LPCXpresso

Getting started with NXP LPCXpresso

Rev. 11 — 14 June 2011 User guide

Document information

Info	Content
Keywords	LPCXpresso, LPC1100, LPC1200, LPC1300, LPC1700, LPC1800, LPC2000, LPC2900, LPC3000, LPC3100, LPC3200, LPC4300
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
11	20110614	• Updates for LPCXpresso 4- new screen shots • Added 1.1 V4 new features section • Updated comprehensive supported parts list • Updated create new project process
10	20110407	• Added Fig 45 and Fig 46.
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 42 and Fig 43
8	20110211	• Updated Section 3.2 • Added Fig 39, Fig 40, Fig 41, Fig 44
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 47
3	20100315	• Updated Fig 38 • Added Fig 39
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 and board part numbers on the LPCXpresso platform:

- **LPC1100**: LPC1102 LPC1111/101 LPC1111/102 LPC1111/201 LPC1111/202 LPC1112/101 LPC1112/102 LPC1112/201 LPC1112/202 LPC1113/201 LPC1113/202 LPC1113/301 LPC1113/302 LPC1114/201 LPC1114/202 LPC1114/301 LPC1114/302 LPC11C12/301 LPC11C14/301 LPC11C22/301 LPC11C24/301 LPC11U12/201 LPC11U13/201 LPC11U14/201
  - OM11049: LPC1114/302
  - OM13014: LPC11U14
  - OM13012: LPC11C24
- **LPC1200**: LPC1224/201 LPC1224/221 LPC1225/301 LPC1225/321 LPC1226/301 LPC1227/301
  - OM13008: LPC1227
- **LPC1300**: LPC1311 LPC1311/01 LPC1313 LPC1313/01 LPC1342 LPC1343
  - OM11048: LPC1343
- **LPC1700**: LPC1751 LPC1752 LPC1754 LPC1756 LPC1758 LPC1759 LPC1763 LPC1764 LPC1765 LPC1766 LPC1767 LPC1768 LPC1769 LPC1774 LPC1776 LPC1777 LPC1778 LPC1785 LPC1786 LPC1787 LPC1788
  - OM13000: LPC1769
- **LPC1800**: Coming soon
- **LPC2000**: LPC2109 LPC2109/01 LPC2134 LPC2142 LPC2362 LPC2929
- **LPC3000**: LPC3130 LPC3250
- **LPC4000**: Coming soon

LPCXpresso base board products:

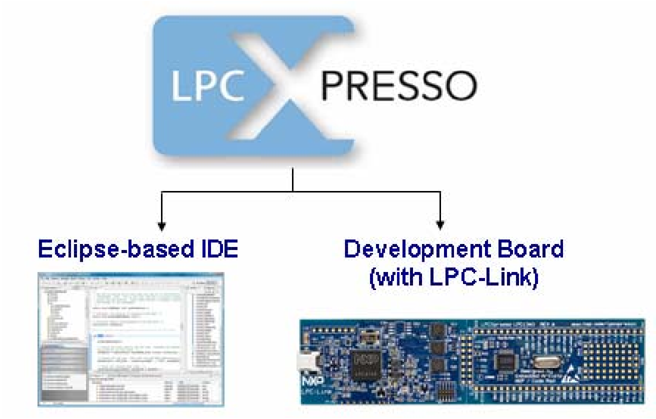
- OM11083: Embedded Artists Base Board for LPCXpresso and mbed
- OM13009: Embedded Artists Motor Control Kit for LPCXpresso
- OM13016: NGX mbed-LPCXpresso baseboard

1.1 LPCXpresso 4 new features

- Support for CMSIS v2.0 included
- Based on eclipse Helios and gcc 4.5.1
- New part support

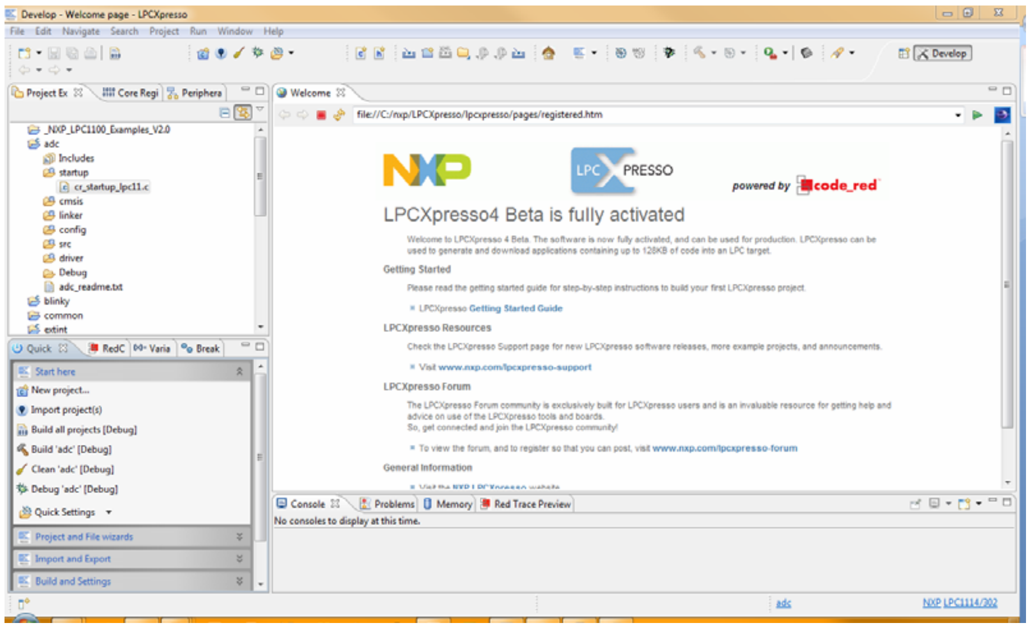
For more information on LPCXpresso 4 new features visit:

http://support.code-red-tech.com/CodeRedWiki/NewInVersion4



1.2 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.



LPCXpresso4 Beta is fully activated

Welcome to LPCXpresso 4 Beta. The software is now fully activated, and can be used for production. LPCXpresso can be used to generate and download applications containing up to 128KB of code into an LPC target.

Getting Started

Please read the getting started guide for step-by-step instructions to build your first LPCXpresso project.

- LPCXpresso Getting Started Guide

LPCXpresso Resources

Check the LPCXpresso Support page for new LPCXpresso software releases, more example projects, and announcements.

- Visit www.nxp.com/lpcxpresso-support

LPCXpresso Forum

The LPCXpresso Forum community is exclusively built for LPCXpresso users and is an invaluable resource for getting help and advice on use of the LPCXpresso tools and boards.
So, get connected and join the LPCXpresso community!

- To view the forum, and to register so that you can post, visit www.nxp.com/lpcxpresso-forum

General Information

No consoles to display at this time.

Fig 1. LPCXpresso IDE

1.3 LPCXpresso development board

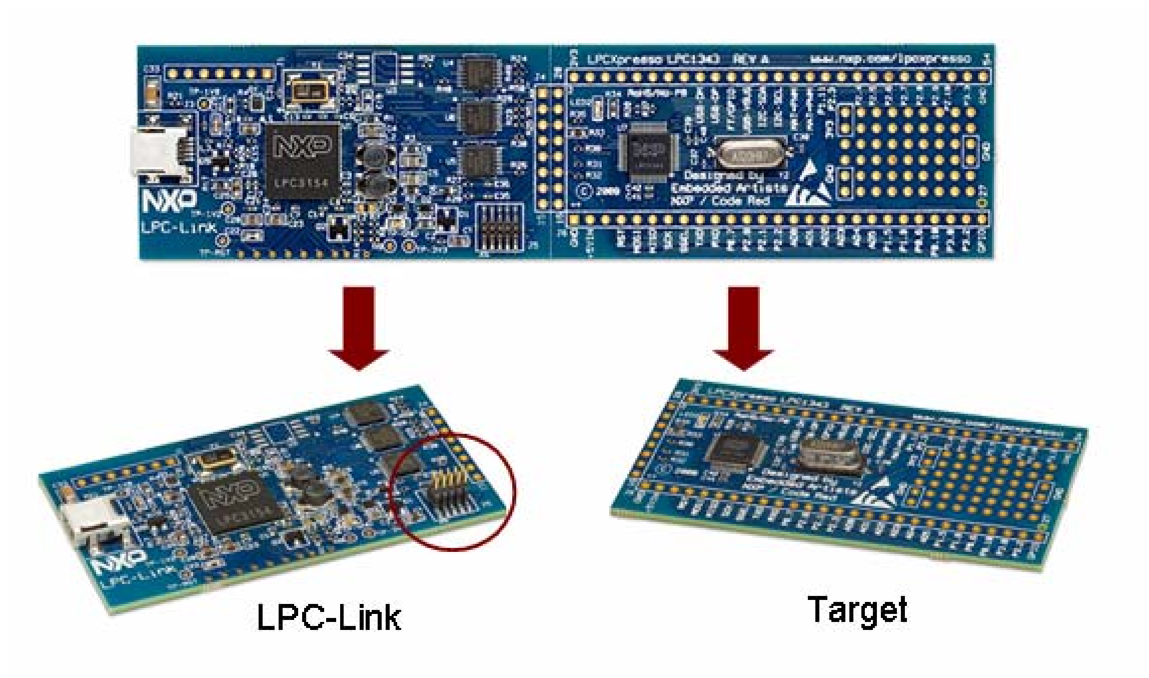


Fig 2. LPCXpresso development board

1.4 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.

1.5 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

1.6 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

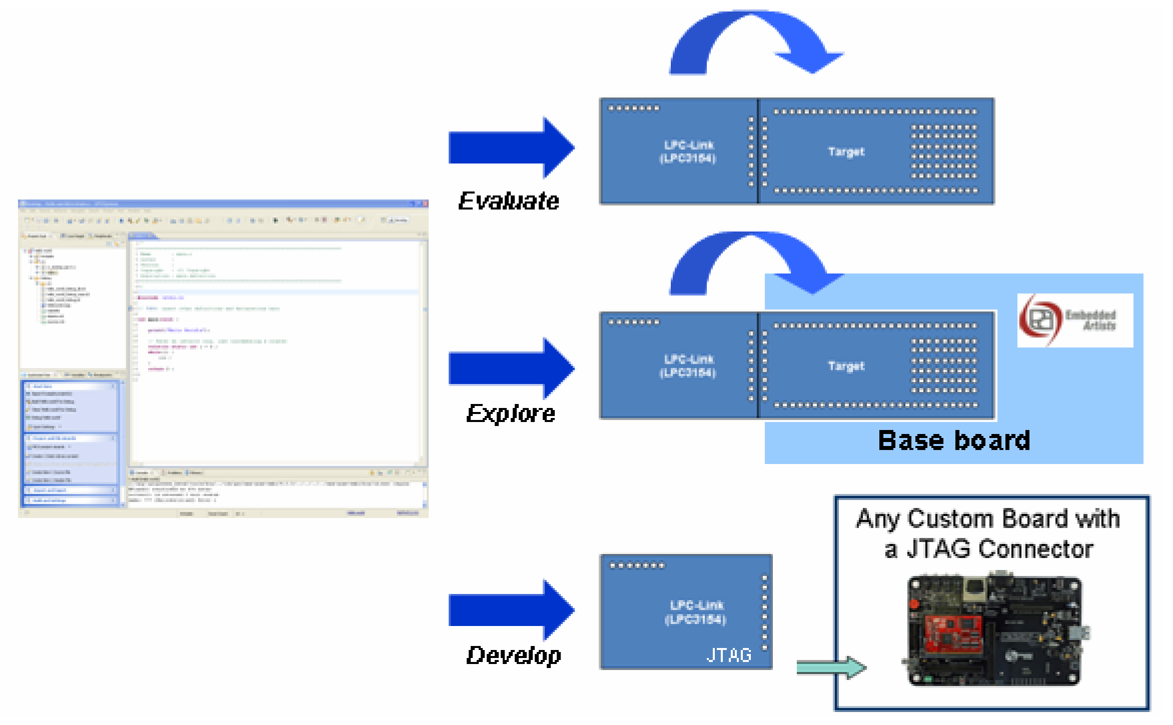


Fig 3. Product development stages

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

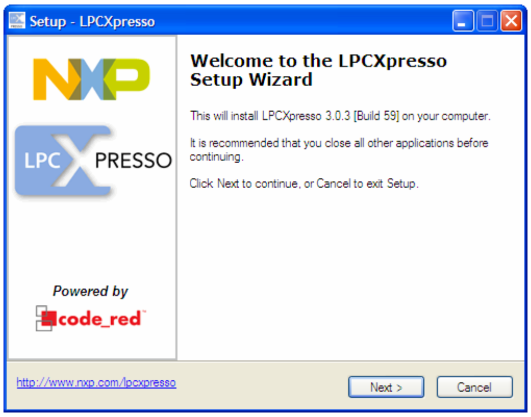
3.1 System requirements

Operating System	Microsoft® Windows - XP 32-bit (SP2 or greater) Microsoft® Windows - Vista 32-bit or 64-bit Microsoft® Windows - Windows 7 32-bit or 64-bit Linux - Ubuntu 9, 10, and 11 Linux - Fedora 12 and 13
System RAM	512 MB minimum (1 GB recommended)
Hard Disk	300+ MB of available space.
Screen/Display Adaptor	1024x768 minimum recommended
Internet Connection	High-speed internet is recommended to download and register the software

Note: LPCXpresso may install and run on other Linux distributions. However, only the distributions listed above have been tested. 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.



