU toolchain with an optimized C library that gives engineers all the tools necessary to develop high-quality software solutions quickly and cost-effectively. The C programming environment includes professional-level features. There is syntax coloring, source formatting, function folding, on- and offline help, and extensive project management automation.

The LPCXpresso target board, jointly developed by NXP, Code Red Technologies, and Embedded Artists (http://www.embeddedartists.com/products/lpcxpresso/), includes an integrated JTAG debugger (LPC-Link), so there's no need for a separate JTAG debug probe. The target portion of the board can connect to expansion boards to provide a greater variety of interfaces, and I/O devices. The on-board LPC-Link debugger provides a high-speed USB to JTAG/SWD interface to the IDE and it can be connected to other debug targets such as a customer prototype. Users can also use the LPCXpresso IDE with the Red Probe JTAG adapter from Code Red Technologies.

Supported LPC products on the LPCXpresso platform:

- LPC1100: All products supported
- LPC1200: All products supported
- LPC1300: All products supported
- LPC1700: LPC1751 LPC1752 LPC1754 LPC1756[1] LPC1758[1] LPC1764 LPC1765[1] LPC1766[1] LPC1767[1] LPC1768[1] LPC1769[1]
- LPC2000: LPC2109 LPC2134 LPC2142 LPC2362 LPC2929
- LPC3000: LPC3130 LPC3250



1. The LPCXpresso platform will allow the user to build an executable of any size but it will restrict code download to 128 kB. These specific products contain more than 128 kB of flash memory. Easy upgrade options are provided for higher memory configurations. Please visit the LPCXpresso website for more details.

1.1 LPCXpresso IDE

LPCXpresso's IDE is a highly integrated software development environment for NXP's LPC Microcontrollers, which includes all the tools necessary to develop high quality software solutions in a timely and cost effective fashion. LPCXpresso is based on Eclipse with many LPC specific enhancements. It also features the latest version of the industry standard GNU tool chain with a proprietary optimized C library providing professional quality tools at low cost. The LPCXpresso IDE can build an executable of any size with full code optimization and it supports a download limit of 128 kB after registration. LPCXpresso supports the full embedded product design cycle by moving beyond chip evaluation boards and supporting development on external target boards.

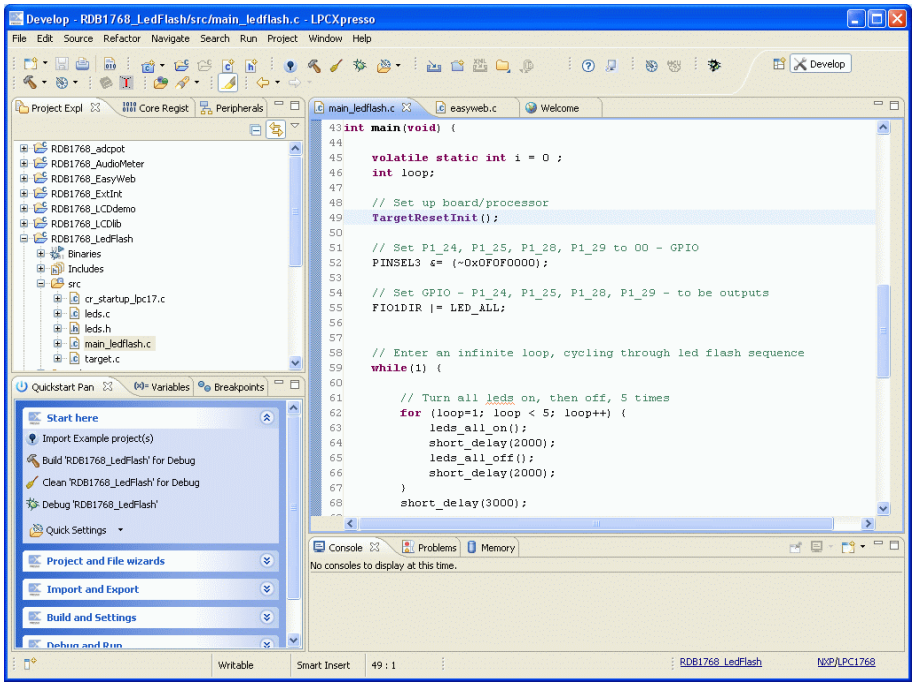


Fig 1. LPCXpresso IDE

1.2 LPCXpresso development board

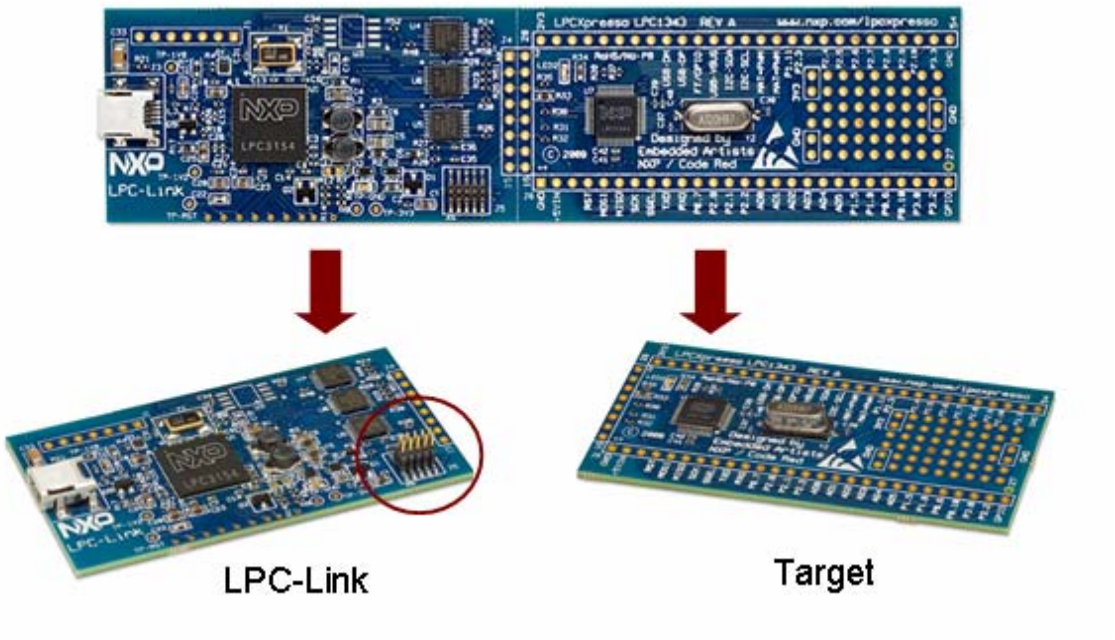


Fig 2. LPCXpresso development board

1.3 LPC-LINK JTAG/SWD debugger

The LPCXpresso board contains a JTAG/SWD debugger called the “LPC-Link” and a target MCU. LPC-Link is equipped with a 10-pin JTAG header (highlighted on the above image) and it seamlessly connects to the target via USB (the USB interface and other debug features are provided by NXP’s ARM9 based LPC3154 MCU). Cutting the tracks between the LPC-link and the target will make the LPC-Link a stand-alone JTAG debugger. This enables the LPCXpresso platform to be connected to an external target and used to develop for a wide variety of NXP’s Cortex-M0, Cortex-M3, and ARM7/9 based applications. Currently supported microcontroller products include LPC1700, LPC1300, LPC1200, and LPC1100 series and specific members of the LPC2000 and LPC3000 families.

1.4 Integrated evaluation target

The target includes a small prototyping area and easily accessible connections for expansion. The LPCXpresso board with target can be used

- On its own for software development and benchmarking
- Connected to an off-the-shelf baseboard for rapid proof-of-concepts
- Connected to customer-designed board for a full prototype

1.5 LPCXpresso partners

NXP has partnered with Code Red Technologies and Embedded Artists for the LPCXpresso platform. For added flexibility and higher memory configurations, the

LPCXpresso platform can easily be upgraded to include full-blown suites from Code Red Technologies and more advanced hardware kits from Embedded Artists. Please visit the LPCXpresso webpage for more information.

2. Evaluate, explore and develop

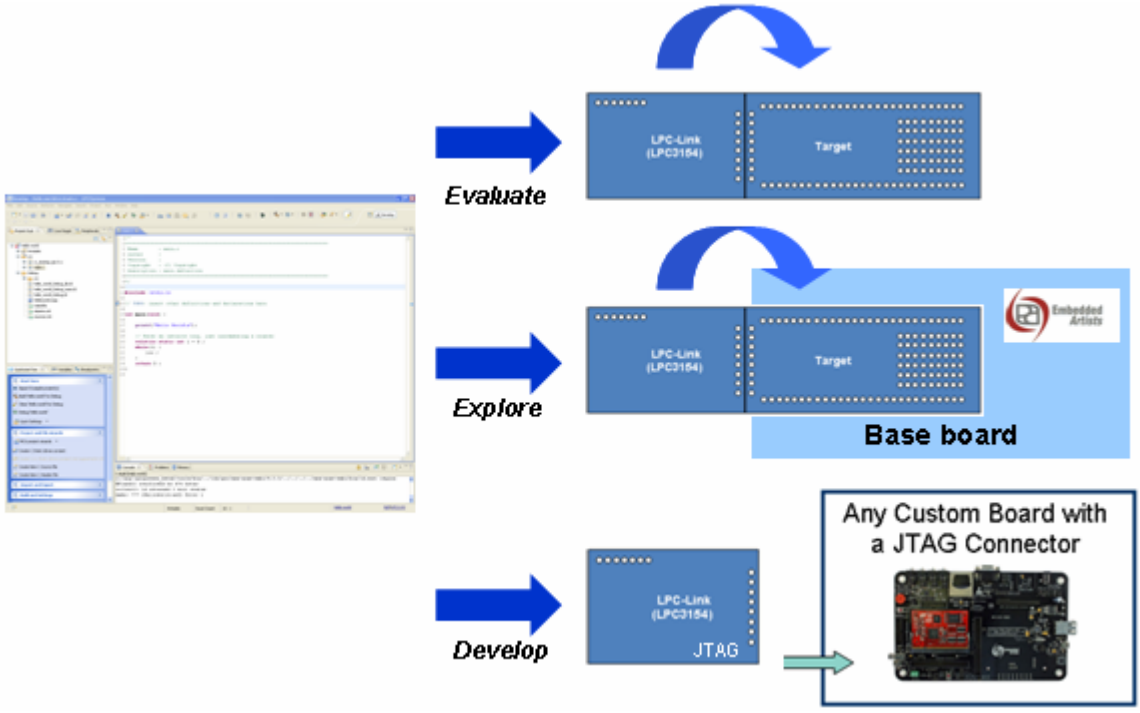


Fig 3. Product development stages

Users can envisage three stages from evaluation to product development. During evaluation, features and peripherals of the target MCU can be easily tested with the prototyping area and easily accessible connections on the target board. Complementing the target board are also easy-to-use example projects and a handy Getting Started guide. For rapid proof-of-concepts, users can get an off-the-shelf base board from Embedded Artists and quickly explore the next level of applications. And finally LPCXpresso users can seamlessly develop their final application by using the LPC-Link's 10-pin JTAG connector to attach any JTAG-capable custom board. This way, users can now enjoy the same user experience right from evaluation to product development.

3. Installation

3.1 System requirements

Operating System	Microsoft® Windows - XP 32-bit or 64-bit (SP2 or greater) Microsoft® Windows - Vista 32-bit or 64-bit Microsoft® Windows - Windows 7 32-bit or 64-bit Linux - Ubuntu 9 and 10 Linux - Fedora Core 12 and 13
System RAM	512 MB minimum (1 GB recommended)
Hard Disk	225 MB of available space.
Screen/Display Adaptor	1024x768 minimum recommended
Internet Connection	High-speed internet is recommended to download and register the software

Note: Desktop virtualization tools supporting a linux or Windows guest with USB support can be used to run LPCXpresso on other computing platforms.

3.2 Installation process

LPCXpresso is installed into a single directory, of your choice. Multiple versions can be installed simultaneously without any issues. The installation process is to double-click the installer file after downloading. Then click “next” on the setup wizard. To install under linux, the downloaded file should be marked as executable first using chmod +r.

Fig 4. Setup wizard

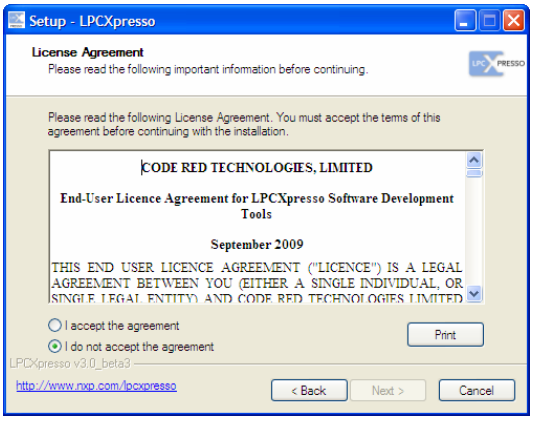


Fig 5. Setup wizard

Read the license agreement then click next. There are a number of other screens on the setup wizard, but generally the default options can be accepted. After the install, an information file will be displayed. Click “Next.” Congratulations! Your LPCXpresso installation is complete!

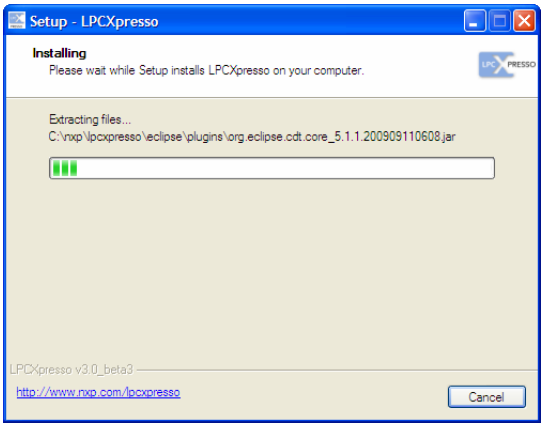


Fig 6. Setup wizard

3.3 Activation

To activate your product from LPCXpresso, choose Help->Product activation->Create serial number and register. Once the wizard is open, click “Copy to clipboard” to copy the LPCXpresso serial number into the clipboard. This serial number is based on your machine’s hardware and operating system configuration, but contains no personally identifiable information. Now click the button to open the registration activation page. This should display a web form. After completing the form, you will receive an activation code via email within a few minutes. Highlight the activation code in your email program, and select Copy to place it into the Windows clipboard. Now, choose select Help->Product activation->Enter activation code from within LPCXpresso. Paste the product activation code into the Product activation dialog by right clicking in the Activation code field and choosing “Paste.” Then click the “OK” button. You should receive a dialog confirming acceptance of the activation code. It is also possible to complete LPCXpresso activation on a PC that is offline as long as another PC has access to the Internet. Refer to Fig 7 for the process.