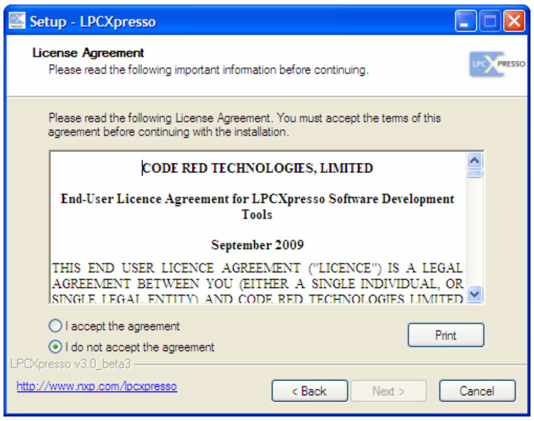
Welcome to the LPCXpresso Setup Wizard

This will install LPCXpresso 3.0.3 [Build 59] on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Fig 4. Setup wizard


Setup - LPCXpresso

License Agreement

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

CODE RED TECHNOLOGIES, LIMITED

End-User Licence Agreement for LPCXpresso Software Development Tools

September 2009

THIS END USER LICENCE AGREEMENT ("LICENCE") IS A LEGAL AGREEMENT BETWEEN YOU (EITHER A SINGLE INDIVIDUAL, OR SINGLE LEGAL ENTITY) AND CODE RED TECHNOLOGIES LIMITED

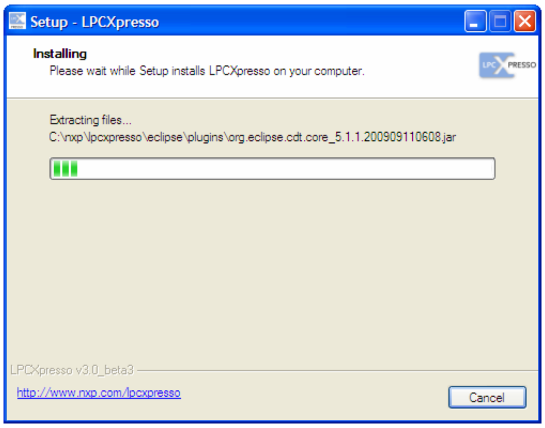
I accept the agreement [✓ I do not accept the agreement]

LPCXpresso v3.0_beta3

http://www.nxp.com/lpcxpresso

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!


Setup - LPCXpresso

Installing

Please wait while Setup installs LPCXpresso on your computer.

Extracting files...

C:\nxp\lpcxpresso\eclipse\plugins\org.eclipse.cdt.core_5.1.1.200909110608.jar

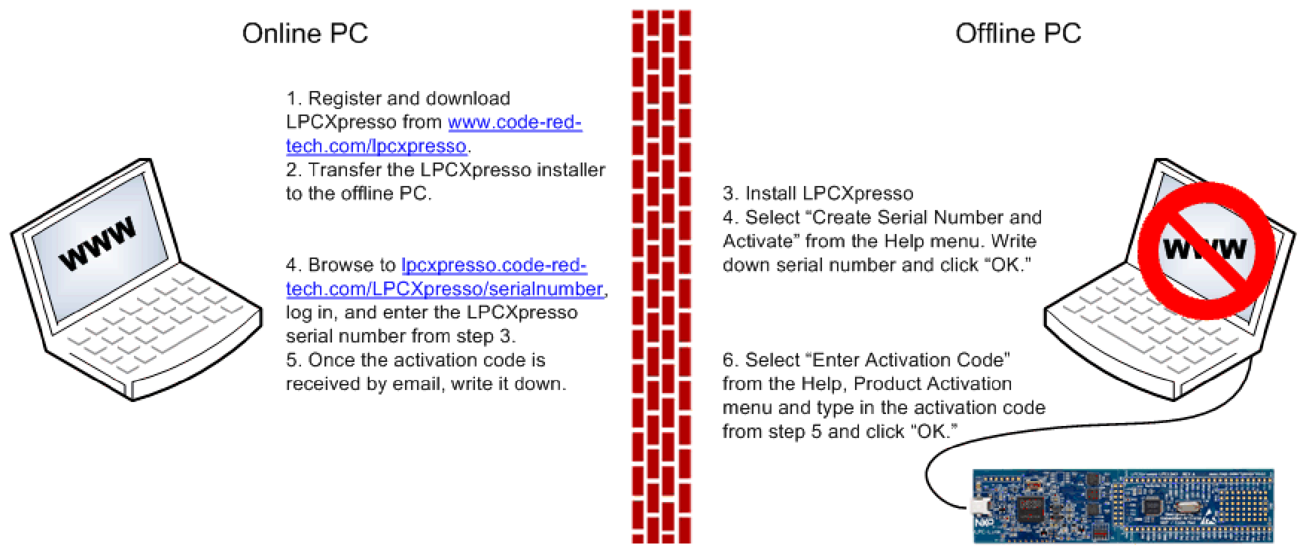
LPCXpresso v3.0_beta3

http://www.nxp.com/lpcxpresso

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.



1. Register and download LPCXpresso from www.code-red-tech.com/lpcxpresso.
2. Transfer the LPCXpresso installer to the offline PC.

3. Install LPCXpresso
4. Select “Create Serial Number and Activate” from the Help menu. Write down serial number and click “OK.”

4. Browse to lpcxpresso.code-red-tech.com/LPCXpresso/serialnumber, log in, and enter the LPCXpresso serial number from step 3.
5. Once the activation code is received by email, write it down.

6. Select “Enter Activation Code” from the Help, Product Activation menu and type in the activation code from step 5 and click “OK.”

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://support.code-red-tech.com/CodeRedWiki.

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.

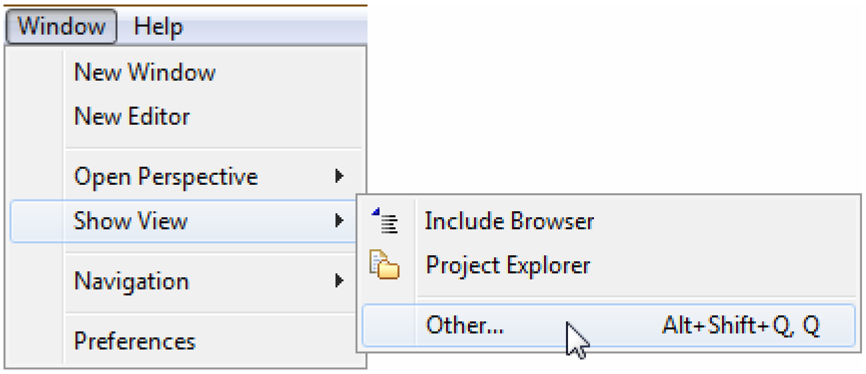
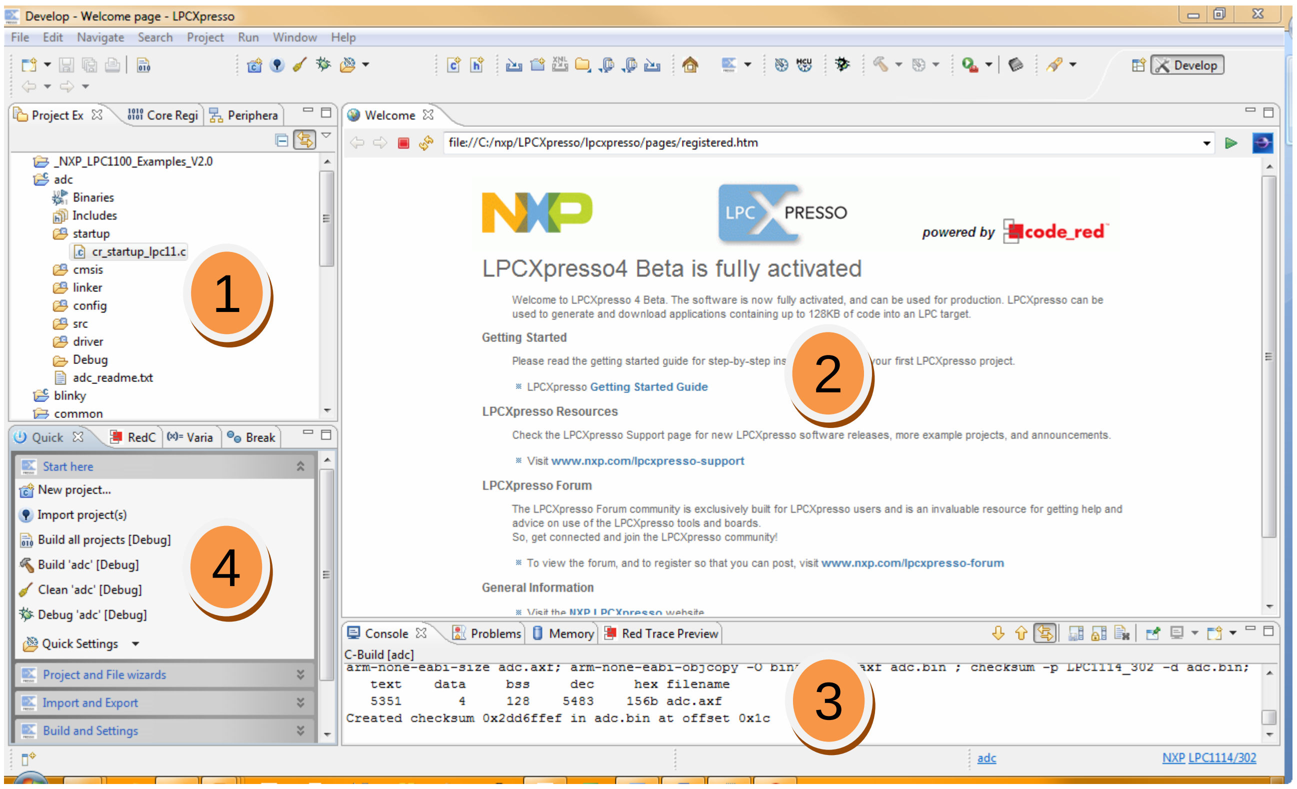


Fig 8. Show view/other menu

4.1.1 Single perspective (code development)



LPCXpresso4 Beta is fully activated

Welcome to LPCXpresso 4 Beta. The software is now fully activated, and can be used for production. LPCXpresso can be used to generate and download applications containing up to 128KB of code into an LPC target.

Getting Started

Please read the getting started guide for step-by-step ins[illegible]your first LPCXpresso project.

- LPCXpresso Getting Started Guide

LPCXpresso Resources

Check the LPCXpresso Support page for new LPCXpresso software releases, more example projects, and announcements.

- Visit www.nxp.com/lpcxpresso-support

LPCXpresso Forum

The LPCXpresso Forum community is exclusively built for LPCXpresso users and is an invaluable resource for getting help and advice on use of the LPCXpresso tools and boards.
So, get connected and join the LPCXpresso community!

- To view the forum, and to register so that you can post, visit www.nxp.com/lpcxpresso-forum

General Information

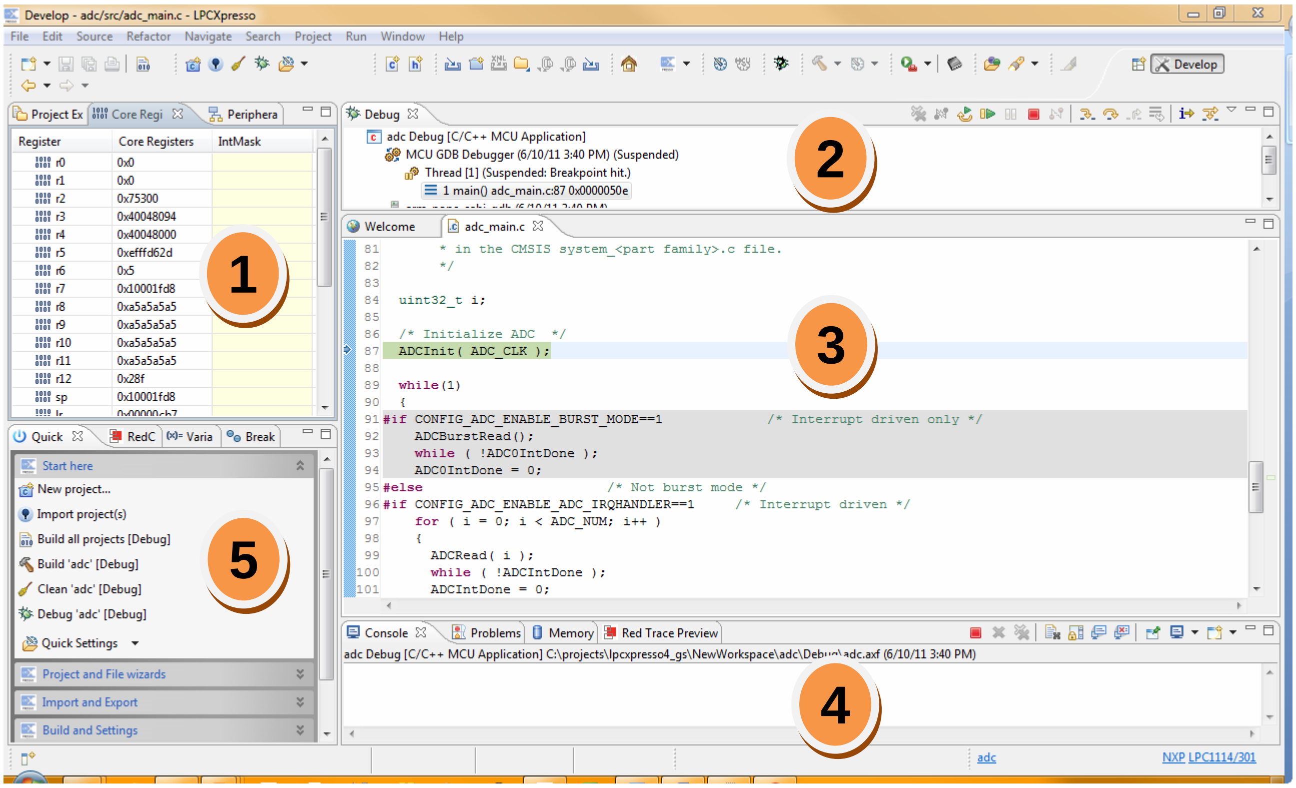
- Visit the NXP LPCXpresso website

```
arm-none-eabi-size adc.axf; arm-none-eabi-objcopy -O bin[illegible]axf adc.bin ; checksum -p LPC1114_302 -d adc.bin;
   text    data     bss     dec     hex filename
   5351       4     128    5483    156b adc.axf
Created checksum 0x2dd6ffef in adc.bin at offset 0x1c
```

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)



Register	Core Registers	IntMask
r0	0x0	
r1	0x0	
r2	0x75300	
r3	0x40048094	
r4	0x40048000	
r5	0xefffd62d	
r6	0x5	
r7	0x10001fd8	
r8	0xa5a5a5a5	
r9	0xa5a5a5a5	
r10	0xa5a5a5a5	
r11	0xa5a5a5a5	
r12	0x28f	
sp	0x10001fd8	
lr	0x00000cb7	

```
adc Debug [C/C++ MCU Application]
  MCU GDB Debugger (6/10/11 3:40 PM) (Suspended)
    Thread [1] (Suspended: Breakpoint hit.)
      1 main() adc_main.c:87 0x0000050e
```

```
     * in the CMSIS system_<part family>.c file.
     */

  uint32_t i;

  /* Initialize ADC  */
  ADCInit( ADC_CLK );

  while(1)
  {
#if CONFIG_ADC_ENABLE_BURST_MODE==1             /* Interrupt driven only */
    ADCBurstRead();
    while ( !ADC0IntDone );
    ADC0IntDone = 0;
#else                          /* Not burst mode */
#if CONFIG_ADC_ENABLE_ADC_IRQHANDLER==1     /* Interrupt driven */
    for ( i = 0; i < ADC_NUM; i++ )
    {
      ADCRead( i );
      while ( !ADCIntDone );
      ADCIntDone = 0;
```

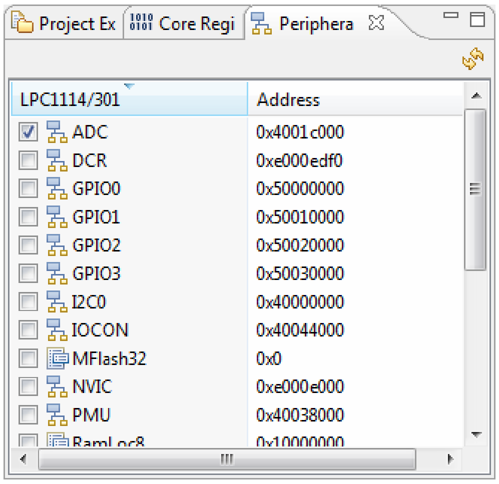
```
adc Debug [C/C++ MCU Application] C:\projects\lpcxpresso4_gs\NewWorkspace\adc\Debug\adc.axf (6/10/11 3:40 PM)
```

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.

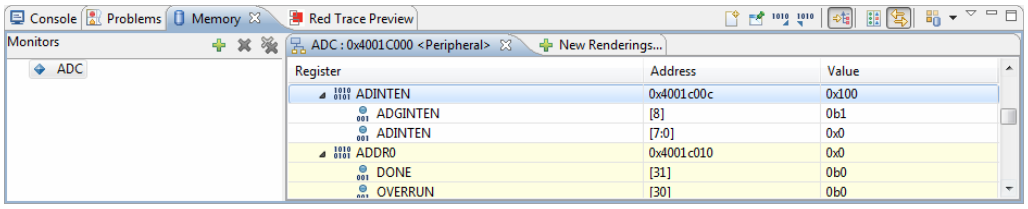
4.1.2.1 Peripheral views

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.



LPC1114/301	Address
☑ ADC	0x4001c000
☐ DCR	0xe000edf0
☐ GPIO0	0x50000000
☐ GPIO1	0x50010000
☐ GPIO2	0x50020000
☐ GPIO3	0x50030000
☐ I2C0	0x40000000
☐ IOCON	0x40044000
☐ MFlash32	0x0
☐ NVIC	0xe000e000
☐ PMU	0x40038000
☐ RamLoc8	0x10000000

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



Register	Address	Value
ADINTEN	0x4001c00c	0x100
ADGINTEN	[8]	0b1
ADINTEN	[7:0]	0x0
ADDR0	0x4001c010	0x0
DONE	[31]	0b0
OVERRUN	[30]	0b0

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.

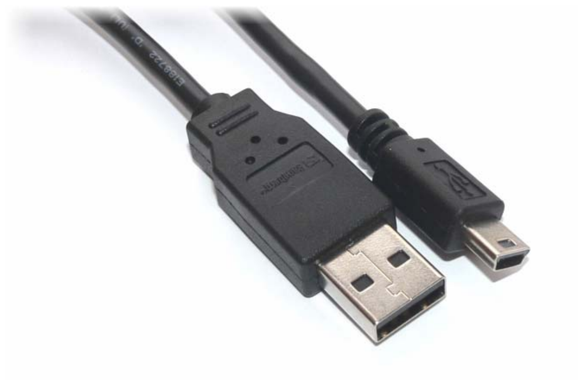


Fig 13. USB 2.0 A/Mini-B cable

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. Examples: Download, build, download and debug

5.1 Downloading NXP sample code from the web

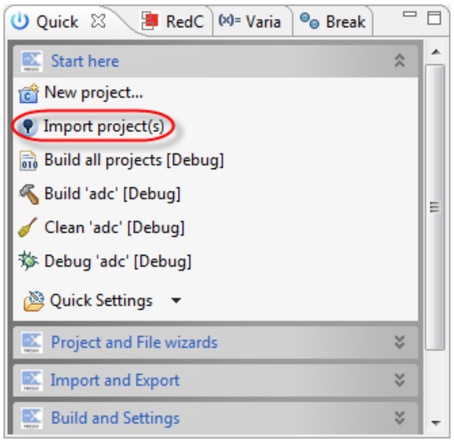
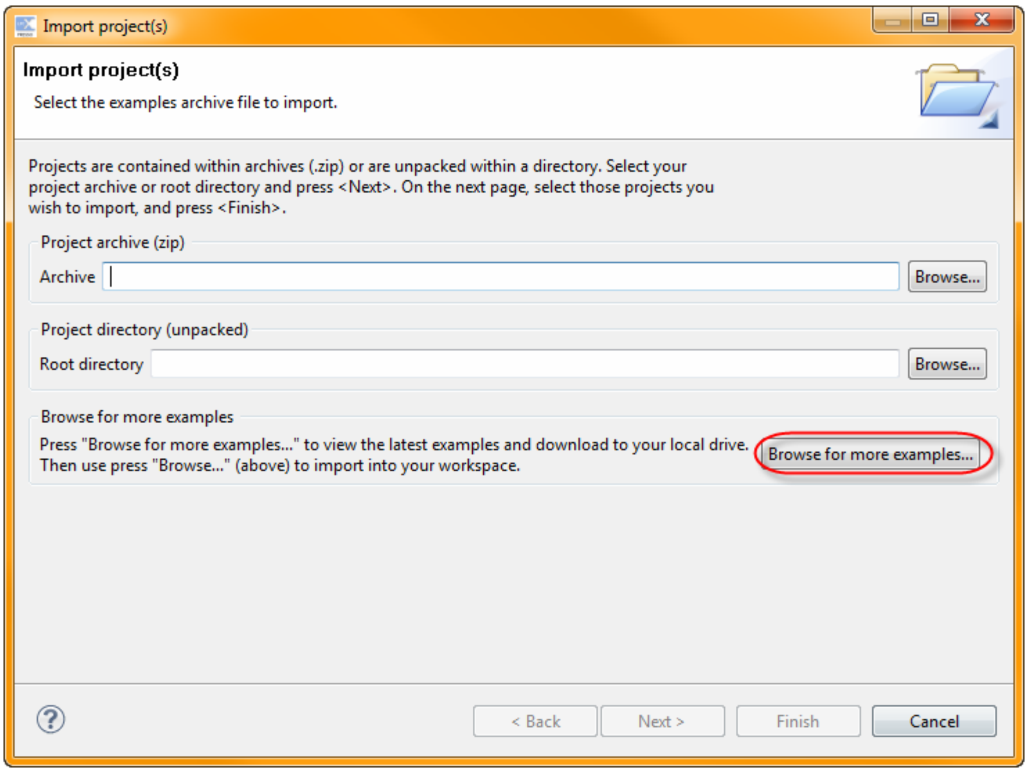


Fig 14. Quickstart panel

First, select “Import project(s)” from the Quickstart panel in the lower left corner of the screen.



Import project(s)

Select the examples archive file to import.

Projects are contained within archives (.zip) or are unpacked within a directory. Select your project archive or root directory and press <Next>. On the next page, select those projects you wish to import, and press <Finish>.

Project archive (zip)

Archive | Browse...

Project directory (unpacked)

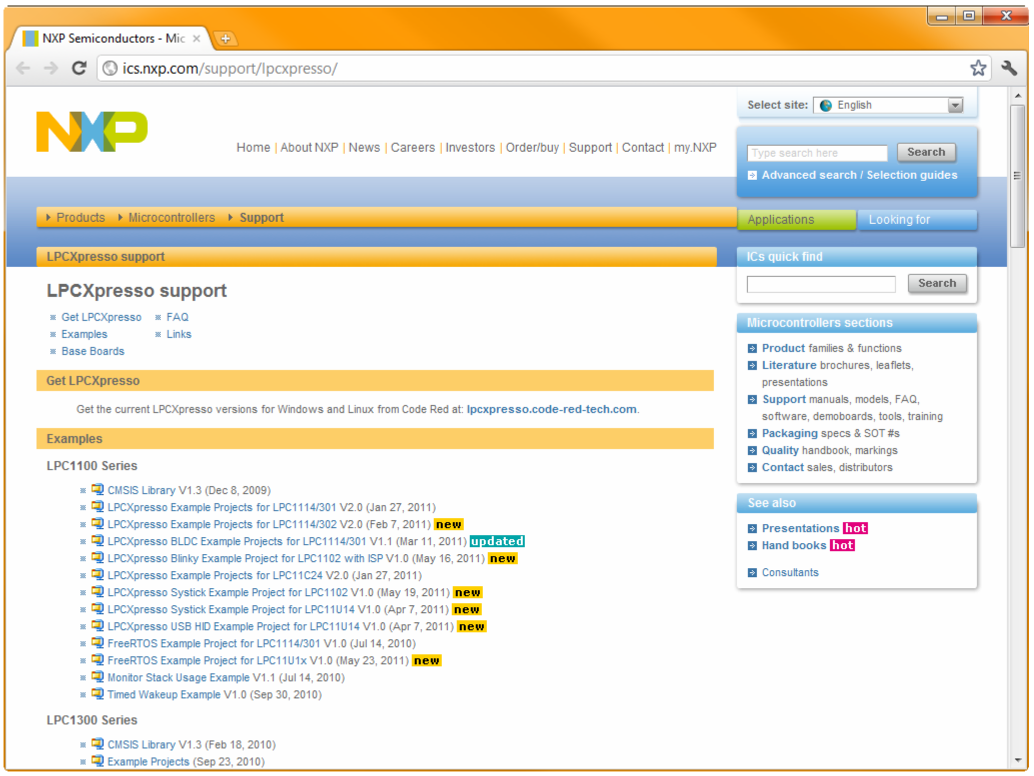
Root directory | Browse...

Browse for more examples

Press "Browse for more examples..." to view the latest examples and download to your local drive. Then use press "Browse..." (above) to import into your workspace. | Browse for more examples...

Fig 15. Import project(s) dialog

Next, select “Browse for more examples…” from the Import project(s) dialog.


NXP Semiconductors - Mic
ics.nxp.com/support/lpcxpresso/
Home | About NXP | News | Careers | Investors | Order/buy | Support | Contact | my.NXP
Select site: English
Type search here
Search
Advanced search / Selection guides
Products
Microcontrollers
Support
Applications
Looking for
ICs quick find
Search

LPCXpresso support

LPCXpresso support

- Get LPCXpresso
- FAQ
- Examples
- Links
- Base Boards

Get LPCXpresso

Get the current LPCXpresso versions for Windows and Linux from Code Red at: lpcxpresso.code-red-tech.com.

Examples

LPC1100 Series

- CMSIS Library V1.3 (Dec 8, 2009)
- LPCXpresso Example Projects for LPC1114/301 V2.0 (Jan 27, 2011)
- LPCXpresso Example Projects for LPC1114/302 V2.0 (Feb 7, 2011) new
- LPCXpresso BLDC Example Projects for LPC1114/301 V1.1 (Mar 11, 2011) updated
- LPCXpresso Blinky Example Project for LPC1102 with ISP V1.0 (May 16, 2011) new
- LPCXpresso Example Projects for LPC11C24 V2.0 (Jan 27, 2011)
- LPCXpresso Systick Example Project for LPC1102 V1.0 (May 19, 2011) new
- LPCXpresso Systick Example Project for LPC11U14 V1.0 (Apr 7, 2011) new
- LPCXpresso USB HID Example Project for LPC11U14 V1.0 (Apr 7, 2011) new
- FreeRTOS Example Project for LPC1114/301 V1.0 (Jul 14, 2010)
- FreeRTOS Example Project for LPC11U1x V1.0 (May 23, 2011) new
- Monitor Stack Usage Example V1.1 (Jul 14, 2010)
- Timed Wakeup Example V1.0 (Sep 30, 2010)

LPC1300 Series

- CMSIS Library V1.3 (Feb 18, 2010)
- Example Projects (Sep 23, 2010)

Microcontrollers sections

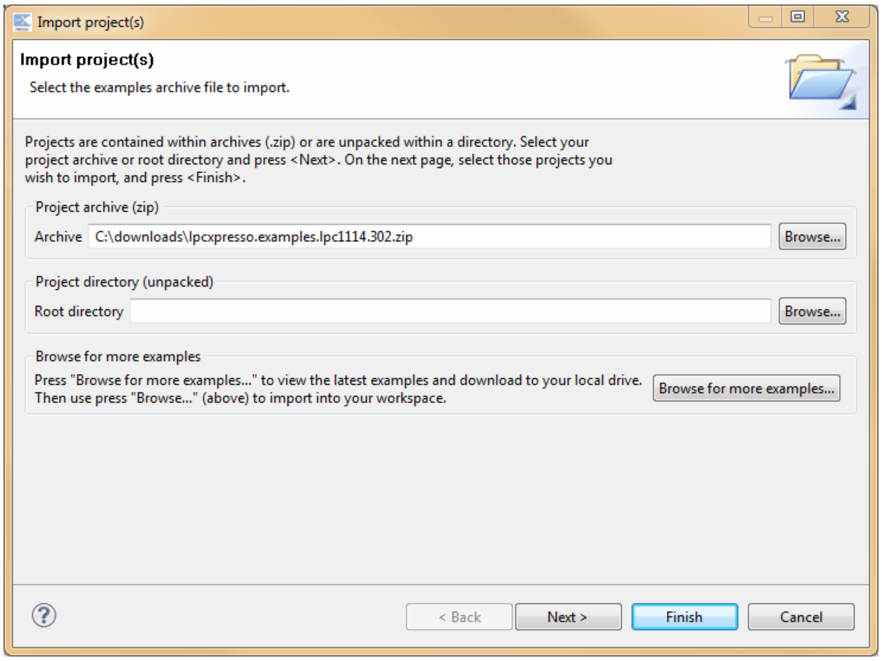
- **Product** families & functions
- **Literature** brochures, leaflets, presentations
- **Support** manuals, models, FAQ, software, demoboards, tools, training
- **Packaging** specs & SOT #s
- **Quality** handbook, markings
- **Contact** sales, distributors

See also

- **Presentations** hot
- **Hand books** hot
- Consultants

Fig 16. http://www.nxp.com/lpcxpresso-support web site

The LPCXpresso-Support web page should appear. It has links to NXP sample code that has been developed for LPCXpresso. Select the sample code that applies to your Target Device and download it. Then, switch back to LPCXpresso and use the Browse button to select the .zip file.