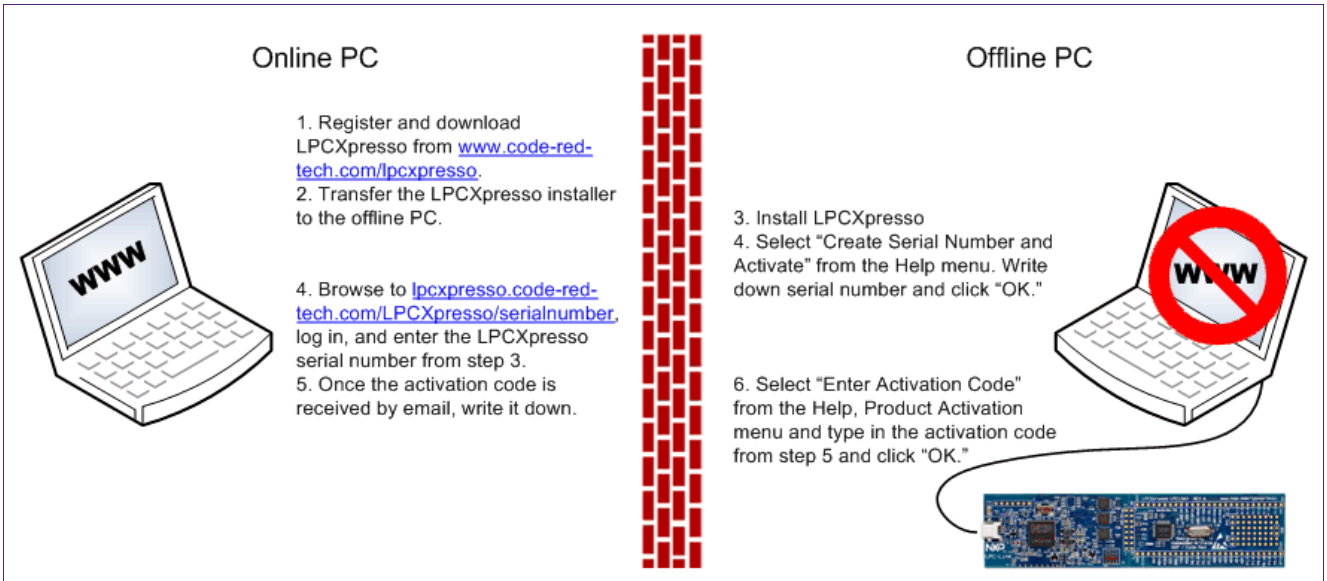


Fig 7. LPCXpresso activation with offline PC

4. Getting familiar with the LPCXpresso IDE

LPCXpresso IDE is based on the Eclipse IDE framework and many of the core features are described well in generic Eclipse documentation and in the help files found in the help menu of the product. Further documentation and pointers to useful documents are also available on the Code Red Technologies Wiki at http://lpcxpresso.code-red-tech.com/LPCXpresso/softwareknowledgebase.

4.1 Layout of the LPCXpresso desktop

LPCXpresso's Desktop contains many windows. Each window is called a View, because it displays a particular view of data in the LPCXpresso environment. This data could be source code, hex dumps, disassembly, memory contents, or more. Views can be opened, moved, docked, and closed, and the layout of the currently displayed Views can be saved and restored. A specific configuration of Views is called a 'Perspective.' Typically, LPCXpresso operates in a single perspective under which both the code development & debug sessions operate as shown on the next page. The single perspective greatly simplifies the Eclipse environment and enhances the entire LPCXpresso experience.

All Views in the Perspective can be moved around by dragging and dropping. If a View is accidentally closed, it can be restored by selecting it from the Show View dialog. The Show View dialog can be opened from the Show View Other... option in the Window menu.

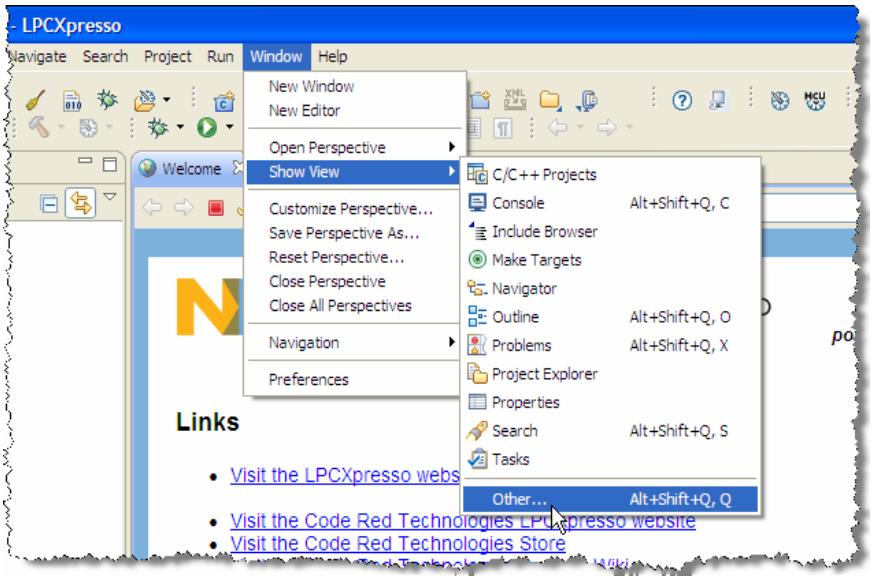


Fig 8. Show view/other menu

4.1.1 Single perspective (code development)

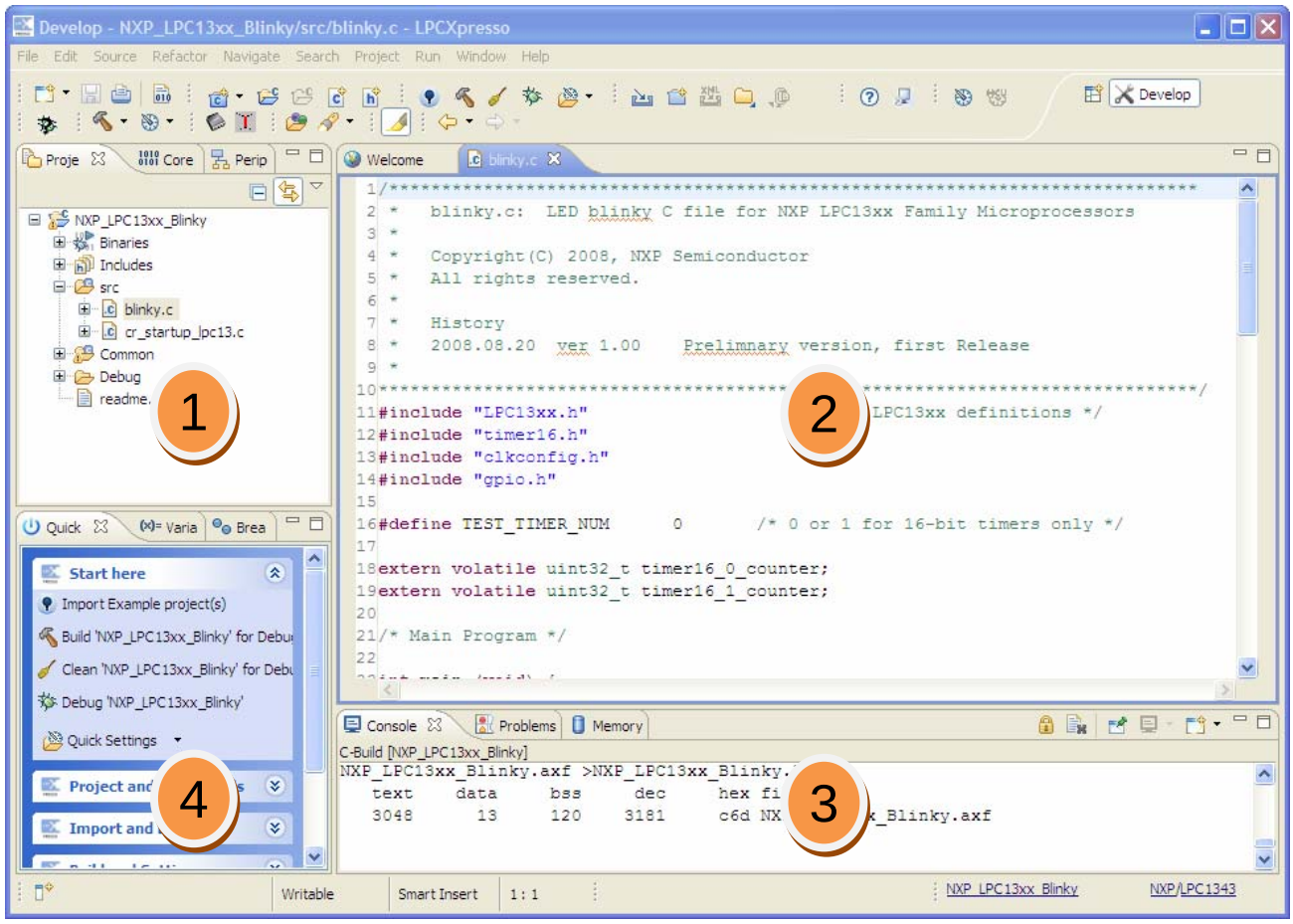


Fig 9. Single perspective (develop)

1. Project Explorer View: The ‘Project Explorer’ gives you a view of all the projects in your current ‘Workspace’. A ‘Workspace’ is a collection of projects that are stored in a single Workspace Directory on your computer.
2. Editor: On the upper right is the editor which allows modification and saving of source code as well as setting breakpoints in debug mode.
3. Console and Problems Views: On the lower right are the Console and Problems Views. The Console View displays status information on compiling and debugging, as well as program output. The Problem View (available by changing tabs) shows all compiler errors and will navigate the Editor View to the error location.
4. Quick Start View: Below, the ‘Quick Start’ view has fast links to commonly used features. This is the best place to go to find options such as Build, Debug, and Import.

4.1.2 Single perspective (debug)

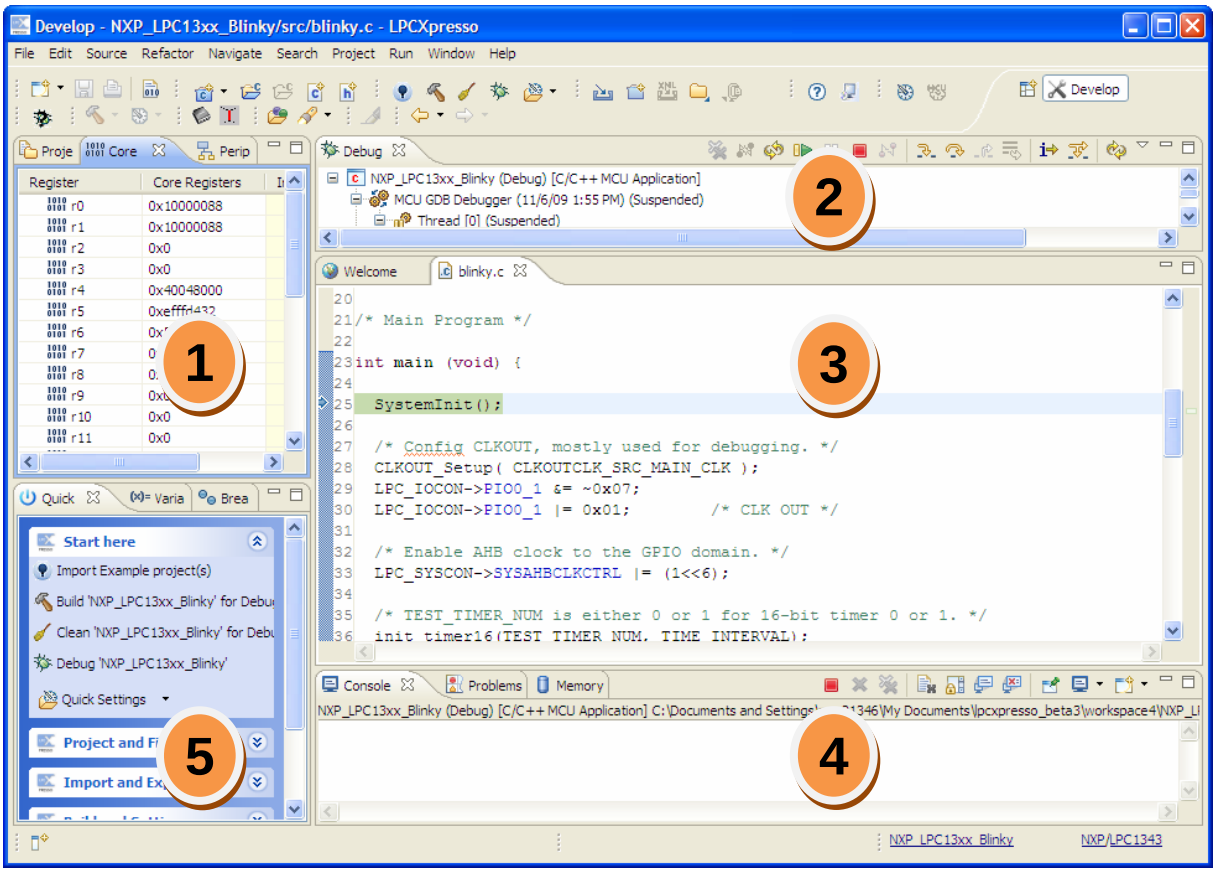


Fig 10. Single perspective (debug)

1. Core Register View: This shows all of the registers in the processor core. Registers that have changed from step to step are highlighted in yellow.
2. Debug View: This shows you the stack trace and the debug toolbar. Using the icons at the top of the view, you can step through code or execute at full speed. In the ‘stopped’ state, you can click on any particular function and inspect its local variables in the right hand panel on the Variables tab.
3. Editor: In here you will see the code you are executing and can step from line to line. By pressing the ‘i’ icon at the top of the Debug view, you can switch to stepping by assembly instruction. Clicking in the left margin will set and delete breakpoints.
4. Console View: On the lower right is the Console View. The Console View displays status information on compiling and debugging, as well as program output.
5. Quick Start View: Below, the ‘Quick Start’ view has fast links to commonly used features. This is the best place to go to find options such as Build, Debug, and Import.

4.1.2.1 Peripheral views

LPCXpresso includes full, annotated and interactive debug views of all the peripherals. Access to the views is found on the Peripherals View (click the Peripherals tab) behind the Core Registers view. Each peripheral can be selected, and it is displayed in the 'Memory' view which is located behind the 'Console' view at the bottom of the LPCXpresso desktop.

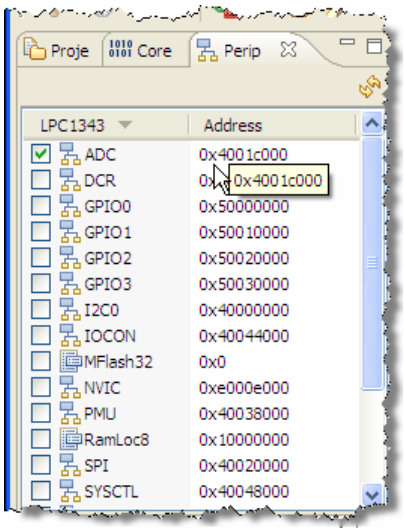


Fig 11. Display of peripheral view showing selection of ADC peripheral

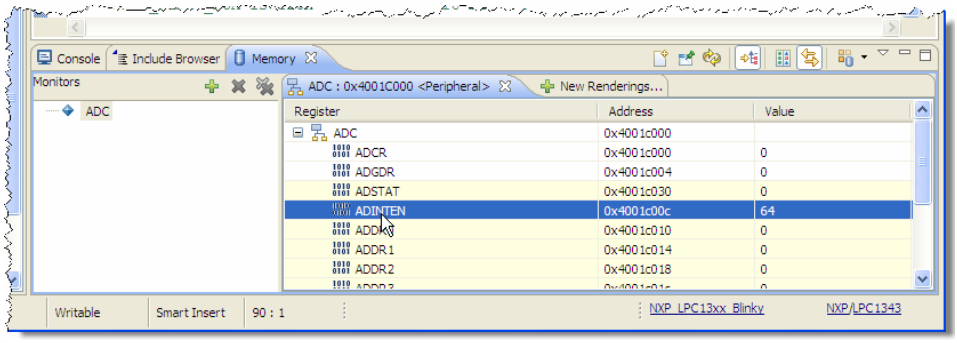


Fig 12. Display of memory view showing detail of ADC peripheral registers

4.2 Connecting the target

To begin development, the LPCXpresso can be connected to a PC using a USB 2.0 A/Mini-B cable.

Fig 13. USB 2.0 A/Mini-B cable

If you are debugging a prototype board or a target containing a different MCU, see the Appendix for a pinout to connect the debugger section of the LPCXpresso to an external target.

5. Blinky: Build, download and debug

5.1 Importing the blinky project using the Quickstart panel

Example projects make great starting points for your own embedded projects.

In LPCXpresso, Quickstart Panel brings together all of the most frequently used operations for Embedded Microcontroller development. The Quickstart Panel is by default located in the lower left of LPCXpresso.

- On the Quickstart Panel, click on the ‘Start Here’ sub-panel, and click on ‘Import Example project(s)’.

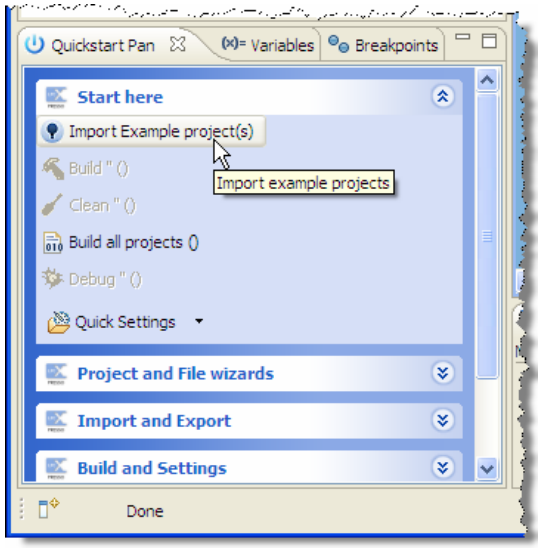


Fig 14. Quickstart panel

- Click on ‘Browse’ and find the LPCXpresso examples directory. If this is the first time you have used the Importer, then it should automatically be at the right directory; otherwise, the default install directory is ‘C:\nxp\lpcxpresso\Examples’. Once in this directory, pick an archive for your particular evaluation board and click ‘Open’ and then ‘Next’.

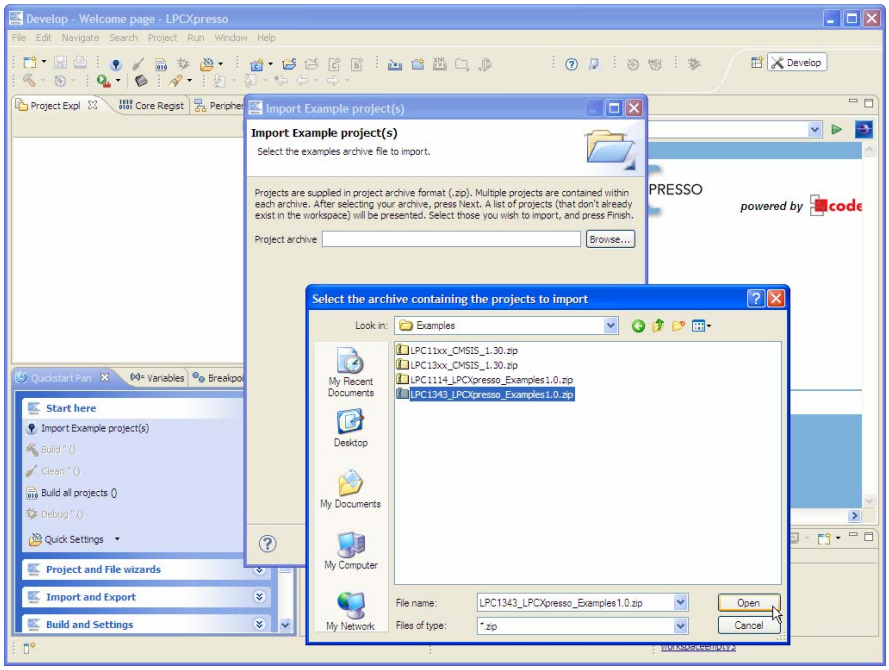


Fig 15. Examples directory

- You will then be presented with a list of projects within the archive as shown. Make sure that the ‘Blinky’ Project and the CMSIS Project are checked.
- Click ‘Finish’ and the two Projects will be imported into the current workspace by LPCXpresso.
- Now click ‘Build all projects (Debug)’ on the Quickstart Panel to build the Blinky example and CMSIS library.

5.2 Debugging/running ‘blinky’ on your LPCXpresso board

In LPCXpresso, when you start to debug, your program will automatically download to the target and be programmed into flash memory.

To start debugging on your target, simply highlight the project in the Project Explorer and then in the Quickstart Panel select ‘Debug project ‘Blinky’. LPCXpresso may then show a dialog asking you to select which executable (Release or Debug) to use. Select ‘Debug,’ Then click on ‘OK’ to continue with the download and debug of the ‘Debug’ build of your project.

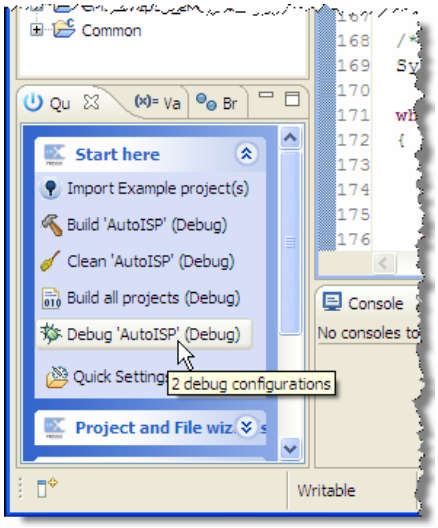


Fig 16. Debug

You may also enter debug mode by clicking the bug icon on the top LPCXpresso toolbar.

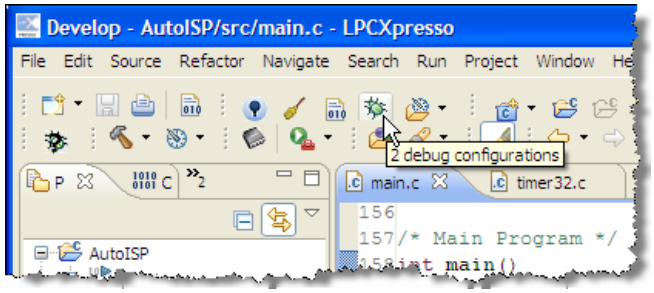


Fig 17. Bug icon

You are then presented with the debug view and toolbar and have run control over the code running on your target. The toolbar will pop up above the code window.

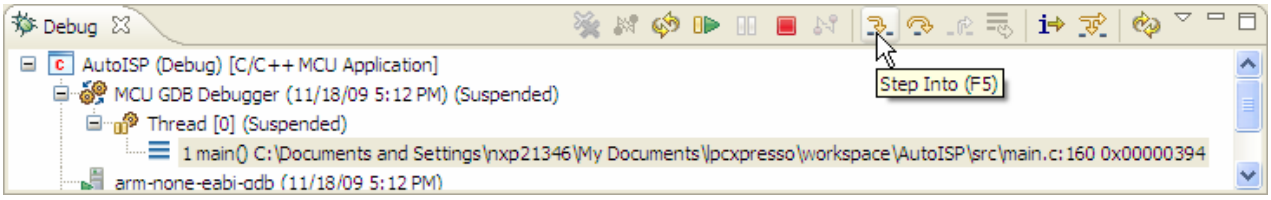


Fig 18. Debug toolbar

You can now do the following with the buttons towards the top of the ‘Debug’ view:

	Run the program.
	Step Over C/C++ line.
	Step into a function.
	Stop the debugger.
	Pause Execution of the running program.
	Instruction stepping mode (disassembly)

Fig 19. Debug buttons

6. LPCXpresso IDE tips and tricks

6.1 Installing Eclipse plugins

The LPCXpresso IDE contains many of the features of the Eclipse open-source IDE from http://www.eclipse.org. The browse and install plugin function is present in the help menu. To access it, choose Help -> Install New Software. This will display the Eclipse Install Software dialog which will allow browsing and installing of Eclipse plugins.

6.2 Debugging tips

6.2.1 Debug features not enabled

All of the LPCXpresso features are context-sensitive. If features are disabled, double-check that you are navigated into a .c file in an open project on the Project Explorer View, or some menu items and toolbar buttons may be disabled. If your workspace contains projects that create libraries such as CMSIS, please note that debug features will be disabled if you are currently editing a .c file that is part of a library project.

6.2.2 Incorrect registers displayed or errors starting debug

Make sure that the correct NXP LPC microcontroller part is selected in LPCXpresso. The current part number is displayed in the status bar at the bottom of the LPCXpresso window. It can be changed by holding down the Ctrl key and clicking. A dialog will appear allowing selection of the correct part number.

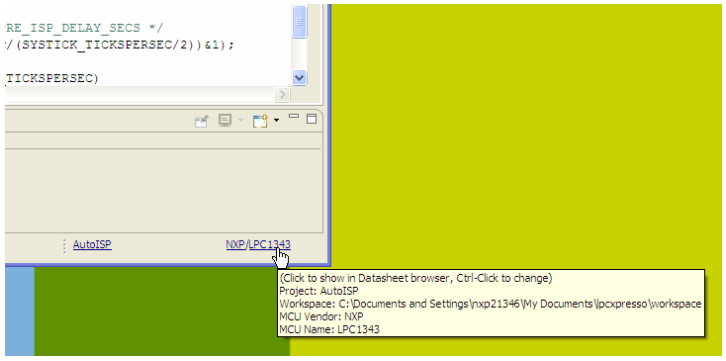


Fig 20. Current part number

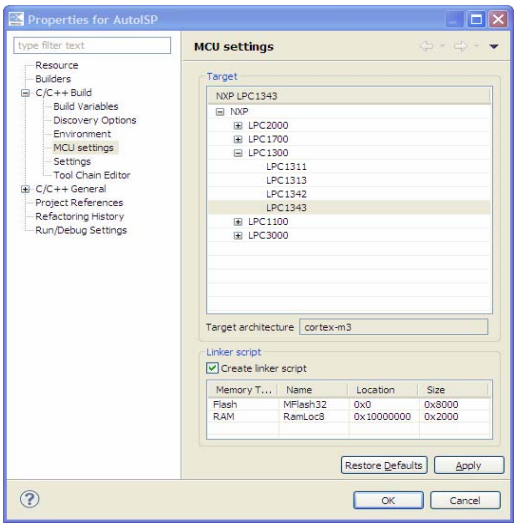


Fig 21. Selecting correct part number

6.2.3 Optimization issues

When optimization is enabled, it will reorder code. What this means is that the code from multiple C lines will be intermingled. In addition, assignments and initializations might be pulled out of loops so they are only executed once. Changes like these will make the code confusing to debug. Some symptoms you might see are breakpoints that only work the first time through, or seeing the debugger's current line indicator fail to advance or even move backwards when you click step. It is best to always use –O0 for debugging. Since optimization can make such a big difference in code size and performance, it is a good idea to test your project with optimization and plan for a final build that is optimized.

6.2.4 Displaying assembly instructions

Click the i-> icon. This changes the Instruction Stepping Mode to step by processor instructions, rather than source lines. This also shows the disassembly view around the current instruction.

6.2.5 Exiting debug mode and stopping debugging

To stop debug press the 'Stop' button (red square) shown in the toolbar at the top of the debug view.

6.2.6 Recovery of board

After playing around with the LPCXpresso board, especially when trying out new PLL settings, reconfiguring the SWDIO/SWDCLK pin functions, disabling AHBCLKCTRL bits, or trying power down modes, the board may be disabled and no longer enter debug mode. This is caused by code on the on-board flash that incorrectly disables the system clocks or the debug port soon after reset before the debugger can connect to the core. The easiest solution to this is to load a working project into LPCXpresso, ground the ISP pin (see the chip User's Manual for details) and then try to enter debug mode.

Grounding the ISP pin during reset will put the target MCU into In-System Programming (ISP) mode. It will wait for a command through the serial port or the USB port. This temporarily prevents the troublesome code in flash from starting. Although ISP is designed to enable serial and USB updates, while ISP is running, the LPCXpresso toolchain is able to connect to the Cortex core and reprogram the flash. After the flash is

reprogrammed, disconnect the ISP pin (pull it high or allow it to float) and stop debugging. Now you should be able to debug code again.

6.3 Datasheet browser

The LPCXpresso IDE comes with an integrated web browser that will direct viewers to the datasheet of the device. Just click on the button below to see the browser in action.