Import project(s)

Import project(s)

Select the examples archive file to import.

Projects are contained within archives (.zip) or are unpacked within a directory. Select your project archive or root directory and press <Next>. On the next page, select those projects you wish to import, and press <Finish>.

Project archive (zip)

Archive: C:\downloads\lpcxpresso.examples.lpc1114.302.zip

Project directory (unpacked)

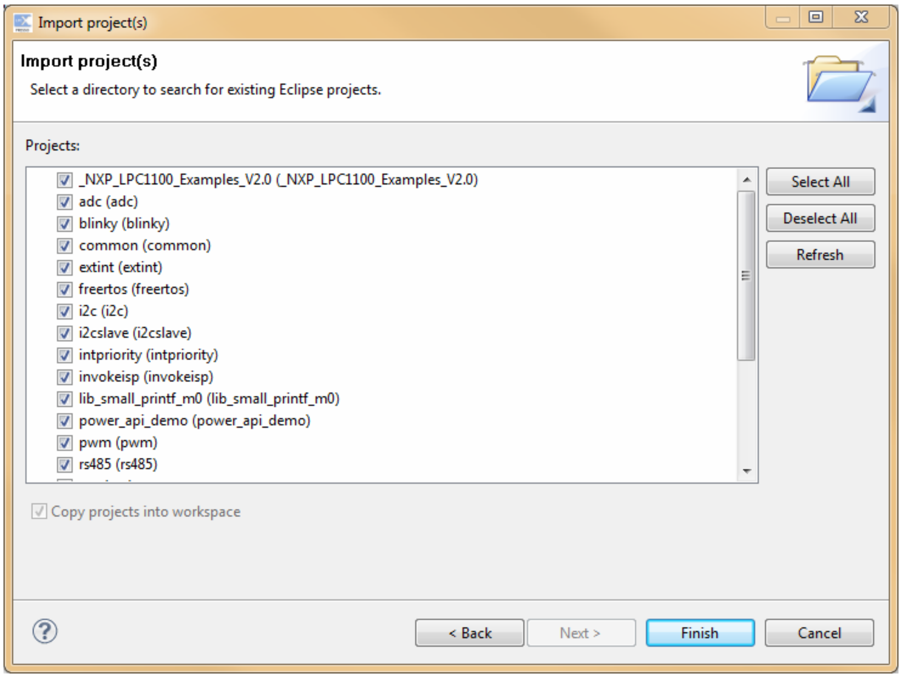
Root directory:

Browse for more examples

Press "Browse for more examples..." to view the latest examples and download to your local drive. Then use press "Browse..." (above) to import into your workspace.

Fig 17. Zip file from LPCXpresso-Support web site selected for importing

Now click the Next button and then choose which projects to import from the .zip file.



Import project(s)

Import project(s)

Select a directory to search for existing Eclipse projects.

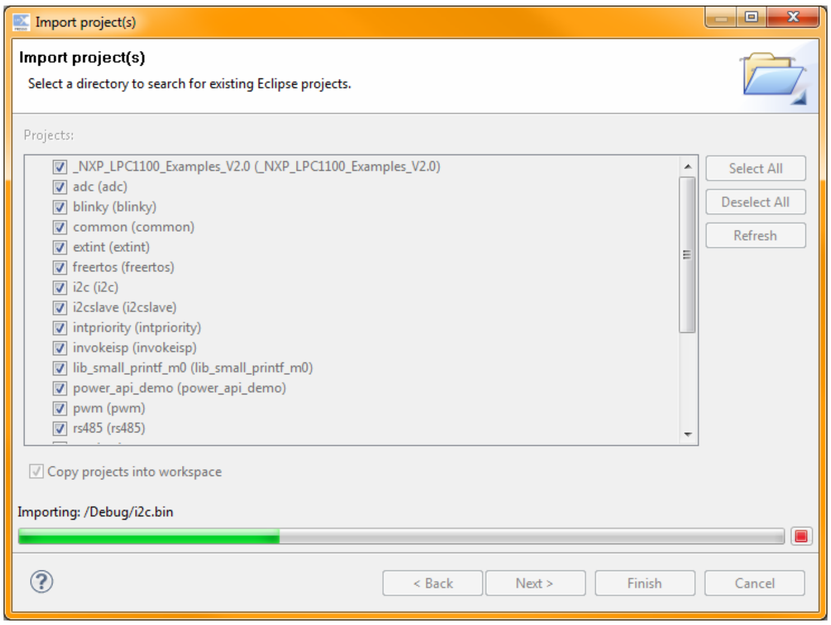
Projects:

- [✓] _NXP_LPC1100_Examples_V2.0 (_NXP_LPC1100_Examples_V2.0)
- [✓] adc (adc)
- [✓] blinky (blinky)
- [✓] common (common)
- [✓] extint (extint)
- [✓] freertos (freertos)
- [✓] i2c (i2c)
- [✓] i2cslave (i2cslave)
- [✓] intpriority (intpriority)
- [✓] invokeisp (invokeisp)
- [✓] lib_small_printf_m0 (lib_small_printf_m0)
- [✓] power_api_demo (power_api_demo)
- [✓] pwm (pwm)
- [✓] rs485 (rs485)

[✓] Copy projects into workspace

Fig 18. Import projects(s):-Selecting which projects to import

Often, there will be references between projects in a .zip file so it is best to import all of them.


Import project(s)

Select a directory to search for existing Eclipse projects.

Projects:

- ☑ _NXP_LPC1100_Examples_V2.0 (_NXP_LPC1100_Examples_V2.0)
- ☑ adc (adc)
- ☑ blinky (blinky)
- ☑ common (common)
- ☑ extint (extint)
- ☑ freertos (freertos)
- ☑ i2c (i2c)
- ☑ i2cslave (i2cslave)
- ☑ intpriority (intpriority)
- ☑ invokeisp (invokeisp)
- ☑ lib_small_printf_m0 (lib_small_printf_m0)
- ☑ power_api_demo (power_api_demo)
- ☑ pwm (pwm)
- ☑ rs485 (rs485)

☑ Copy projects into workspace

Importing: /Debug/i2c.bin

Fig 19. Import projects: Progress indicator

- There is also a local LPCXpresso examples directory. This is often located at C:\nxp\lpcxpresso\lpcxpresso\examples.

5.2 Debugging/running 'blinky' or another project 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 'Projectname' [Debug].

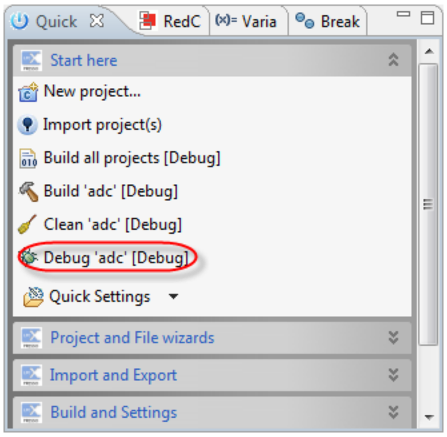


Fig 20. Debug

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

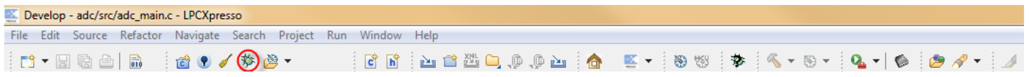
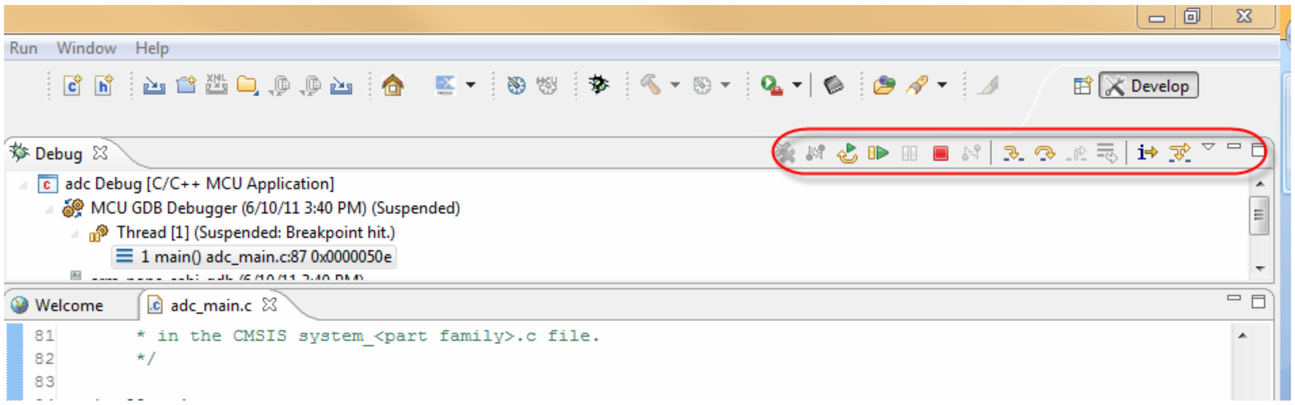

Develop - adc/src/adc_main.c - LPCXpresso
File Edit Source Refactor Navigate Search Project Run Window Help

Fig 21. Bug icon

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



```
adc Debug [C/C++ MCU Application]
  MCU GDB Debugger (6/10/11 3:40 PM) (Suspended)
    Thread [1] (Suspended: Breakpoint hit.)
      1 main() adc_main.c:87 0x0000050e
```

```
 * in the CMSIS system_<part family>.c file.
 */
```

Fig 22. 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.
i⇒	Instruction stepping mode (disassembly)

Fig 23. Debug buttons

6. LPCXpresso IDE tips and tricks

6.1 Installing Eclipse plugins

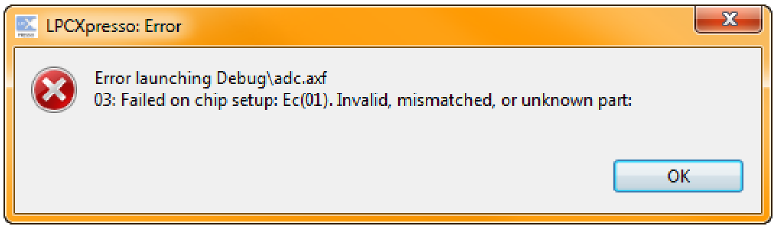
The LPCXpresso IDE contains many of the features of the Eclipse open-source IDE from http://www.eclipse.org. The browse and install plugin function is present in the help menu. To access it, choose Help -> Install New Software. This will display the Eclipse Install Software dialog which will allow browsing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

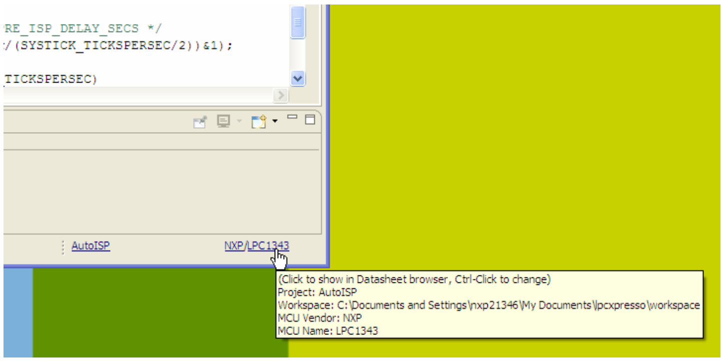


LPCXpresso: Error

```
Error launching Debug\adc.axf
03: Failed on chip setup: Ec(01). Invalid, mismatched, or unknown part:
```

Fig 24. Error starting debug

LPCXpresso checks the target chip ID against the currently selected chip ID for the project and will not start if there is not a match. 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.

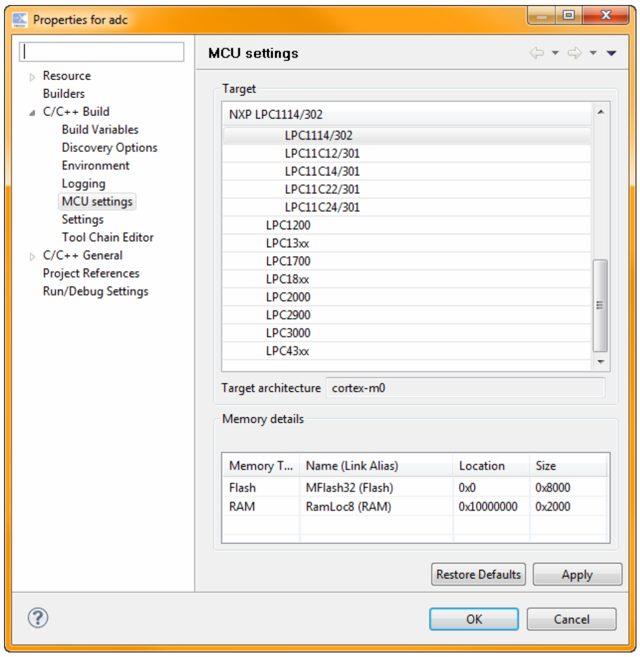


```
RE_ISP_DELAY_SECS */
/(SYSTICK_TICKSPERSEC/2))&1);

TICKSPERSEC)
```

```
(Click to show in Datasheet browser, Ctrl-Click to change)
Project: AutoISP
Workspace: C:\Documents and Settings\nxp21346\My Documents\lpcxpresso\workspace
MCU Vendor: NXP
MCU Name: LPC1343
```

Fig 25. Current part number



MCU settings

Target

NXP LPC1114/302
- LPC1114/302
- LPC11C12/301
- LPC11C14/301
- LPC11C22/301
- LPC11C24/301
- LPC1200
- LPC13xx
- LPC1700
- LPC18xx
- LPC2000
- LPC2900
- LPC3000
- LPC43xx

Target architecture: cortex-m0

Memory details

Memory T...	Name (Link Alias)	Location	Size
Flash	MFlash32 (Flash)	0x0	0x8000
RAM	RamLoc8 (RAM)	0x10000000	0x2000

Fig 26. 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.