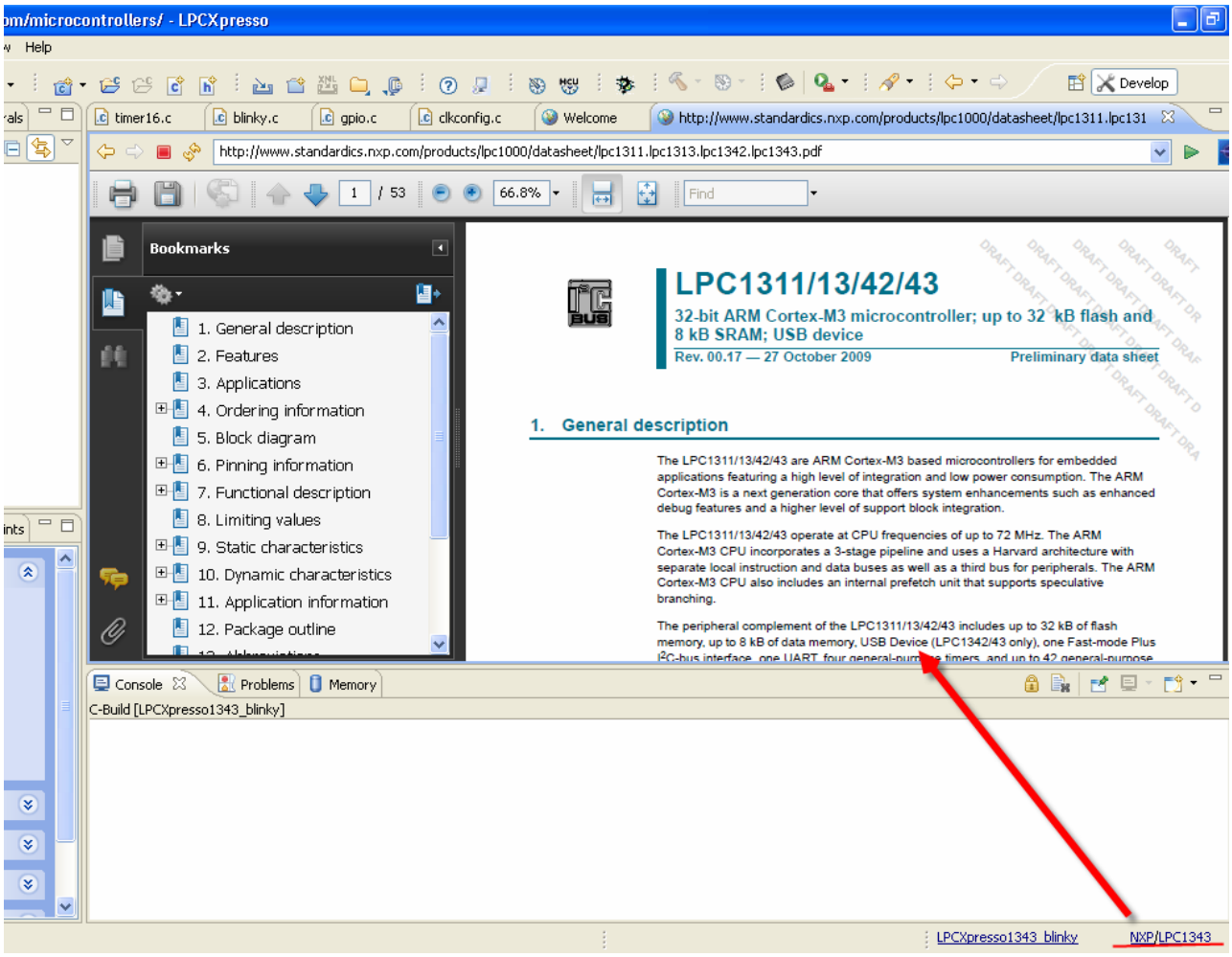


Fig 22. Integrated web browser

6.4 Code size

6.4.1 printf

When optimizing a project for size, if you are using printf, make sure that Redlib is selected as the standard library in the Projects Properties dialog. This option must be set in two different places, to configure both the header files, and the libraries. It should also be set both for the Debug target and the Release target. To get to the header file option, select settings in the C/C++ Build folder in the tree on the left. Make sure that the Tool Settings tab is active, and then select Target under MCU C Compiler in the tree on the

right. Under the Configuration drop-down, select the Debug or Release target. Under Use headers for C library, choose Redlib. This is currently the default setting.

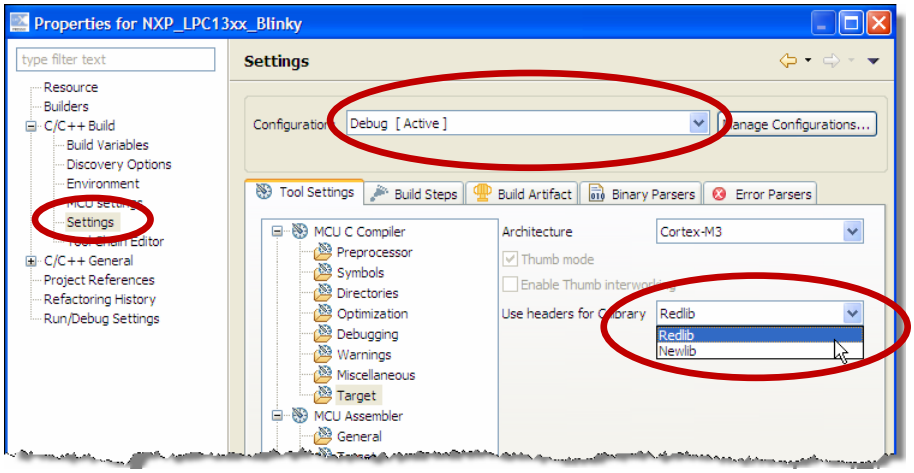


Fig 23. Header file option

To reconfigure the standard library setting for the linker, in the same dialog, select Target under MCU Linker. In this case, there are several options for Redlib. The default is Redlib (semihost) which allows full I/O to the PC through the LPCXpresso console.

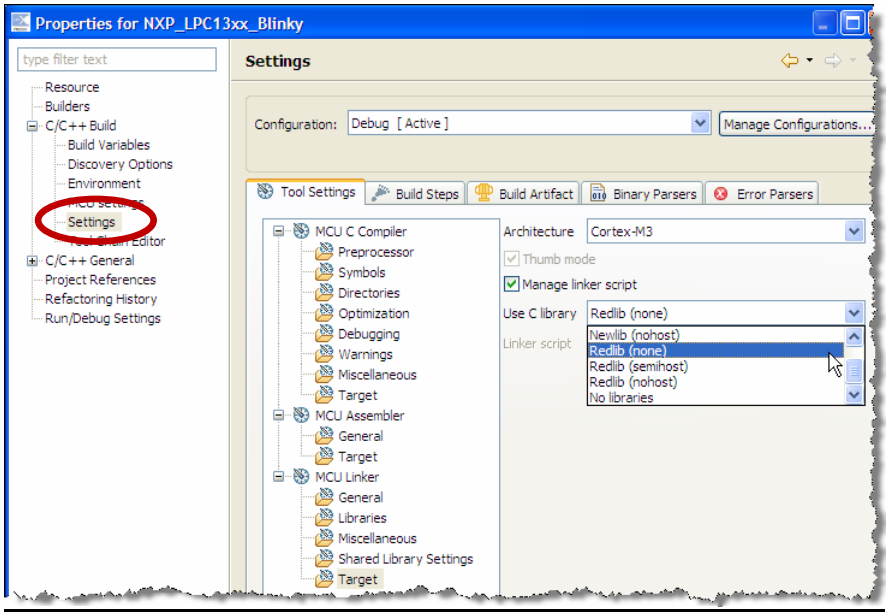


Fig 24. Reconfigure library setting

The printf implementation in Redlib is about half the size of the implementation in Newlib. A smaller printf library can be used in Redlib if floating point formatting strings are not used. To select this smaller library, define the symbol CR_INTEGER_PRINTF to the compiler (i.e. -DCR_INTEGER_PRINTF). To save even more space, avoid using printf or

any C standard library functions and select Redlib (none). Depending on your printf settings and code, this could free up 10K to 20K of flash memory.

6.4.2 Optimization

Optimization can do a lot to save flash memory. It can be configured in the same dialog as the C standard library. Choose “Optimization” under “MCU C Compiler” in the “Tool Settings” tab. Higher levels of optimization will typically result in higher performance, but may result in larger code size. It is best to use –O0 for debugging and higher levels for Release. For best code size try –Os –mword-relocations. To further reduce code, add --gc-sections to the project linker flags. This causes the linker to remove unused functions from the compiled code. --gc-sections is enabled by default in new projects created by the project wizard. If you are working with an existing project, you may need to manually add this option to your project. --gc-sections is safe to use in both Release and Debug builds. There are many optimization options available for GCC. Visit http://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html to see all of them.

6.5 Showing hidden views

A view is an on-screen representation of something in the IDE. A view can be source code, the project tree, or a debug window. If you accidentally close a view, you can open it again by going to the Window menu and choosing Show View and Other. It is a good idea to browse through the Show View window to see what is available.

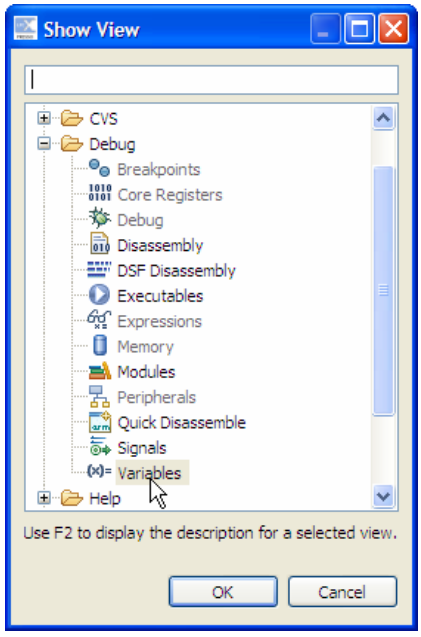


Fig 25. Show view window

This will present a dialog allowing you to pick a view and display it.

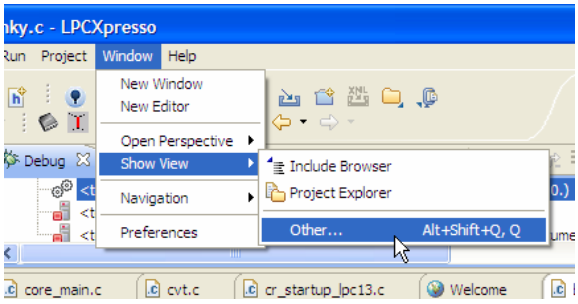


Fig 26. Pick a view to display

6.6 Creating a 'skeleton' project in a new Workspace

LPCXpresso includes several project Templates to help get started quickly.

6.6.1 Create a new Workspace

From the 'File' menu hover over 'Switch Workspace' and then select 'Other…' from the bottom of the list. You will then see the 'Workspace Launcher' dialog shown below.

Enter or browse to the new path for your workspace. We have called our new workspace 'NewWorkspace'.

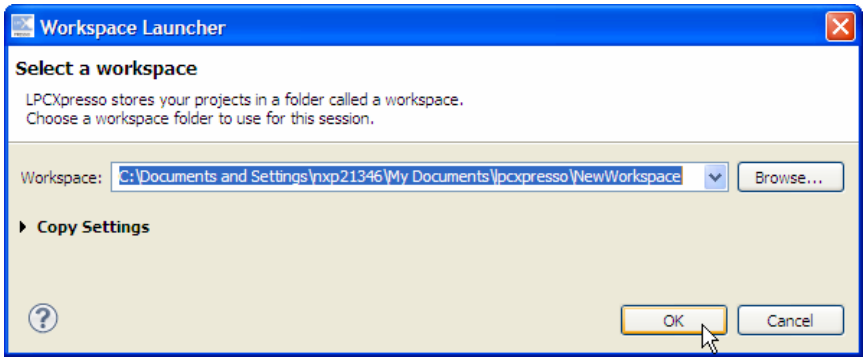


Fig 27. Workspace launcher

Then click on OK to re-open LPCXpresso with this new workspace selected.

6.6.2 Create the 'Skeleton' project

- If you are using a Cortex-based part, first, import the CMSIS header files for the chip family you are planning to work with. To do this, click "Import Example Project" again and navigate to the CMSIS<version/part>.zip. Once this project is added to your workspace, click "Build all projects (Debug)" in the Quickstart Panel.
- Click on the 'Projects and File Wizards' tab of the 'Quickstart Panel'.
- Click on 'MCU project wizards' and select "Create NXP Project' for your architecture.
- Enter a project name when the dialog appears. In this case we will use 'MyProject' then click 'Next.'

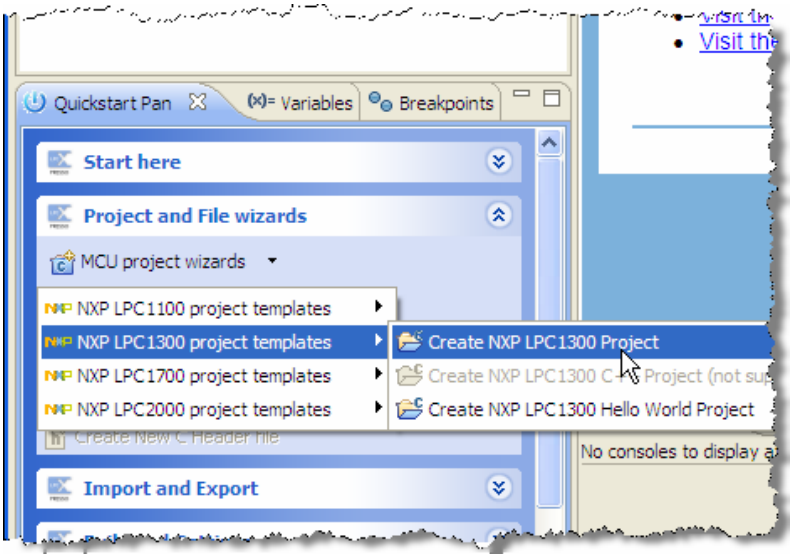


Fig 28. Enter project name

- If you are using a Cortex-based part, the wizard will ask if you would like to use CMSIS. CMSIS stands for Cortex Microcontroller Software Interface Standard. CMSIS defines a common way to access peripheral registers and to define interrupts. Please check the “Use CMSIS” box and click “Next.”

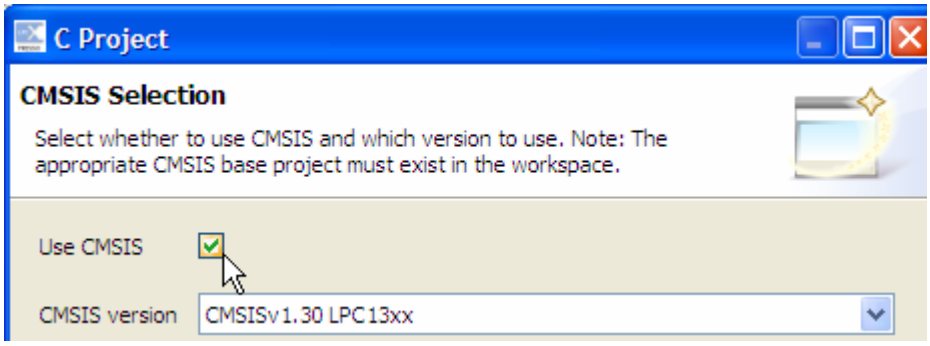


Fig 29. CMSIS selection

- The next section will ask you for “Source file information.” You may fill out your Author and Copyright text.

- Then click next and this dialog appears so that you can select what build configuration you want to create. By default both ‘Debug’ and ‘Release’ are to be created which is the most commonly used setting.

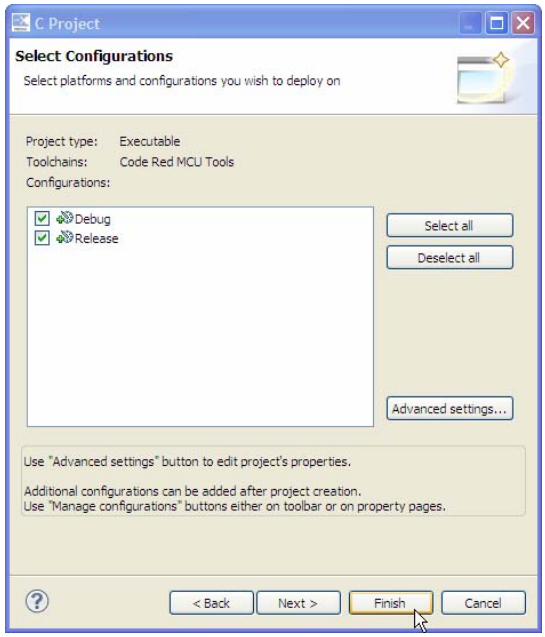


Fig 30. Select build configuration

- Finally, the ‘SelectProcessorType’ dialog appears. In here you can select your target microcontroller. Then click on finish and you have a project for your selected microcontroller. In this example, we are selecting an LPC1343. You should make sure to select the microcontroller that is on your target board. After clicking “Finish,” your sample project will be created.