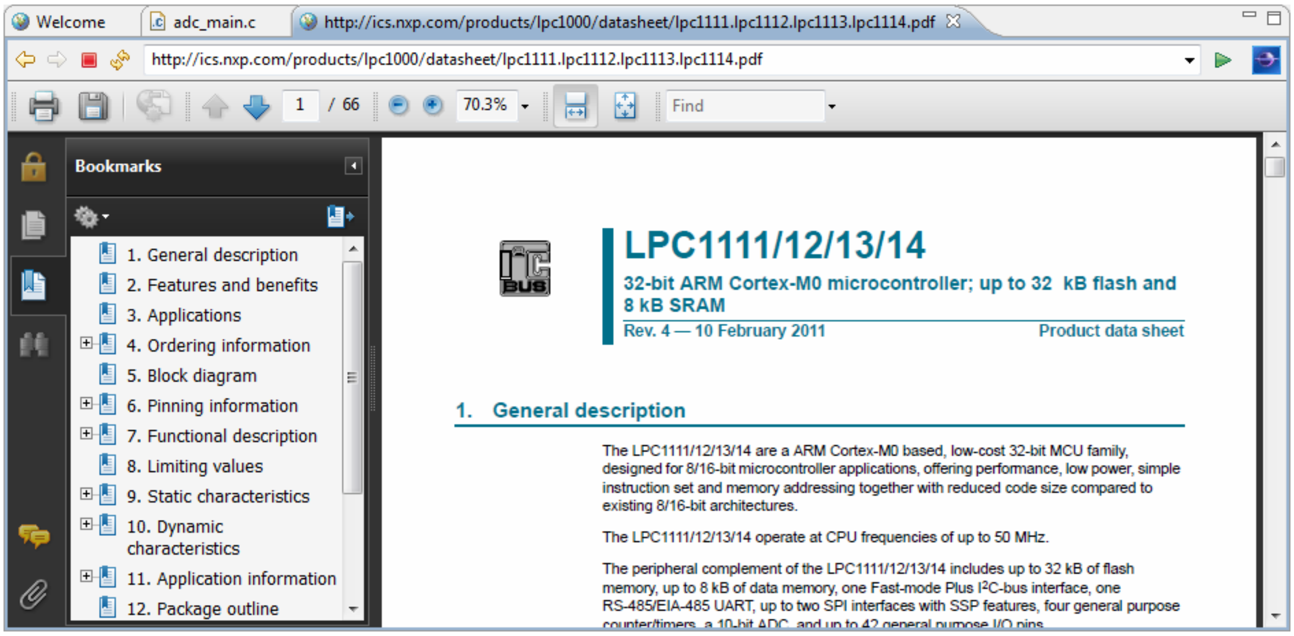
6.2.6 Recovery of board

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 part number in the lower right border of the LPCXpresso window to see the browser in action.



LPC1111/12/13/14

32-bit ARM Cortex-M0 microcontroller; up to 32 kB flash and 8 kB SRAM

Rev. 4 — 10 February 2011 Product data sheet

1. General description

The LPC1111/12/13/14 are a ARM Cortex-M0 based, low-cost 32-bit MCU family, designed for 8/16-bit microcontroller applications, offering performance, low power, simple instruction set and memory addressing together with reduced code size compared to existing 8/16-bit architectures.

The LPC1111/12/13/14 operate at CPU frequencies of up to 50 MHz.

The peripheral complement of the LPC1111/12/13/14 includes up to 32 kB of flash memory, up to 8 kB of data memory, one Fast-mode Plus I^{2}C-bus interface, one RS-485/EIA-485 UART, up to two SPI interfaces with SSP features, four general purpose counter/timers, a 10-bit ADC, and up to 42 general purpose I/O pins.

Fig 27. 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 can be set using the Quick Settings dropdown box in the Quick Start panel.

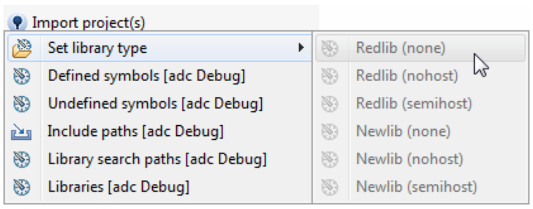


Fig 28. Reconfigure library setting

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.

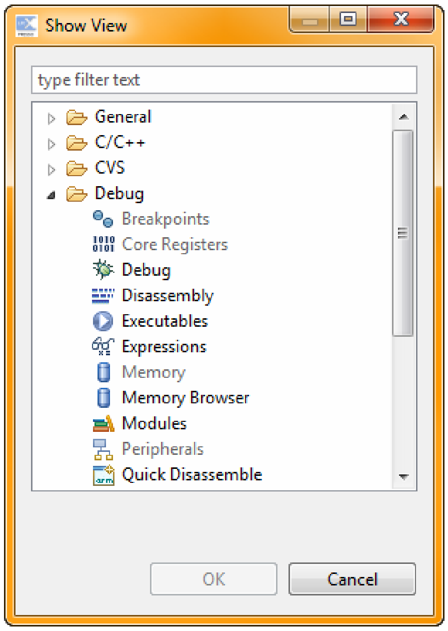


Fig 29. Show view window

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

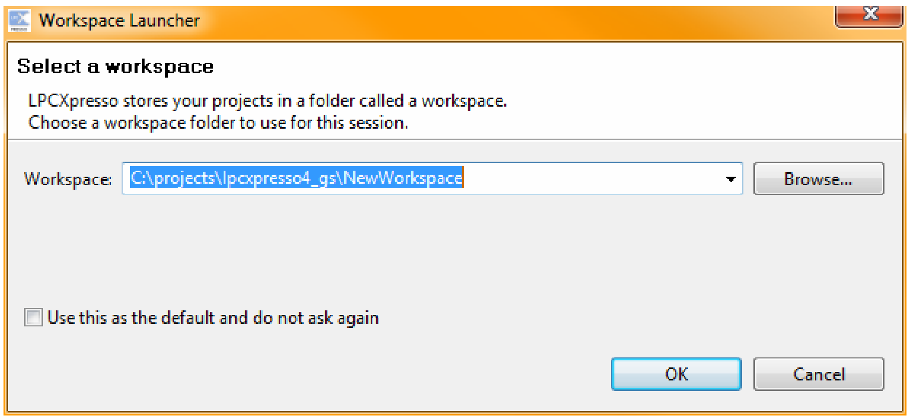
6.6 Creating a ‘skeleton’ project in a new Workspace

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’.



Select a workspace

LPCXpresso stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\projects\lpcxpresso4_gs\NewWorkspace

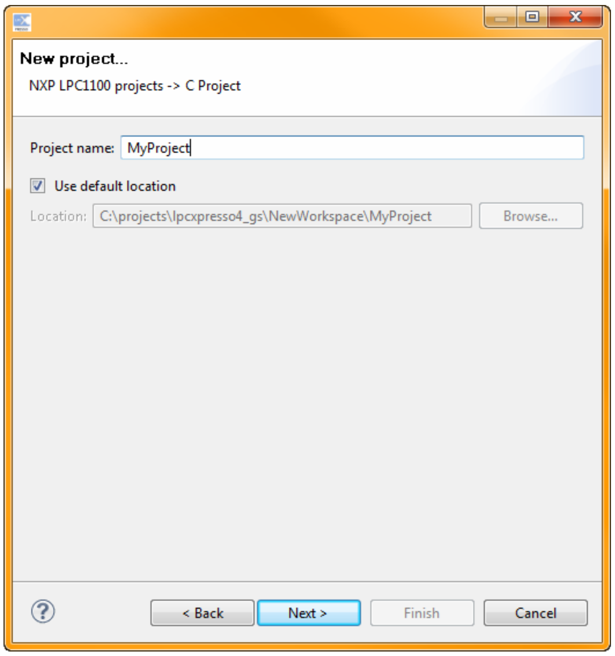
☐ Use this as the default and do not ask again

Fig 30. 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 Project” and navigate to the CMSIS<version/part>.zip. The CMSIS header files are installed with LPCXpresso in C:\nxp\lpcxpresso\lpcxpresso\examples\nxp. Once this project is added to your workspace, click “Build all projects (Debug)” in the Quickstart Panel.
- Click on ‘New project…” and select the NXP C project type for your architecture.
- Click “Next” and enter a project name. In this case we will use ‘MyProject’ then click ‘Next.’



New project...

NXP LPC1100 projects -> C Project

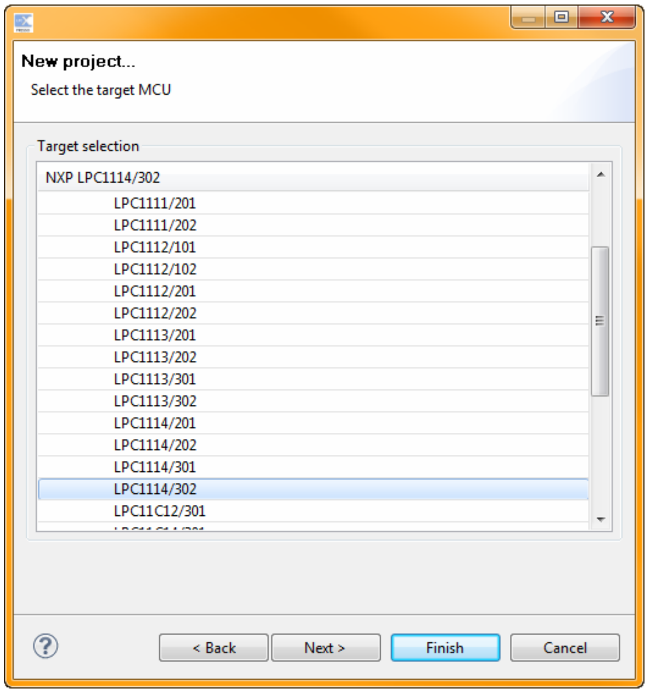
Project name: MyProject

[✓] Use default location

Location: C:\projects\lpcxpresso4_gs\NewWorkspace\MyProject

Fig 31. Enter project name

- The next section of the dialog will ask you to specify which chip in the family you are using.



New project...

Select the target MCU

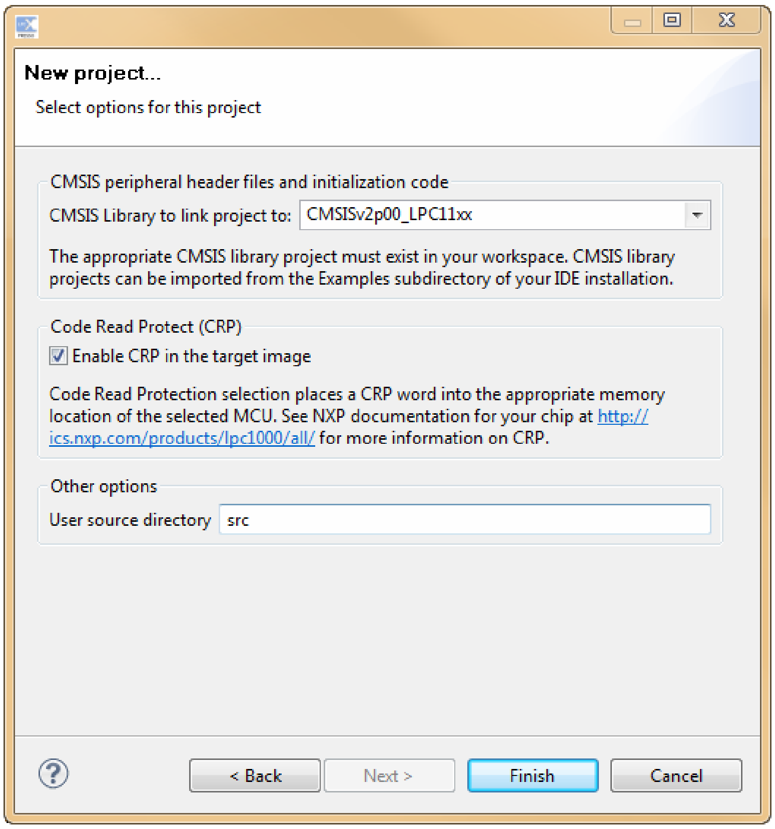
Target selection

NXP LPC1114/302
LPC1111/201
LPC1111/202
LPC1112/101
LPC1112/102
LPC1112/201
LPC1112/202
LPC1113/201
LPC1113/202
LPC1113/301
LPC1113/302
LPC1114/201
LPC1114/202
LPC1114/301
LPC1114/302
LPC11C12/301

Fig 32. Enter project name

-

- If you are using a Cortex-based part, the next step in the wizard will ask which CMSIS project to use. CMSIS stands for Cortex Microcontroller Software Interface Standard. CMSIS defines a common way to access peripheral registers and to define interrupts. Please select the project that you imported into the workspace earlier and click Finish.



New project...

Select options for this project

CMSIS peripheral header files and initialization code

CMSIS Library to link project to: CMSISv2p00_LPC11xx

The appropriate CMSIS library project must exist in your workspace. CMSIS library projects can be imported from the Examples subdirectory of your IDE installation.

Code Read Protect (CRP)

☑ Enable CRP in the target image

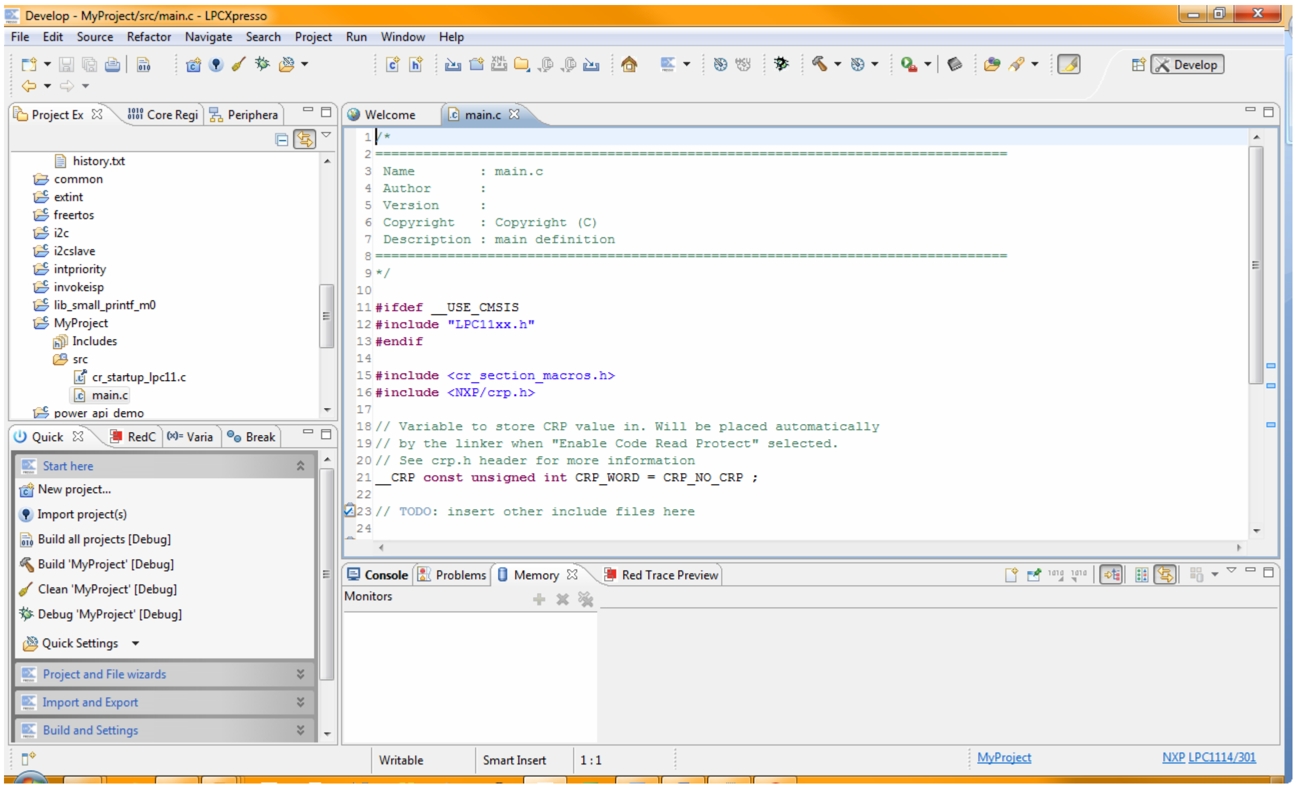
Code Read Protection selection places a CRP word into the appropriate memory location of the selected MCU. See NXP documentation for your chip at http://ics.nxp.com/products/lpc1000/all/ for more information on CRP.

Other options

User source directory: src

Fig 33. CMSIS selection

Congratulations! You have created your first project!



```
/*
===============================================================================
 Name        : main.c
 Author      :
 Version     :
 Copyright   : Copyright (C)
 Description : main definition
===============================================================================
*/

#ifdef __USE_CMSIS
#include "LPC11xx.h"
#endif

#include <cr_section_macros.h>
#include <NXP/crp.h>

// Variable to store CRP value in. Will be placed automatically
// by the linker when "Enable Code Read Protect" selected.
// See crp.h header for more information
__CRP const unsigned int CRP_WORD = CRP_NO_CRP ;

// TODO: insert include other files here

```

Fig 34. Project creation complete

7. Appendix

7.1 LPCXpresso target side schematics

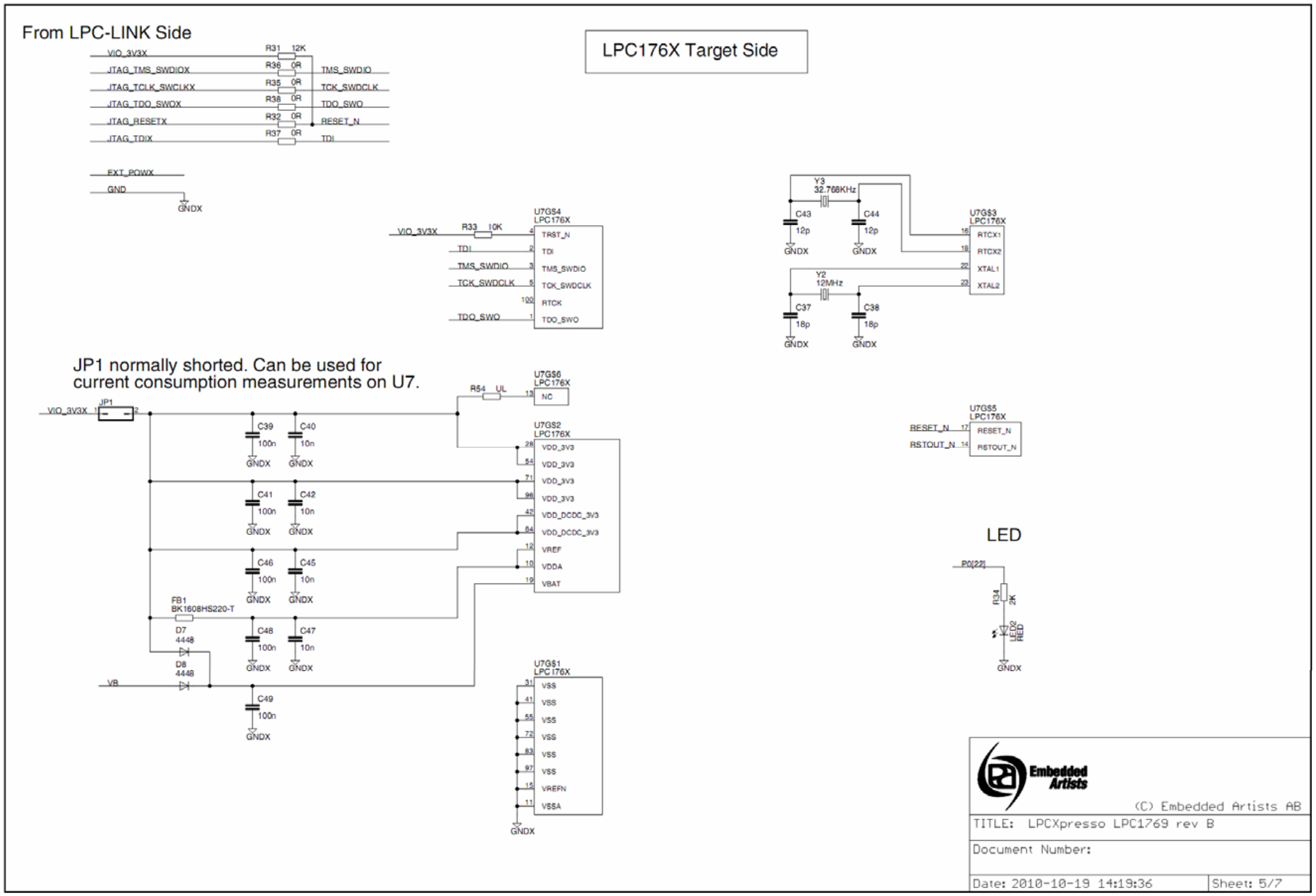


Fig 35. Schematic for the LPCXpresso LPC1769 target side (1 of 3)

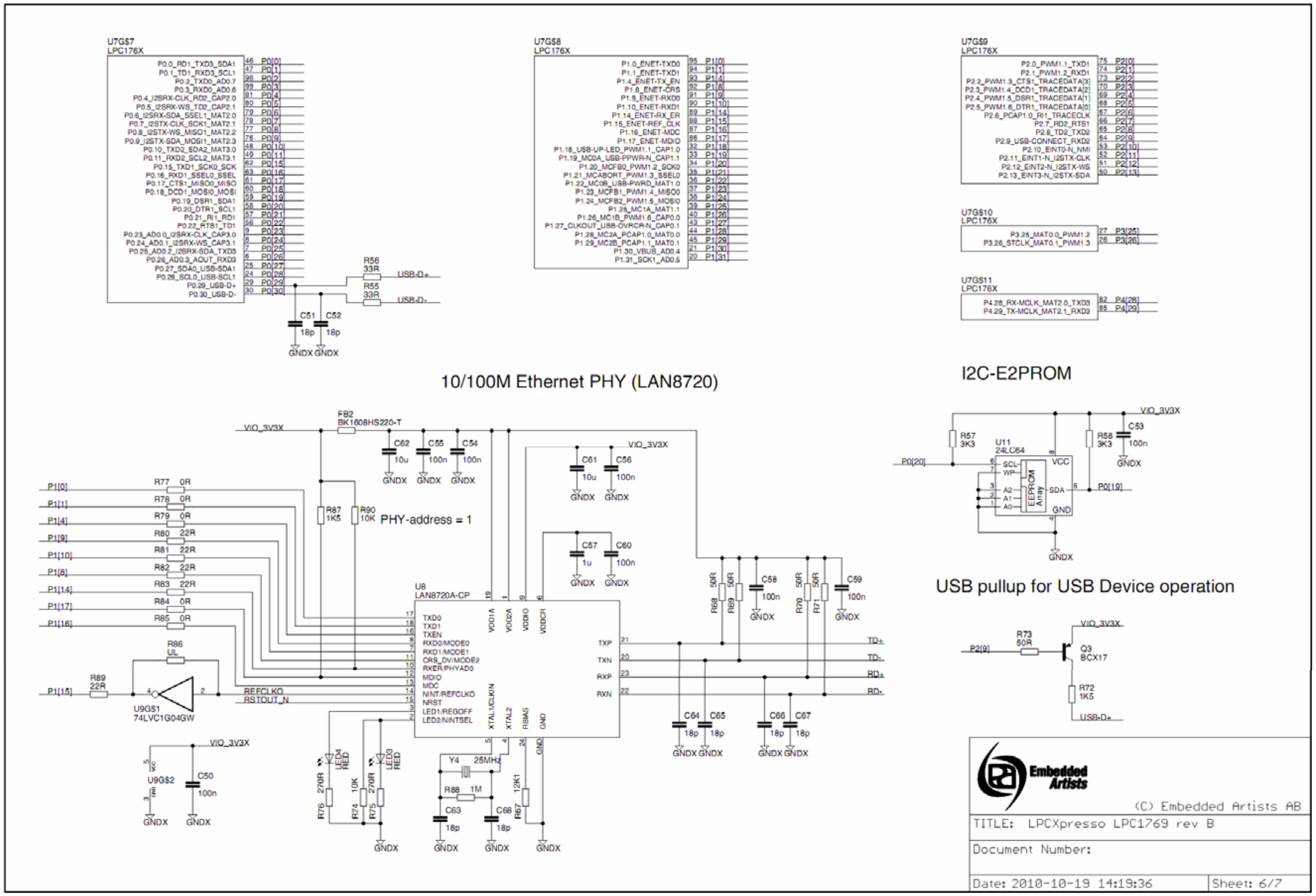
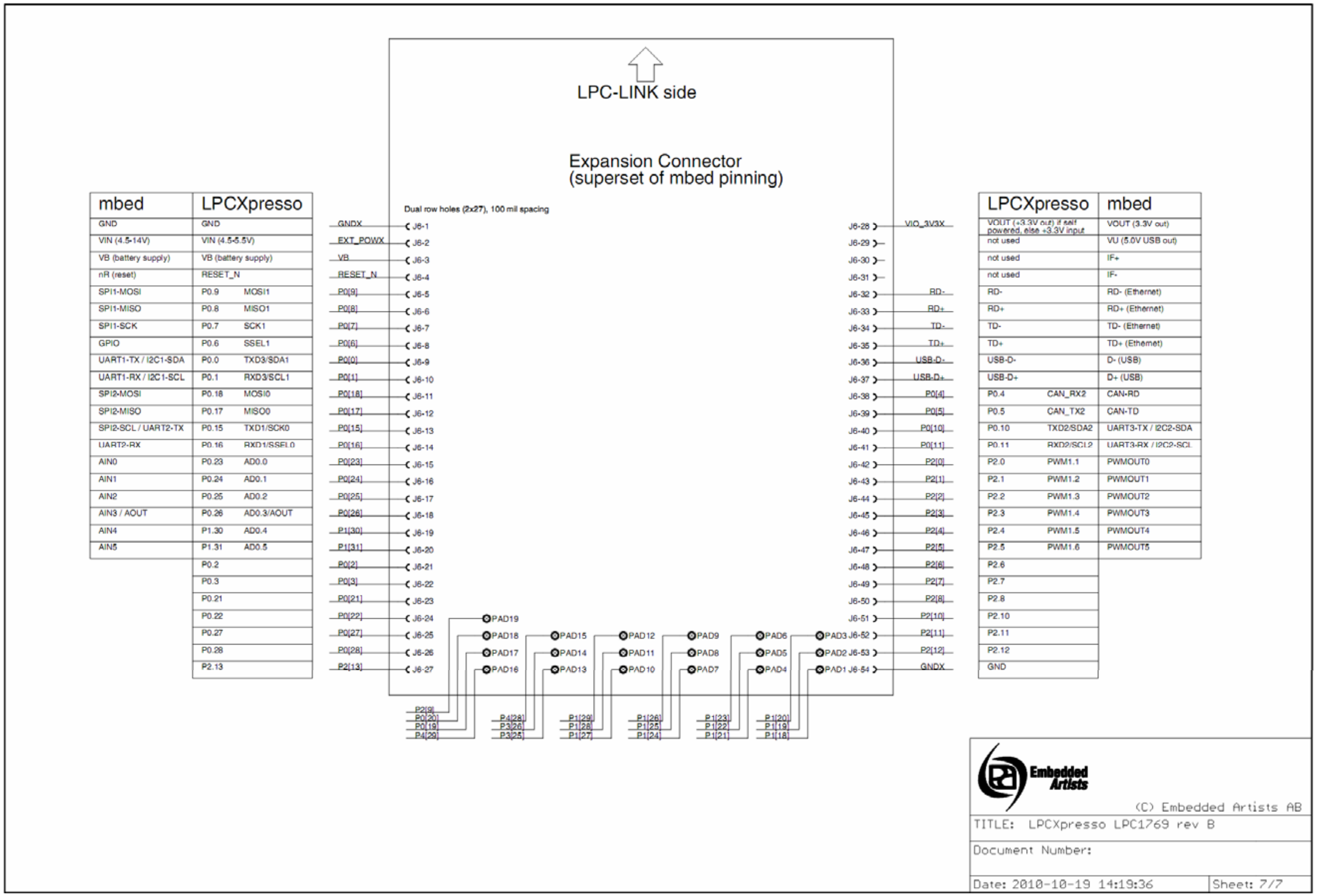