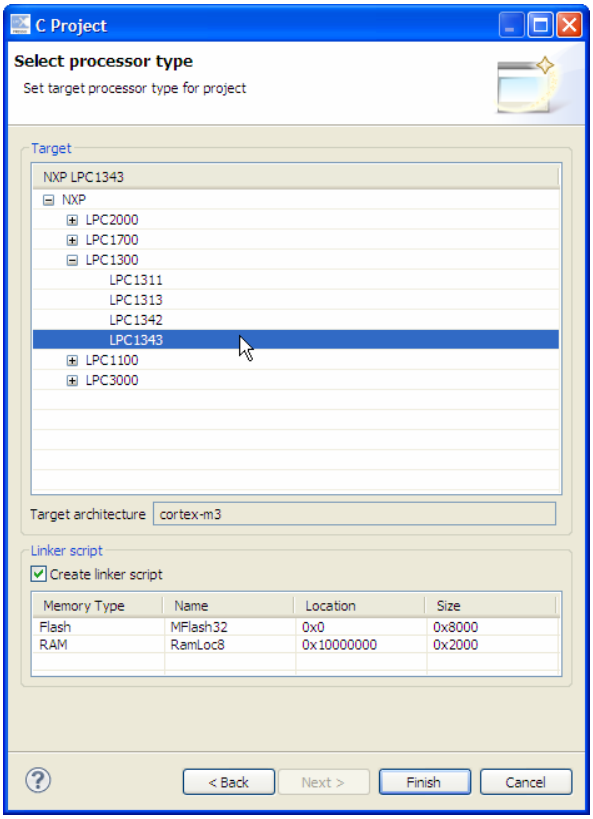


Fig 31. Select processor type

Congratulations! You have created your first project!

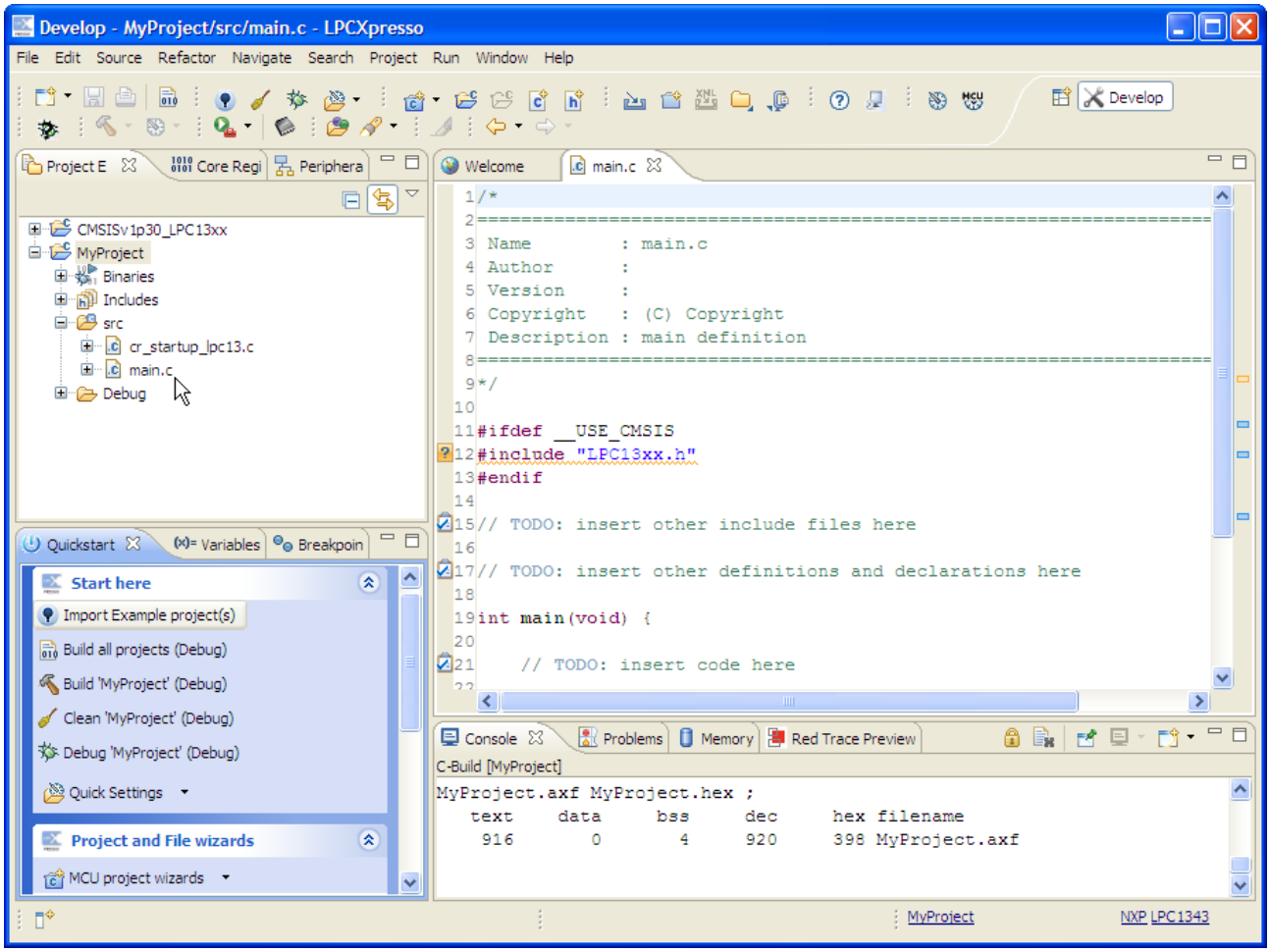


Fig 32. Project creation complete

7. Appendix

7.1 LPCXpresso target side schematics

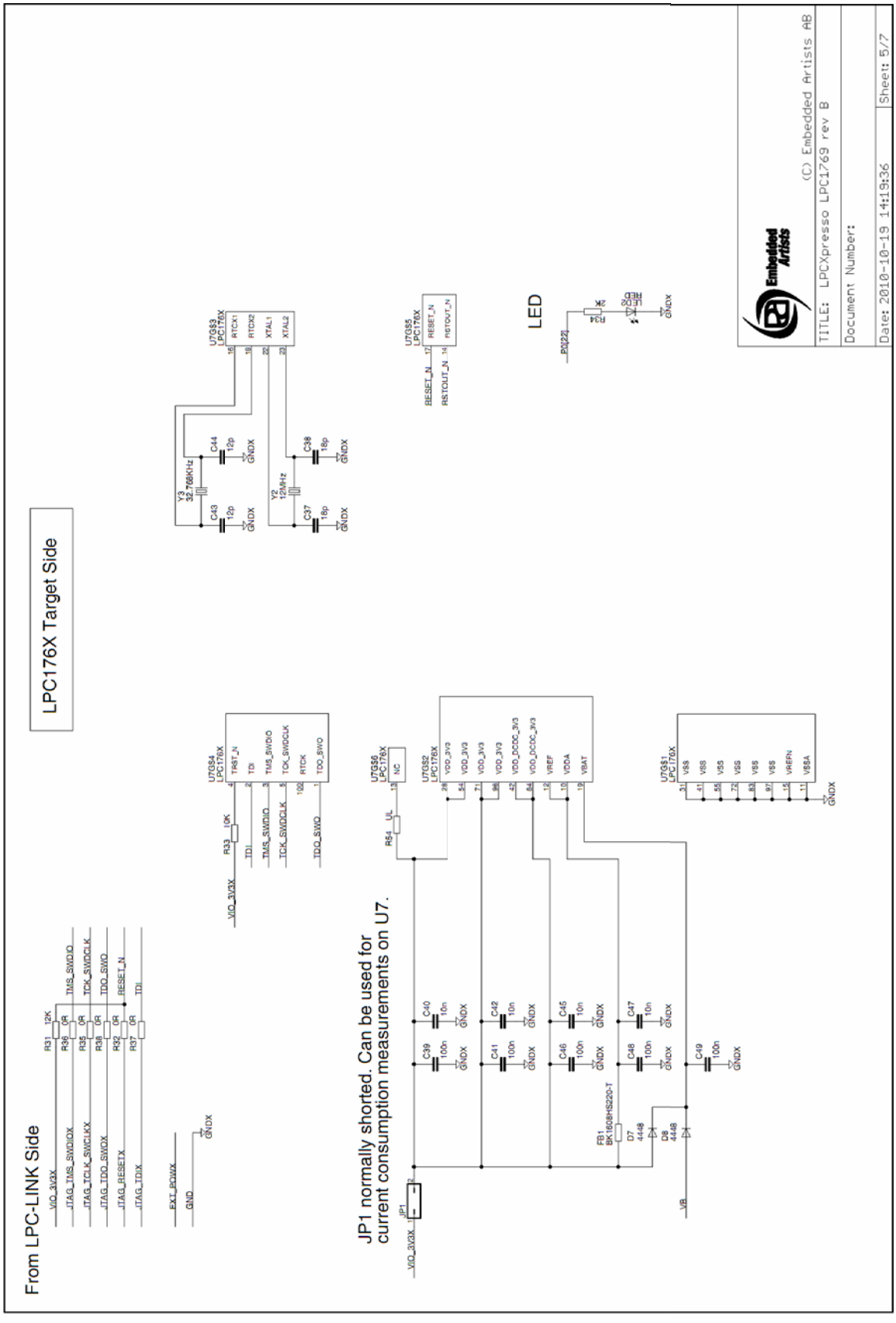


Fig 33. Schematic for the LPCXpresso LPC1769 target side (1 of 3)

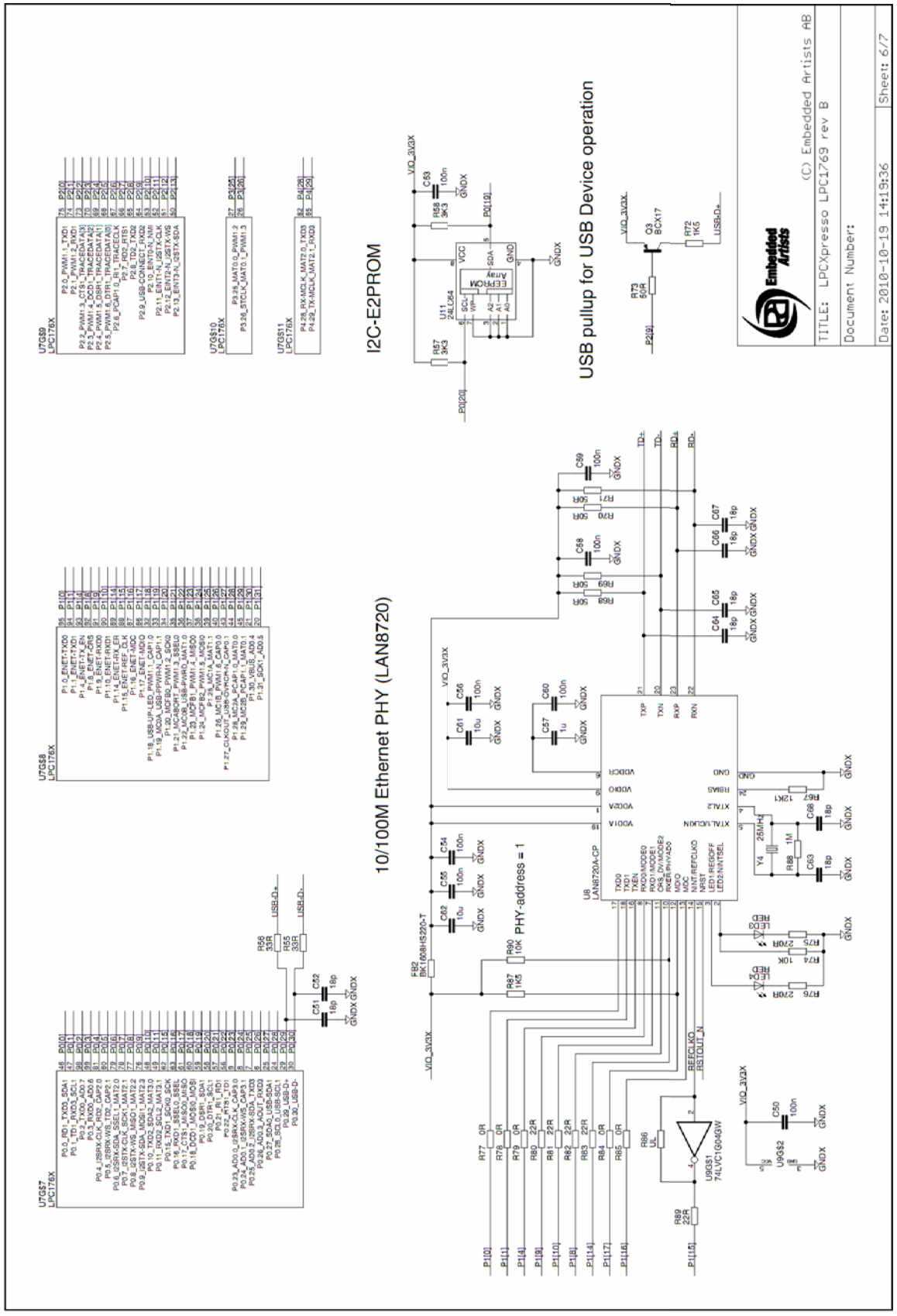


Fig 34. Schematic for the LPCXpresso LPC1769 target side (2 of 3)

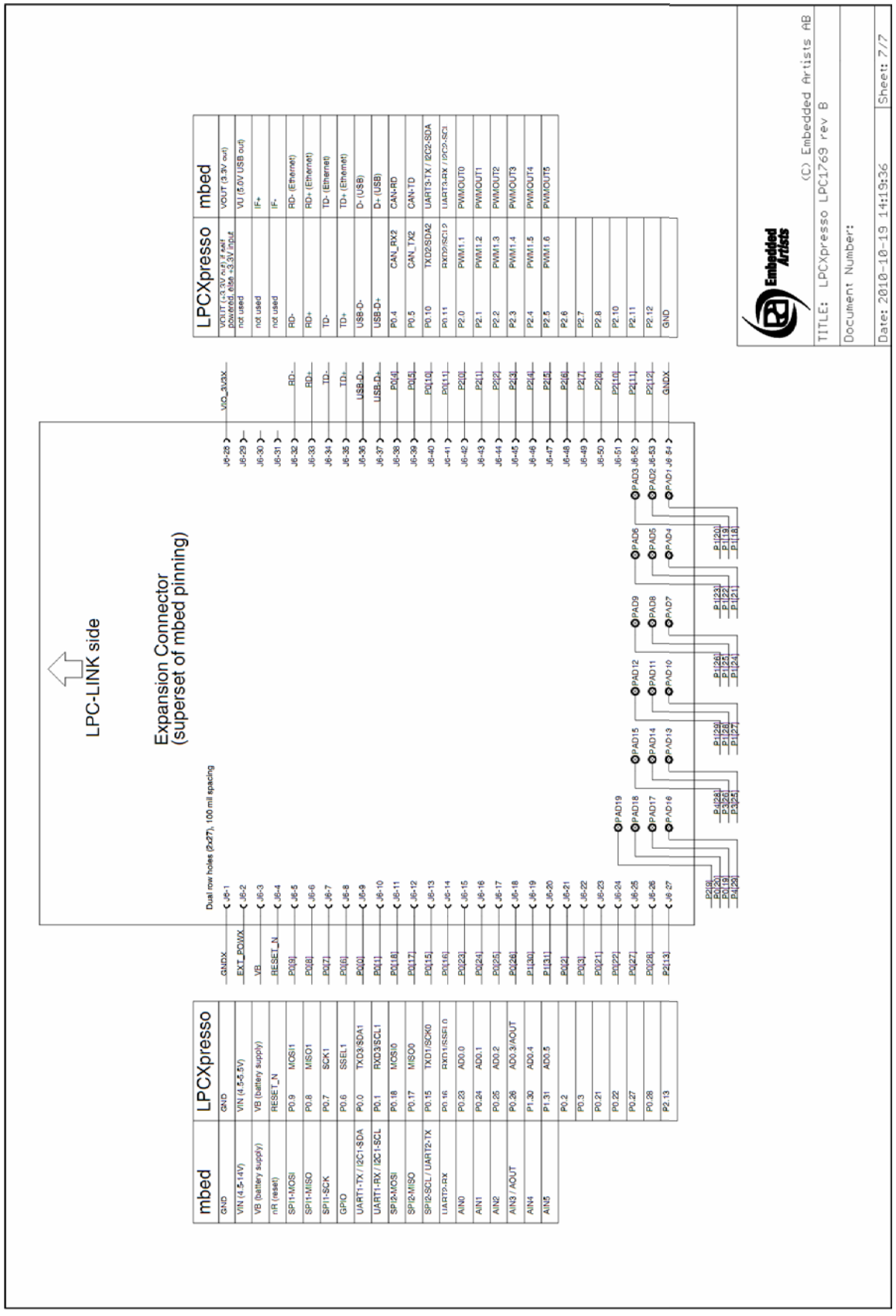
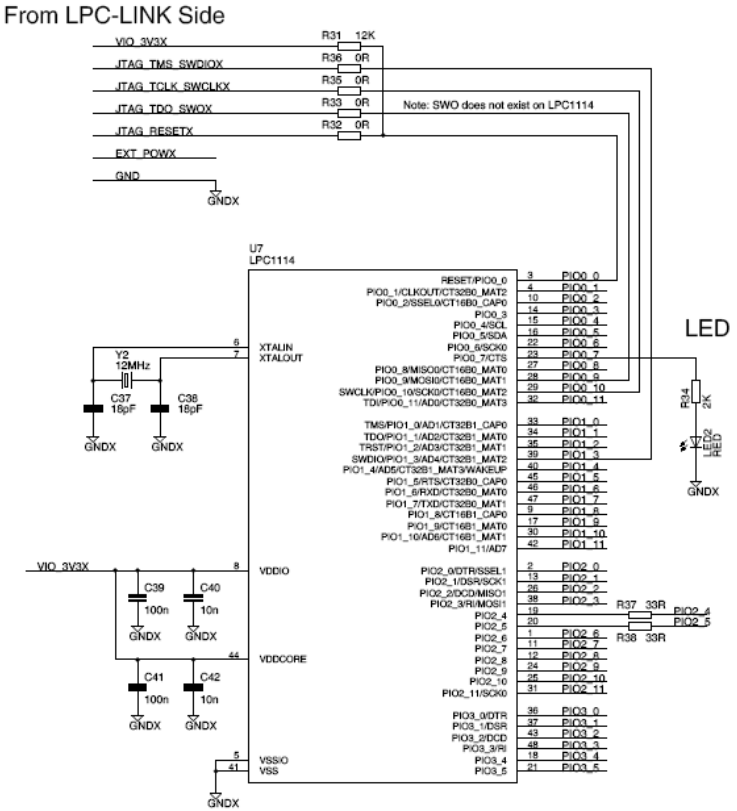


Fig 35. Schematic for the LPCXpresso LPC1769 target side (3 of 3)



mbed	LPCXpresso	
GND	GND	
VIN (4.5-14V)	VIN (4.5-5.5V)	
VB (battery supply)	not used	
nR (reset)	PIO0_0	RESET
SPI1-MOSI	PIO0_9	MOSI/SWO
SPI1-MISO	PIO0_8	MISO
SPI1-SCK	PIO2_11	SCK
GPIO	PIO0_2	SSEL
UART1-TX / I2C1-SDA	PIO1_7	TXD
UART1-RX / I2C1-SCL	PIO1_6	RXD
SPI2-MOSI	PIO0_7	
SPI2-MISO	PIO2_0	
SPI2-SCL / UART2-TX	PIO2_1	
UART2-RX	PIO2_2	
AIN0	PIO0_11	AD0
AIN1	PIO1_0	AD1
AIN2	PIO1_1	AD2
AIN3 / AOUT	PIO1_2	AD3
AIN4	PIO1_3	AD4/SWDIO
AIN5	PIO1_4	AD5
	PIO1_5	
	PIO1_8	
	PIO0_6	
	PIO0_10	SWCLK
	PIO3_0	
	PIO3_1	
	PIO3_2	

J6		J6
GNDX	J6-1	VIO_3V3X
EXT_POWX	J6-2	
	J6-3	
PIO0_0	J6-4	
PIO0_9	J6-5	
PIO0_8	J6-6	
PIO2_11	J6-7	
PIO0_2	J6-8	
PIO1_7	J6-9	
PIO1_6	J6-10	
PIO0_7	J6-11	
PIO2_0	J6-12	
PIO2_1	J6-13	
PIO2_2	J6-14	
PIO0_11	J6-15	
PIO1_0	J6-16	
PIO1_1	J6-17	
PIO1_2	J6-18	
PIO1_3	J6-19	
PIO1_4	J6-20	
PIO1_5	J6-21	
PIO1_8	J6-22	
PIO0_6	J6-23	
PIO0_10	J6-24	
PIO3_0	J6-25	
PIO3_1	J6-26	
PIO3_2	J6-27	

J6		J6
VIO_3V3X	J6-28	VOUT (+3.3V out) if self powered, else +3.3V input
	J6-29	not used
	J6-30	not used
	J6-31	not used
	J6-32	not used
	J6-33	not used
	J6-34	not used
	J6-35	not used
PIO2_4	J6-36	USB_DM Note wrong text in silkscreen
PIO2_5	J6-37	USB_DP Note wrong text in silkscreen
PIO0_1	J6-38	PIO0_1
PIO0_3	J6-39	PIO0_3
PIO0_5	J6-40	PIO0_5 I2C-SDA
PIO0_4	J6-41	PIO0_4 I2C-SCL
PIO1_9	J6-42	PIO1_9
PIO1_10	J6-43	PIO1_10
PIO1_11	J6-44	PIO1_11
PIO2_3	J6-45	PIO2_3
PIO3_4	J6-46	PIO2_4 Note wrong text in silkscreen
PIO3_5	J6-47	PIO2_5 Note wrong text in silkscreen
PIO2_6	J6-48	PIO2_6
PIO2_7	J6-49	PIO2_7
PIO2_8	J6-50	PIO2_8
PIO2_9	J6-51	PIO2_9
PIO2_10	J6-52	PIO2_10
PIO3_3	J6-53	PIO3_3
GNDX	J6-54	GND

mbed
VOUT (3.3V out)
VU (5.0V USB out)
IF+
IF-
RD- (Ethernet)
RD+ (Ethernet)
TD- (Ethernet)
TD+ (Ethernet)
D- (USB)
D+ (USB)
CAN-RD
CAN-TD
UART3-TX / I2C2-SDA
UART3-RX / I2C2-SCL
PWMOUT0
PWMOUT1
PWMOUT2
PWMOUT3
PWMOUT4
PWMOUT5

Fig 36. Schematic for the LPCXpresso LPC1114 target side[2]

2. Design and layout compatible with LPC1343 version. Therefore, PIO2_4/5 and PIO3_4/5 swap. LPC1114 does not have USB, but LPC1343 does. Therefore R37/38. LPC1114 does not have SWO, but PIO0_9 is connected (since LPC1343 has SWO there).