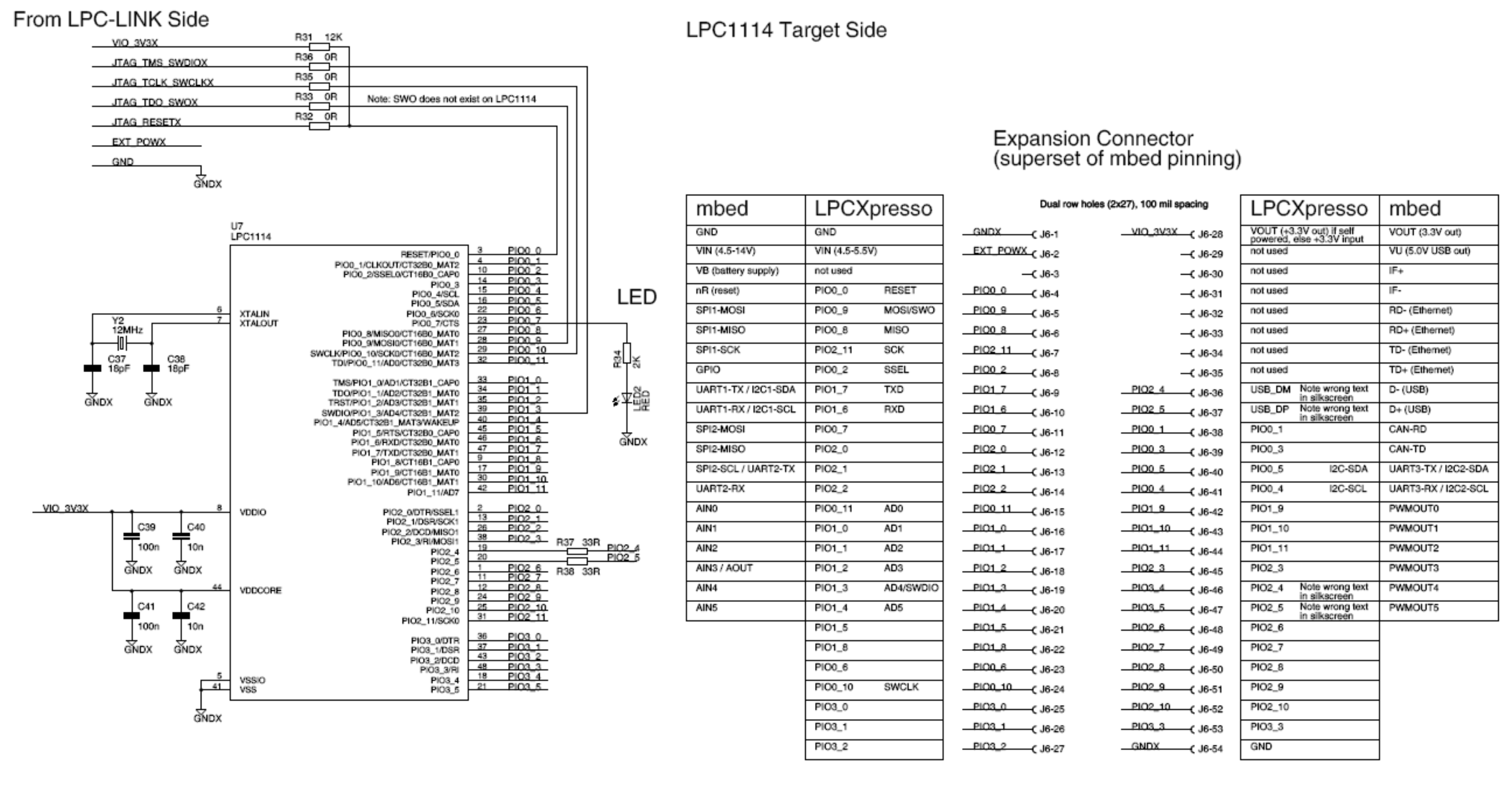
Fig 36. Schematic for the LPCXpresso LPC1769 target side (2 of 3)


LPC-LINK side
Expansion Connector
(superset of mbed pinning)
Dual row holes (2x27), 100 mil spacing
(C) Embedded Artists AB
TITLE: LPCXpresso LPC1769 rev B
Document Number:
Date: 2010-10-19 14:19:36
Sheet: 7/7

mbed	LPCXpresso		Net	Pin
GND	GND		GNDX	J6-1
VIN (4.5-14V)	VIN (4.5-5.5V)		EXT_POWX	J6-2
VB (battery supply)	VB (battery supply)		VB	J6-3
nR (reset)	RESET_N		RESET_N	J6-4
SPI1-MOSI	P0.9	MOSI1	P0[9]	J6-5
SPI1-MISO	P0.8	MISO1	P0[8]	J6-6
SPI1-SCK	P0.7	SCK1	P0[7]	J6-7
GPIO	P0.6	SSEL1	P0[6]	J6-8
UART1-TX / I2C1-SDA	P0.0	TXD3/SDA1	P0[0]	J6-9
UART1-RX / I2C1-SCL	P0.1	RXD3/SCL1	P0[1]	J6-10
SPI2-MOSI	P0.18	MOSI0	P0[18]	J6-11
SPI2-MISO	P0.17	MISO0	P0[17]	J6-12
SPI2-SCL / UART2-TX	P0.15	TXD1/SCK0	P0[15]	J6-13
UART2-RX	P0.16	RXD1/SSEL0	P0[16]	J6-14
AIN0	P0.23	AD0.0	P0[23]	J6-15
AIN1	P0.24	AD0.1	P0[24]	J6-16
AIN2	P0.25	AD0.2	P0[25]	J6-17
AIN3 / AOUT	P0.26	AD0.3/AOUT	P0[26]	J6-18
AIN4	P1.30	AD0.4	P1[30]	J6-19
AIN5	P1.31	AD0.5	P1[31]	J6-20
	P0.2		P0[2]	J6-21
	P0.3		P0[3]	J6-22
	P0.21		P0[21]	J6-23
	P0.22		P0[22]	J6-24
	P0.27		P0[27]	J6-25
	P0.28		P0[28]	J6-26
	P2.13		P2[13]	J6-27

Pin	Net	LPCXpresso		mbed
J6-28	VIO_3V3X	VOUT (+3.3V out) if self powered, else +3.3V input		VOUT (3.3V out)
J6-29		not used		VU (5.0V USB out)
J6-30		not used		IF+
J6-31		not used		IF-
J6-32	RD-	RD-		RD- (Ethernet)
J6-33	RD+	RD+		RD+ (Ethernet)
J6-34	TD-	TD-		TD- (Ethernet)
J6-35	TD+	TD+		TD+ (Ethernet)
J6-36	USB-D-	USB-D-		D- (USB)
J6-37	USB-D+	USB-D+		D+ (USB)
J6-38	P0[4]	P0.4	CAN_RX2	CAN-RD
J6-39	P0[5]	P0.5	CAN_TX2	CAN-TD
J6-40	P0[10]	P0.10	TXD2/SDA2	UART3-TX / I2C2-SDA
J6-41	P0[11]	P0.11	RXD2/SCL2	UART3-RX / I2C2-SCL
J6-42	P2[0]	P2.0	PWM1.1	PWMOUT0
J6-43	P2[1]	P2.1	PWM1.2	PWMOUT1
J6-44	P2[2]	P2.2	PWM1.3	PWMOUT2
J6-45	P2[3]	P2.3	PWM1.4	PWMOUT3
J6-46	P2[4]	P2.4	PWM1.5	PWMOUT4
J6-47	P2[5]	P2.5	PWM1.6	PWMOUT5
J6-48	P2[6]	P2.6		
J6-49	P2[7]	P2.7		
J6-50	P2[8]	P2.8		
J6-51	P2[10]	P2.10		
PAD3 J6-52	P2[11]	P2.11		
PAD2 J6-53	P2[12]	P2.12		
PAD1 J6-54	GNDX	GND		

Fig 37. Schematic for the LPCXpresso LPC1769 target side (3 of 3)

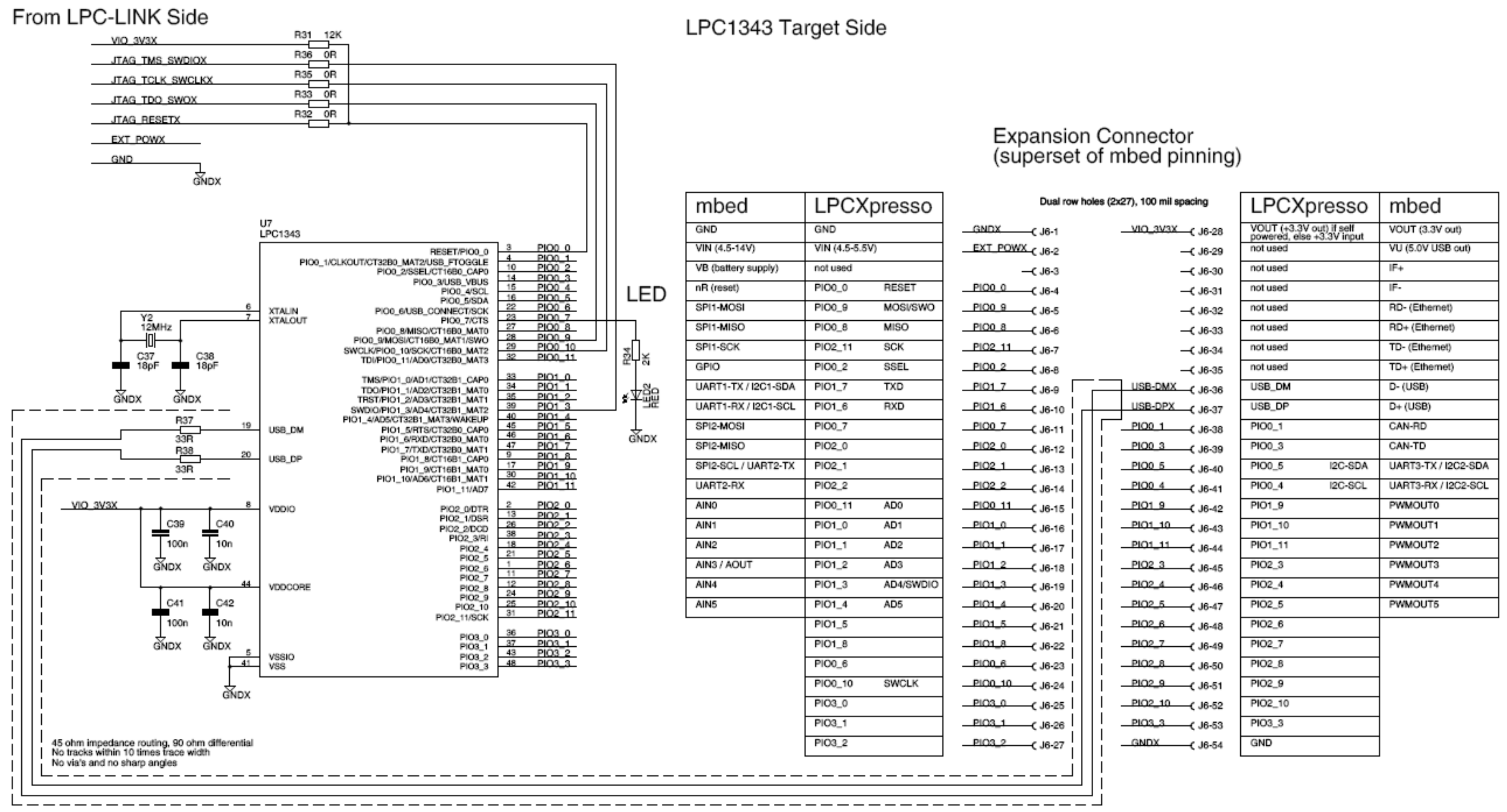

From LPC-LINK Side
LPC1114 Target Side
Note: SWO does not exist on LPC1114
LED
Expansion Connector
(superset of mbed pinning)
Dual row holes (2x27), 100 mil spacing

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	

LPCXpresso		mbed
VOUT (+3.3V out) if self powered, else +3.3V input		VOUT (3.3V out)
not used		VU (5.0V USB out)
not used		IF+
not used		IF-
not used		RD- (Ethernet)
not used		RD+ (Ethernet)
not used		TD- (Ethernet)
not used		TD+ (Ethernet)
USB_DM	Note wrong text in silkscreen	D- (USB)
USB_DP	Note wrong text in silkscreen	D+ (USB)
PIO0_1		CAN-RD
PIO0_3		CAN-TD
PIO0_5	I2C-SDA	UART3-TX / I2C2-SDA
PIO0_4	I2C-SCL	UART3-RX / I2C2-SCL
PIO1_9		PWMOUT0
PIO1_10		PWMOUT1
PIO1_11		PWMOUT2
PIO2_3		PWMOUT3
PIO2_4	Note wrong text in silkscreen	PWMOUT4
PIO2_5	Note wrong text in silkscreen	PWMOUT5
PIO2_6		
PIO2_7		
PIO2_8		
PIO2_9		
PIO2_10		
PIO3_3		
GND		

Fig 38. Schematic for the LPCXpresso LPC1114 target side[1]

1. 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).