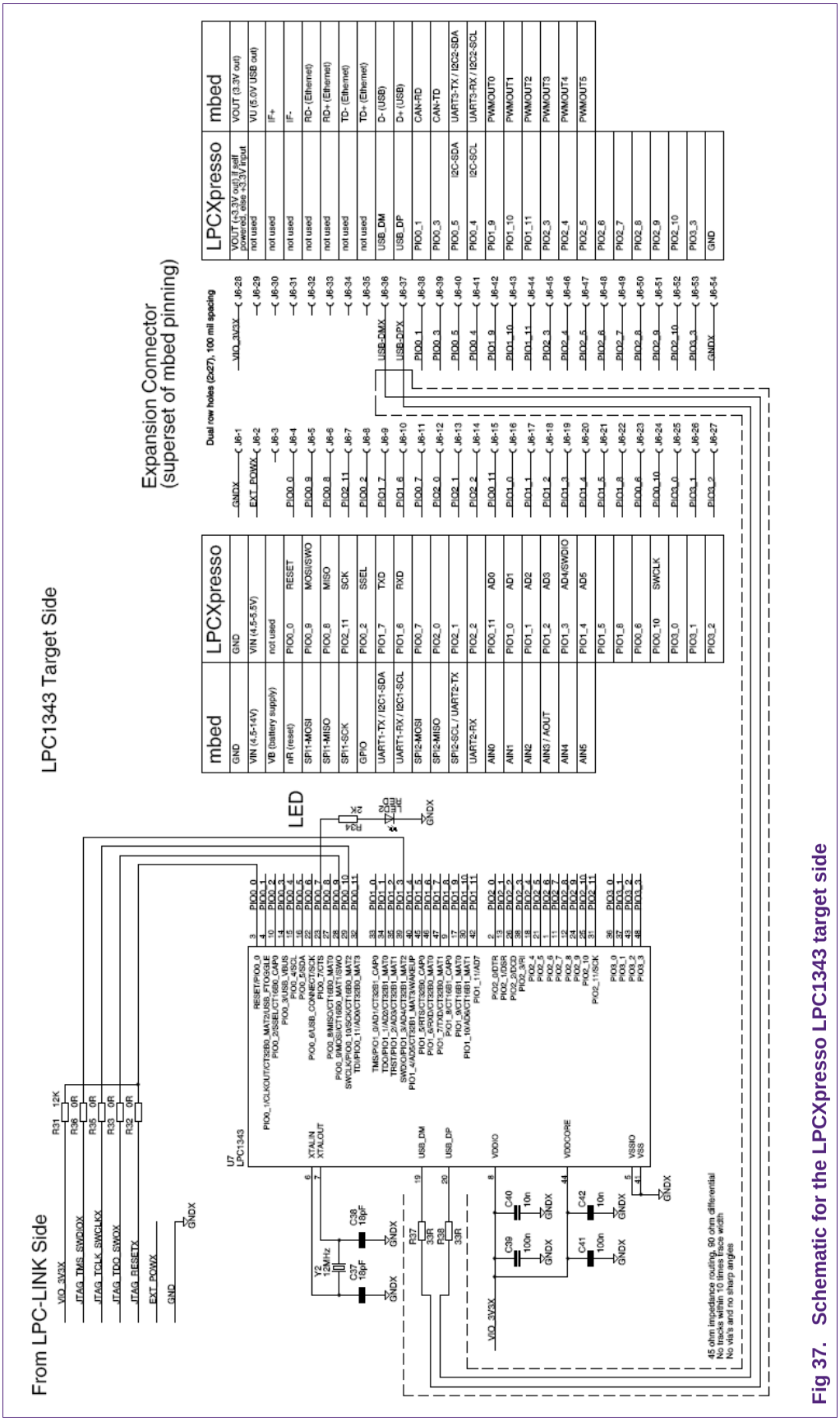


Fig 37. Schematic for the LPCXpresso LPC1343 target side

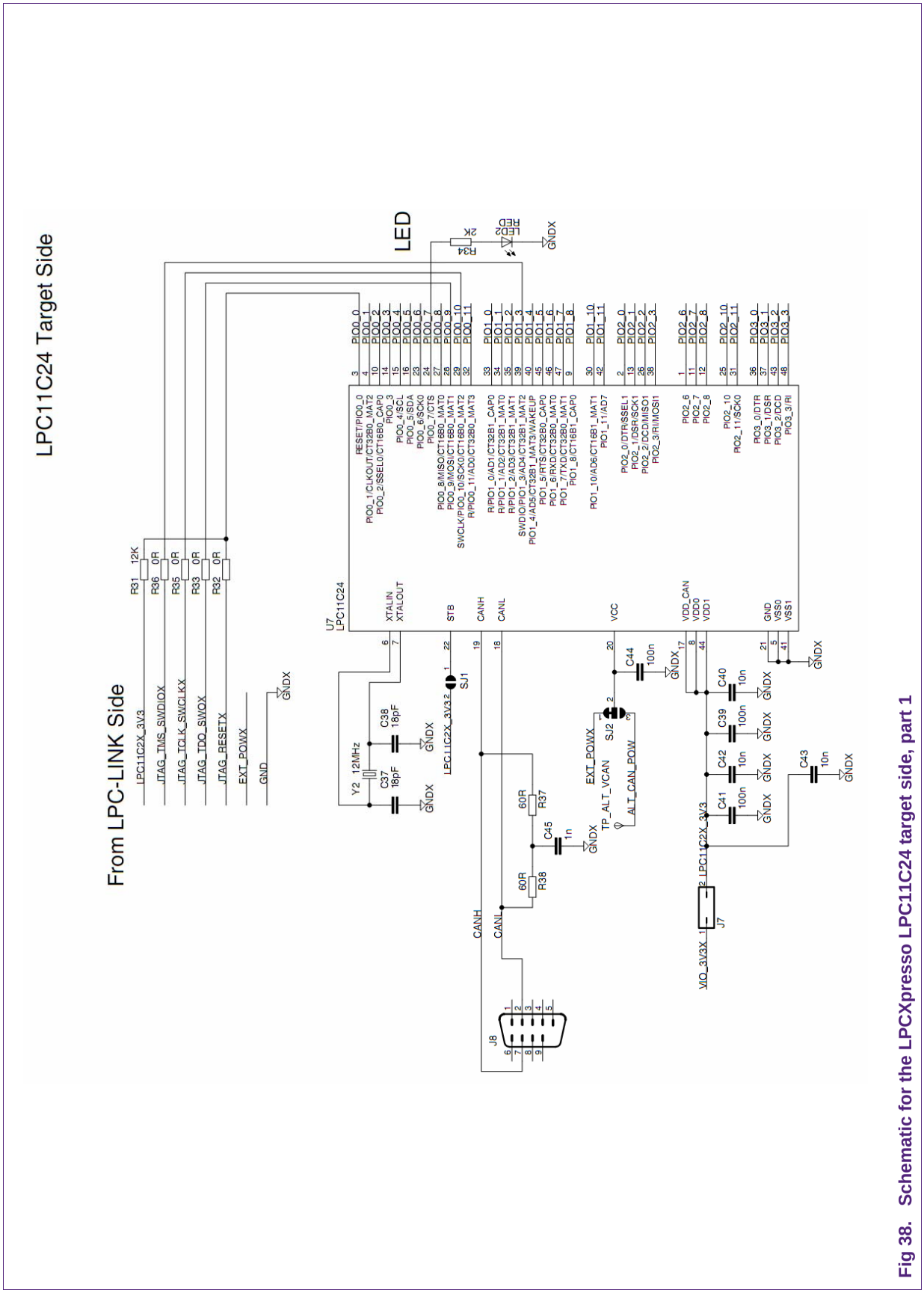


Fig 38. Schematic for the LPCXpresso LPC11C24 target side, part 1

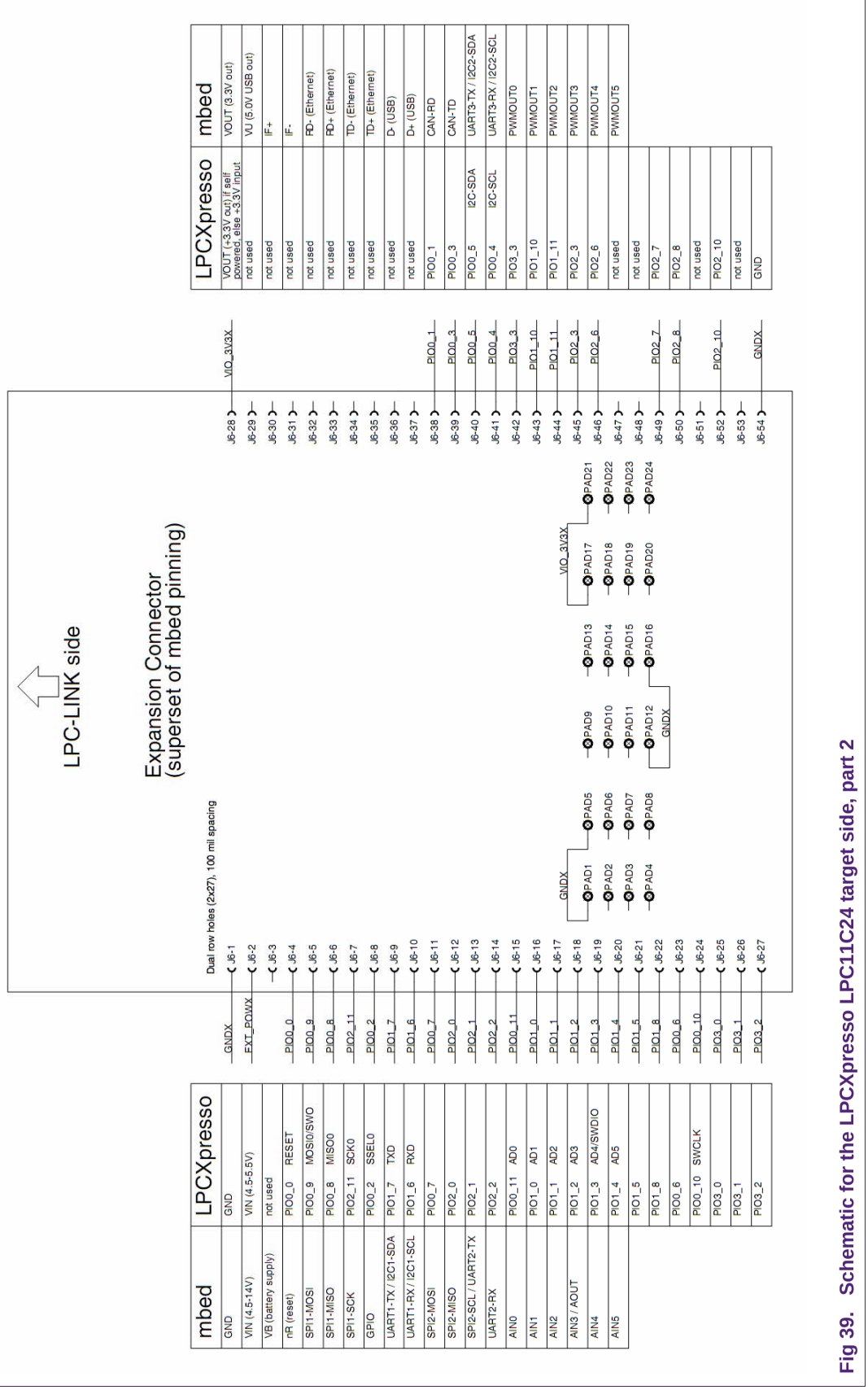


Fig 39. Schematic for the LPCXpresso LPC11C24 target side, part 2

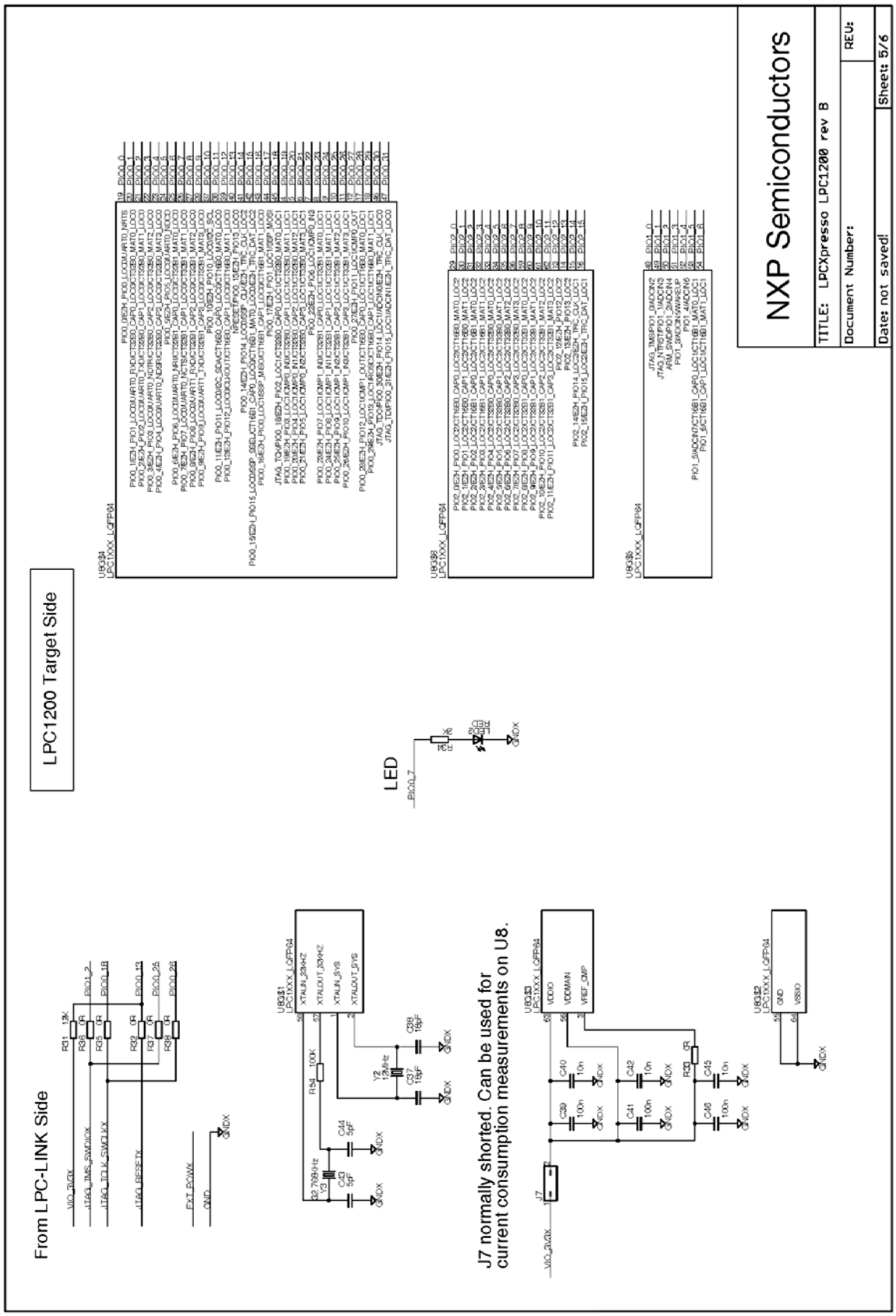


Fig 40. Schematic for the LPCXpresso LPC1200 target side, part 1

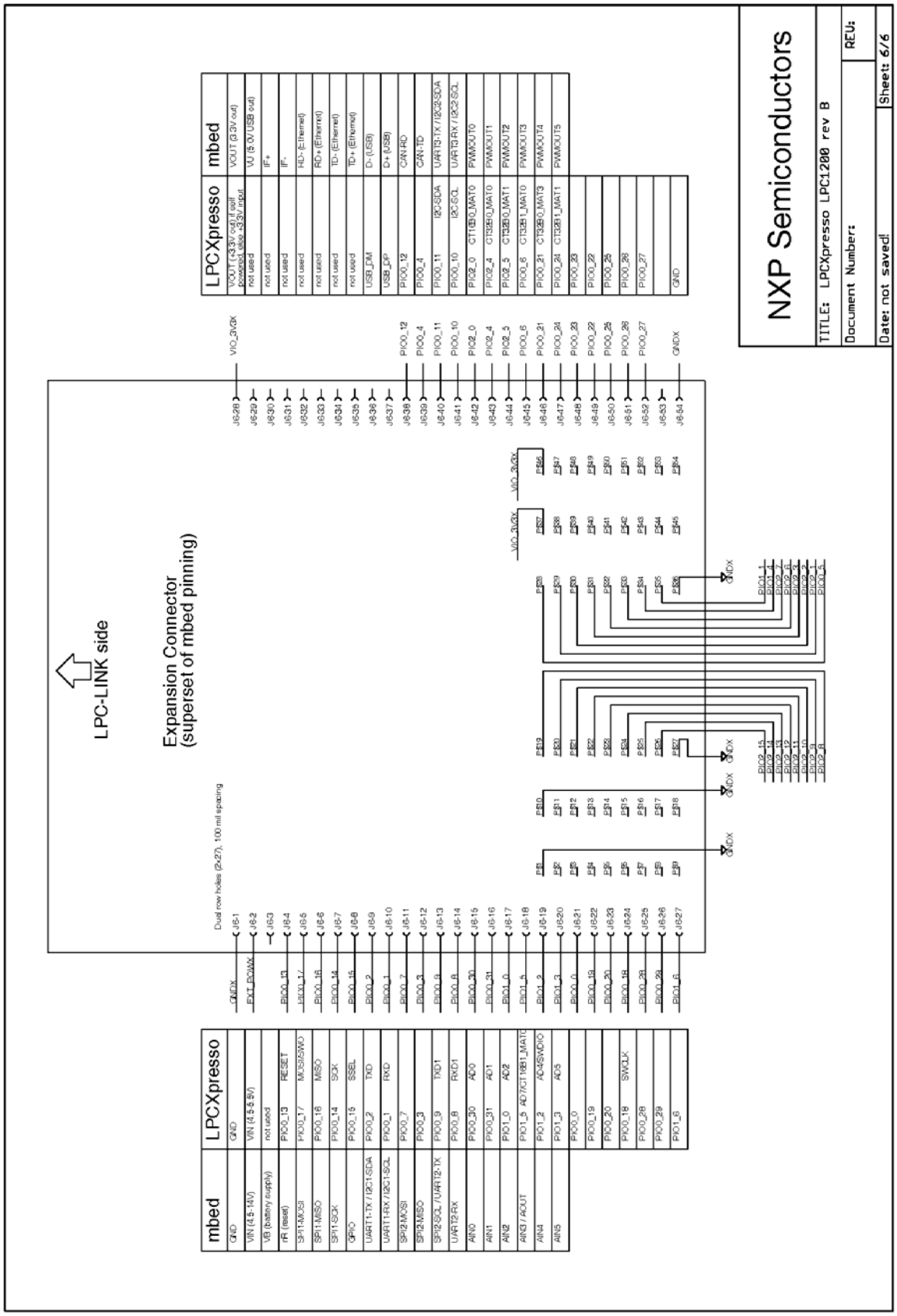


Fig 41. Schematic for the LPCXpresso LPC1200 target side, part 2

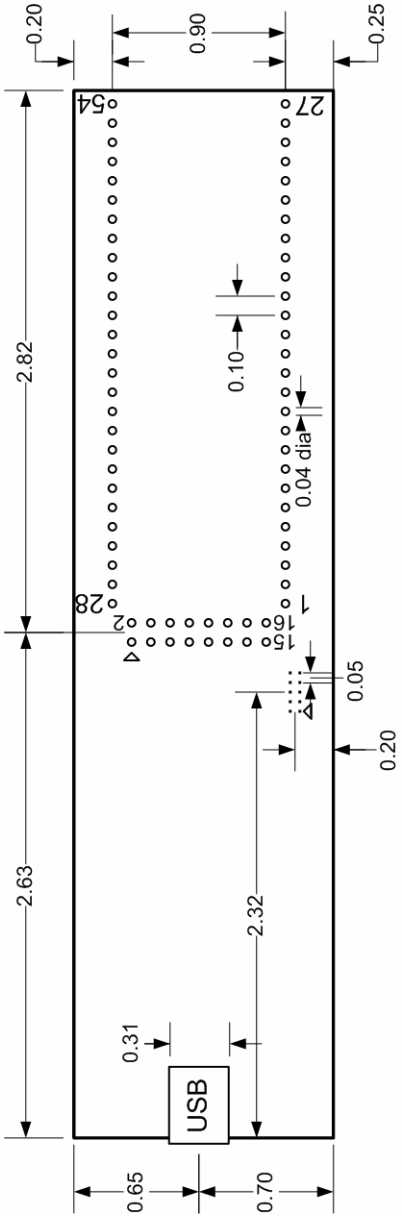


Fig 42. Dimensioned drawing of LPCXpresso LPC1227, LPC1343, LPC1114/301, LPC1114/302, LPC11C24, LPC1769, LPC1768

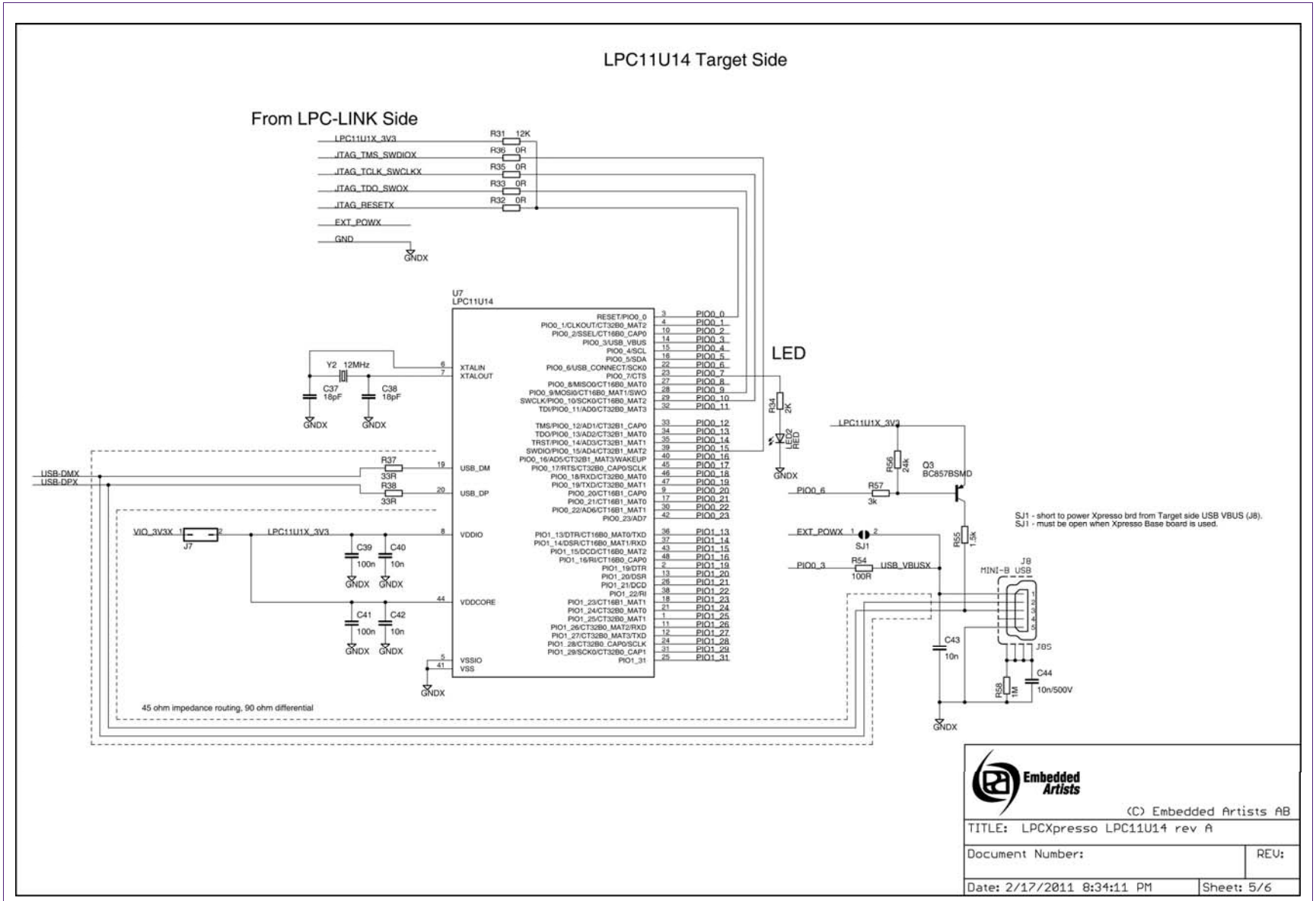


Fig 43. Schematic for the LPCXpresso LPC11U14 target side, part 1

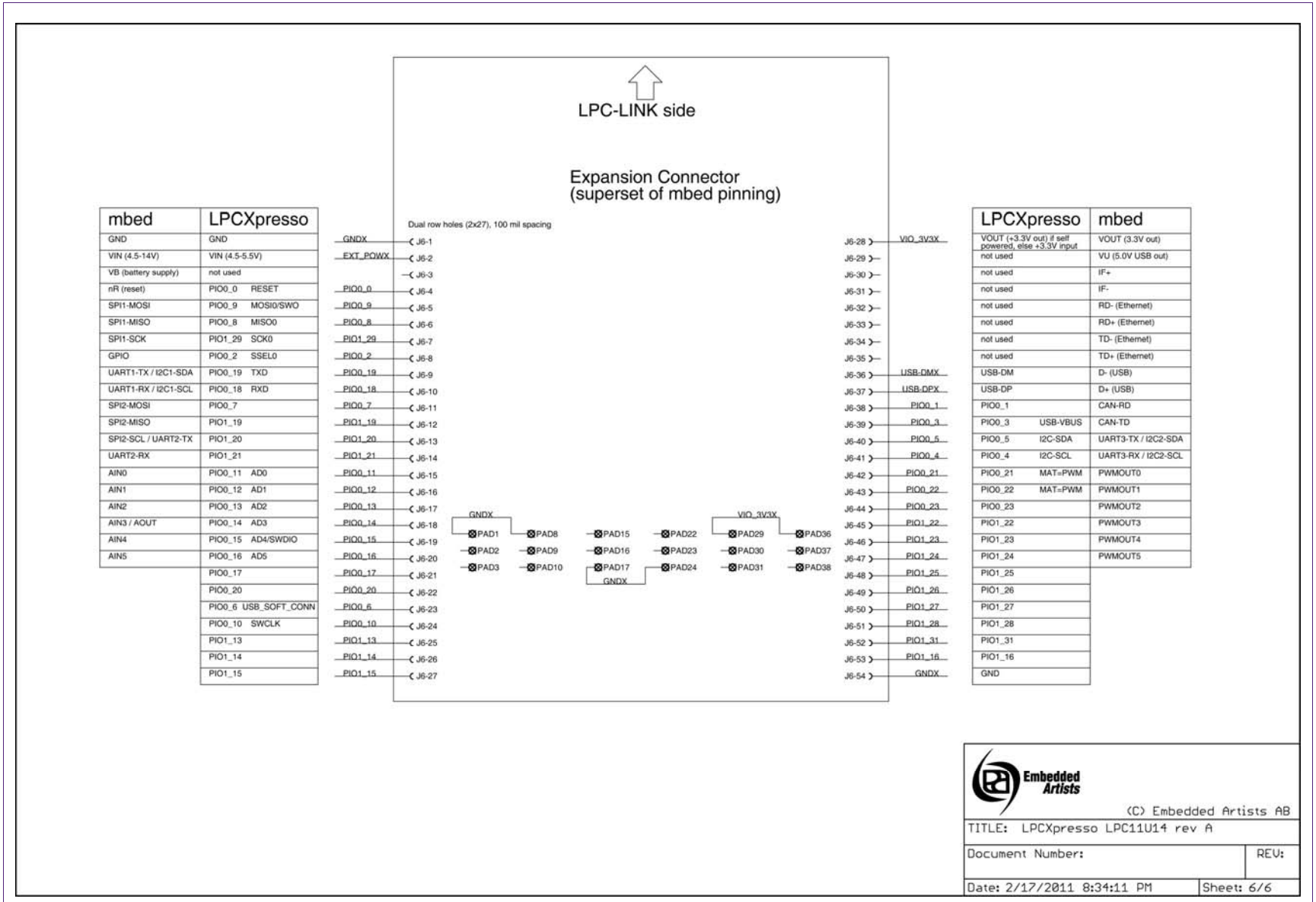


Fig 44. Schematic for the LPCXpresso LPC11U14 target side, part 2

7.2 LPCXpresso PCB pinout and dimensions

The schematics of the LPCXpresso Target and the LPC-LINK debug connector appear in Fig 36 and Fig 37. The LPCXpresso board was designed to be pin compatible with NXP mbed. LPCXpresso can be powered either through the debug mini-USB port, by 3.3V applied to the board, or by 5V applied to the USB connector. A cable for the 10-pin mini JTAG connector on the LPC-LINK debugger portion of LPCXpresso can be purchased from Digi-Key, part number FFSD-05-D-06.00-01-N.

Dimensions: LPCXpresso LPC1343's outer dimensions are 1.35x5.45 inches. It contains two rows of holes 900 mil apart. Each row has 27 connections and holes are drilled at a 100 mil pitch.

Fig 45. LPCXpresso LPC-LINK JTAG/SWO pinout

7.3 Enabling USB connectivity "to LPC1343 target"

The LPCXpresso board is simple yet flexible. Here is a way to configure it to support the development of USB devices using the LPC1343 or other USB-capable NXP microcontroller. The LPC1343 has a USB phy on-chip. To connect the microcontroller to a USB port, it is only necessary to add a USB connector and a pullup resistor.

Note: This simple connection does not implement NXP Soft-Connect to allow soft disconnection and connection to the USB bus. Because of this, the USB connection must be plugged into the PC near the time the USB peripheral is initialized, or after. If the USB port is connected when the LPC USB peripheral is not initialized, the pullup resistor will notify the PC that a USB device is present, yet the microcontroller will not respond because it has not been initialized. This will trigger windows to generate an error regarding a malfunctioning USB device. Unplug and re-plug the device to dismiss the error.

Note 2: Rather than building a cable or wiring a USB Type-A connector, you could take an existing A-B USB cable and cut off the B connector. Then the A side of the cable could be stripped and soldered onto the LPCXpresso board.

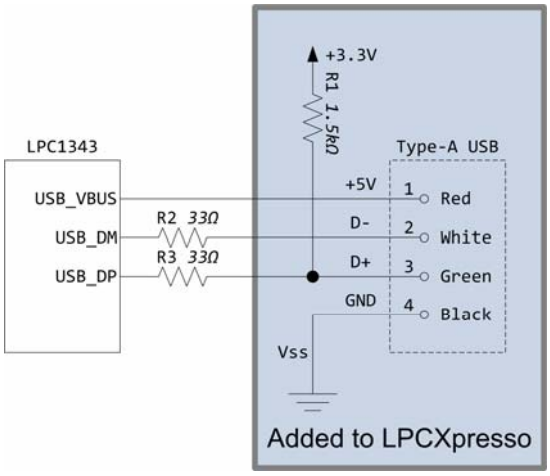


Fig 46. USB retrofit schematic

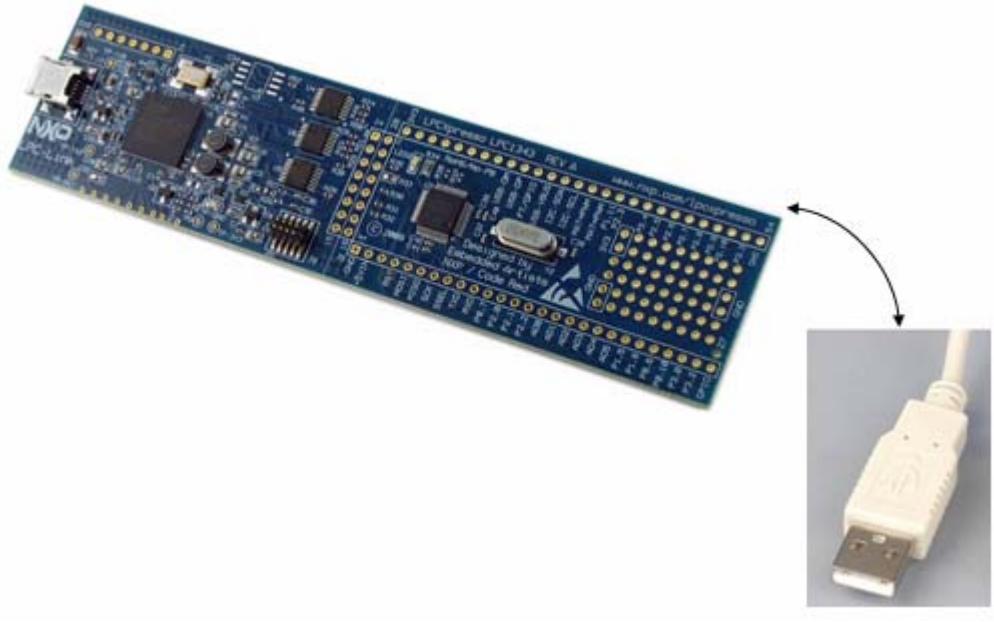


Fig 47. LPC1343 target connected to Type-A USB cable

7.4 Terminology

LPCXpresso

The Code Red Technologies IDE (Integrated Development Environment) based on Eclipse with our own extensions for embedded development.

SWD

Serial Wire Debugging (Single Wire Debugging). This is a debug connection technology available on the Cortex-M3 that allows debug through just 2-wires unlike 5 for JTAG.

ELF (Executable and Linking Format)

This is the object code file format used by our development tool chain and most microprocessor tool chains.

Workspace

LPCXpresso organizes groups of projects into a ‘Workspace’. A workspace is stored as a directory on your host PC and has subdirectories containing individual projects.

Project

An LPCXpresso project. A project contains all of the .c and .h files to build a single microcontroller flash image.

Perspective

In LPCXpresso, a perspective is a particular collection of ‘Views’ that are grouped together to be suitable for a particular use. For example the ‘C/C++ programming’ perspective and the ‘Debug’ perspective.

View

A ‘View’ is a window in LPCXpresso that shows a particular file or activity. A view could be of a C source code file or something live such as a disassembly window or register dump. A ‘Perspective’ is the layout of many ‘Views’.

Semi-hosting

The ability to use IO on your debugger host system for your target embedded system. For example a ‘printf’ will appear in the console window of the debugger.

Debug Target

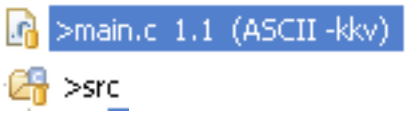
The system being debugged. LPCXpresso includes a target microcontroller on-board, but can also be connected to external targets.

Redlib™

The optimized Code Red Technologies C runtime library (non-GNU). LPCXpresso includes both Redlib and Newlib libraries.

7.5 File Icons and their meaning

Table 1. File icons and their meaning

Icon	Meaning
	C language source file.
	C language header file.
	Project Makefile. Note that LPCXpresso may automatically generate these.
	C language source file that has been excluded from the project build. – Hollowed out character in icon.
	General text file. Often used for files such as linker script files which end with the extension '.ld'.
	The blue double arrows at the top of the icon denote that this particular source file has different project build properties than the rest of the project.
	The orange/gold 'storage drive' denotes that this file is connected to a CVS repository.
	The right arrow denotes that the file or directory has been modified locally and may need committing to the CVS repository, i.e., the local copy is more up to date than the repository.
	This is an executable file.
	Directory containing executables.
	C Project Directory (Project Explorer View)
	C Project Directory linked to a CVS repository. (Project Explorer View)

7.6 References

[1] NXP LPCXpresso http://www.nxp.com/lpcxpresso

[2] NXP LPCZone http://www.nxp.com/lpczone

[3] NXP Microcontrollers http://www.nxp.com/microcontrollers

[4] Code Red Technologies Wiki http://lpcxpresso.code-red-tech.com/LPCXpresso/softwareknowledgebase

[5] Code Red Technologies LPCXpresso page http://www.code-red-tech.com/lpcxpresso

[6] Embedded Artists AB http://www.embeddedartists.com

[7] C, A Reference Manual, Fifth Edition. Samuel P. Harbison III, Guy L. Steele Jr. Prentice Hall.

[8] The Definitive Guide to the ARM Cortex-M3, Joseph Yiu. Newnes

[9] ARM Cortex-M3 Technical Reference Manual. Revision: r2p0.(ARM DDI 0337G). http://infocenter.arm.com/help/index.jsp

[10] ARM Cortex-M0 Technical Reference Manual. Revision: r1p0.(ARM DDI 0413D). http://infocenter.arm.com/help/index.jsp

[11] ARMv7-M Architecture Reference Manual (ARM DDI 0403) http://infocenter.arm.com/help/index.jsp

[12] ARMv6-M Architecture Reference Manual (Cortex-M0/LPC11) http://infocenter.arm.com/help/index.jsp

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 national authorities.

8.3 Trademarks

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

9. List of figures

10. List of tables

11. Contents

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

Date of release: 7 April 2011
Document identifier: LPCXpresso