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	

LPCXpresso		mbed
VOUT (+3.3V out) if self powered, else +3.3V input		VOUT (3.3V out)
not used		VU (5.0V USB out)
not used		IF+
not used		IF-
not used		RD- (Ethernet)
not used		RD+ (Ethernet)
not used		TD- (Ethernet)
not used		TD+ (Ethernet)
USB_DM		D- (USB)
USB_DP		D+ (USB)
PIO0_1		CAN-RD
PIO0_3		CAN-TD
PIO0_5	I2C-SDA	UART3-TX / I2C2-SDA
PIO0_4	I2C-SCL	UART3-RX / I2C2-SCL
PIO1_9		PWMOUT0
PIO1_10		PWMOUT1
PIO1_11		PWMOUT2
PIO2_3		PWMOUT3
PIO2_4		PWMOUT4
PIO2_5		PWMOUT5
PIO2_6		
PIO2_7		
PIO2_8		
PIO2_9		
PIO2_10		
PIO3_3		
GND		

Fig 39. Schematic for the LPCXpresso LPC1343 target side

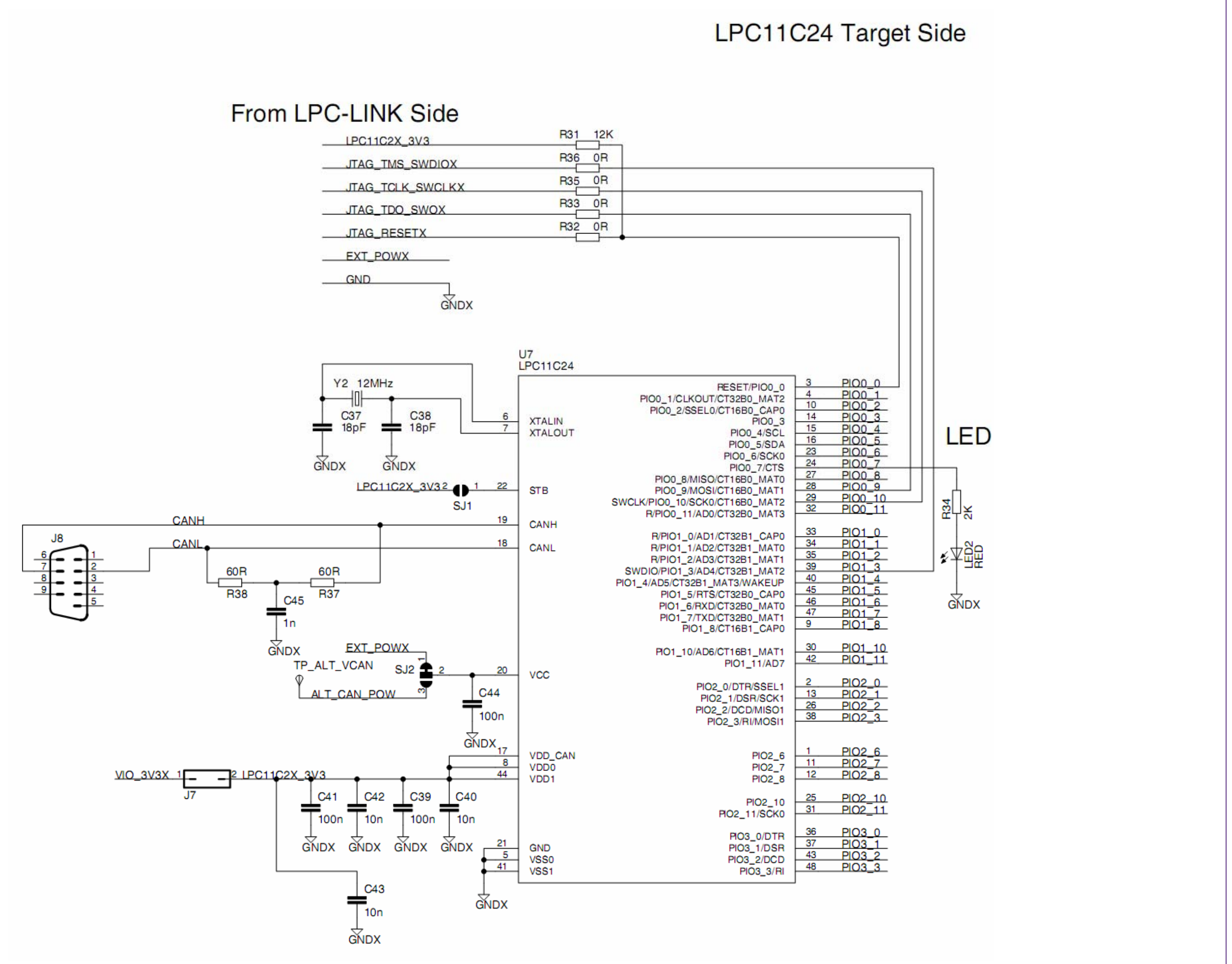
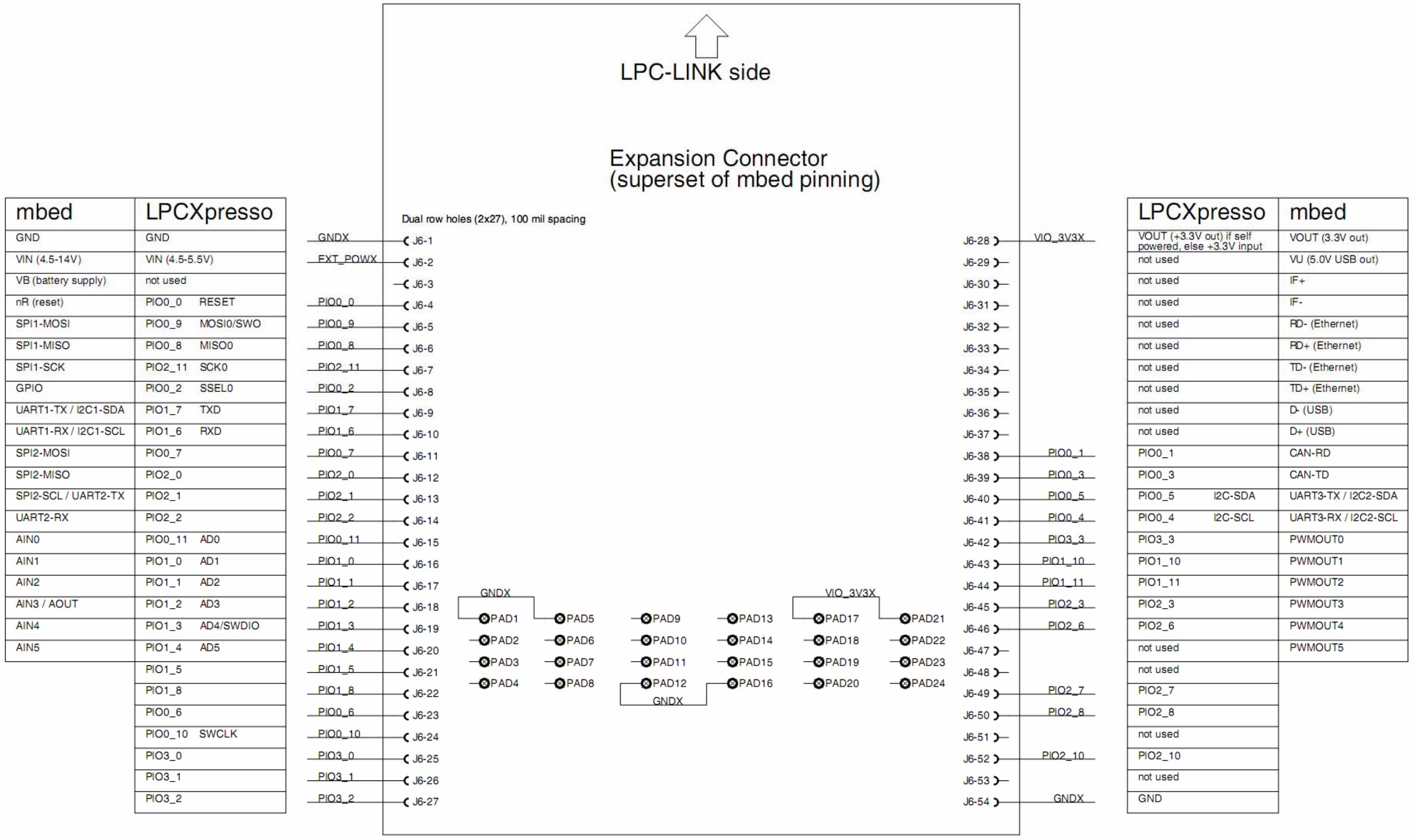


Fig 40. Schematic for the LPCXpresso LPC11C24 target side, part 1



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 MOSI0/SWO
SPI1-MISO	PIO0_8 MISO0
SPI1-SCK	PIO2_11 SCK0
GPIO	PIO0_2 SSEL0
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

LPCXpresso	mbed
VOUT (+3.3V out) if self powered, else +3.3V input	VOUT (3.3V out)
not used	VU (5.0V USB out)
not used	IF+
not used	IF-
not used	RD- (Ethernet)
not used	RD+ (Ethernet)
not used	TD- (Ethernet)
not used	TD+ (Ethernet)
not used	D- (USB)
not used	D+ (USB)
PIO0_1	CAN-RD
PIO0_3	CAN-TD
PIO0_5 I2C-SDA	UART3-TX / I2C2-SDA
PIO0_4 I2C-SCL	UART3-RX / I2C2-SCL
PIO3_3	PWMOUT0
PIO1_10	PWMOUT1
PIO1_11	PWMOUT2
PIO2_3	PWMOUT3
PIO2_6	PWMOUT4
not used	PWMOUT5
not used	
PIO2_7	
PIO2_8	
not used	
PIO2_10	
not used	
GND	

Fig 41. Schematic for the LPCXpresso LPC11C24 target side, part 2

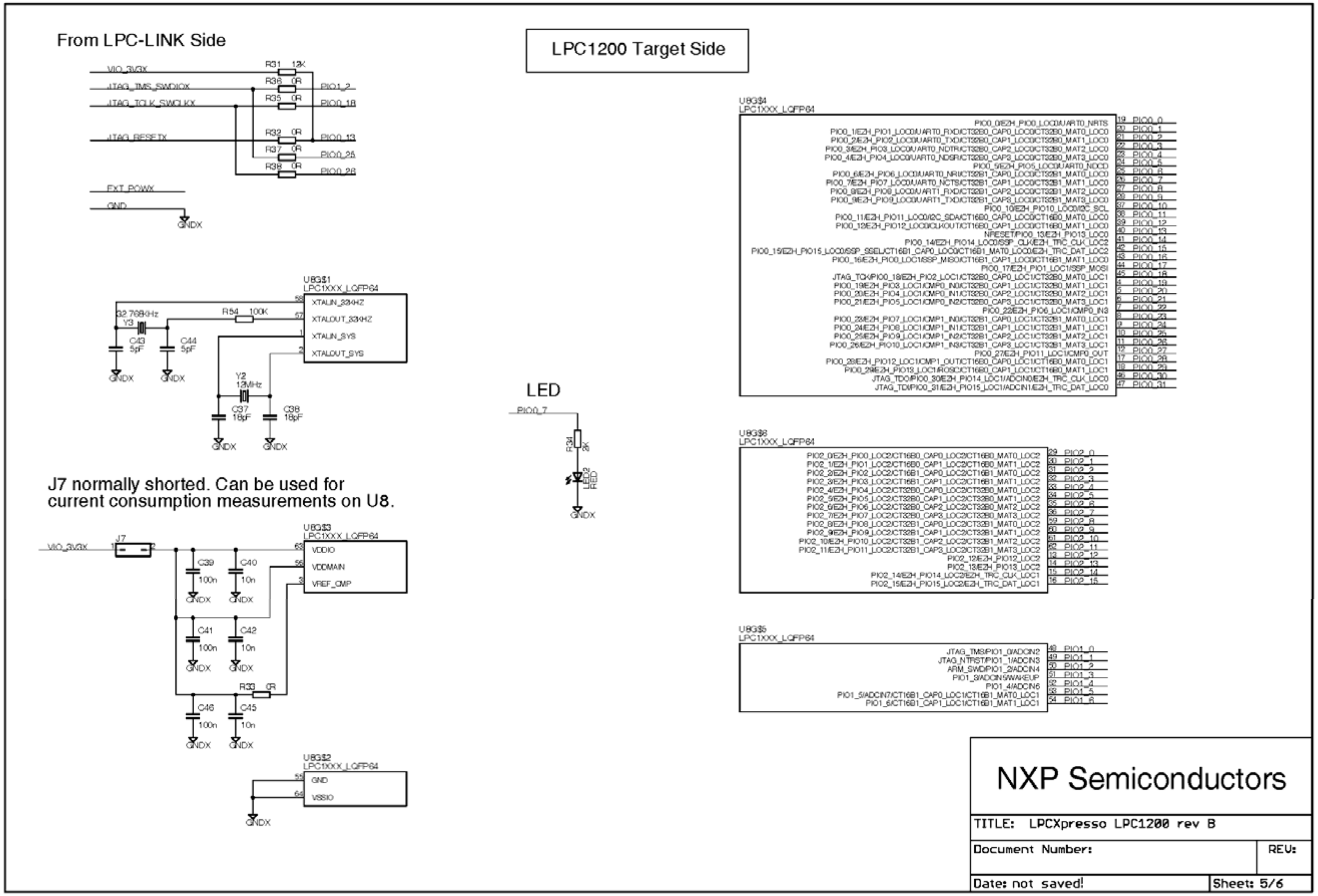
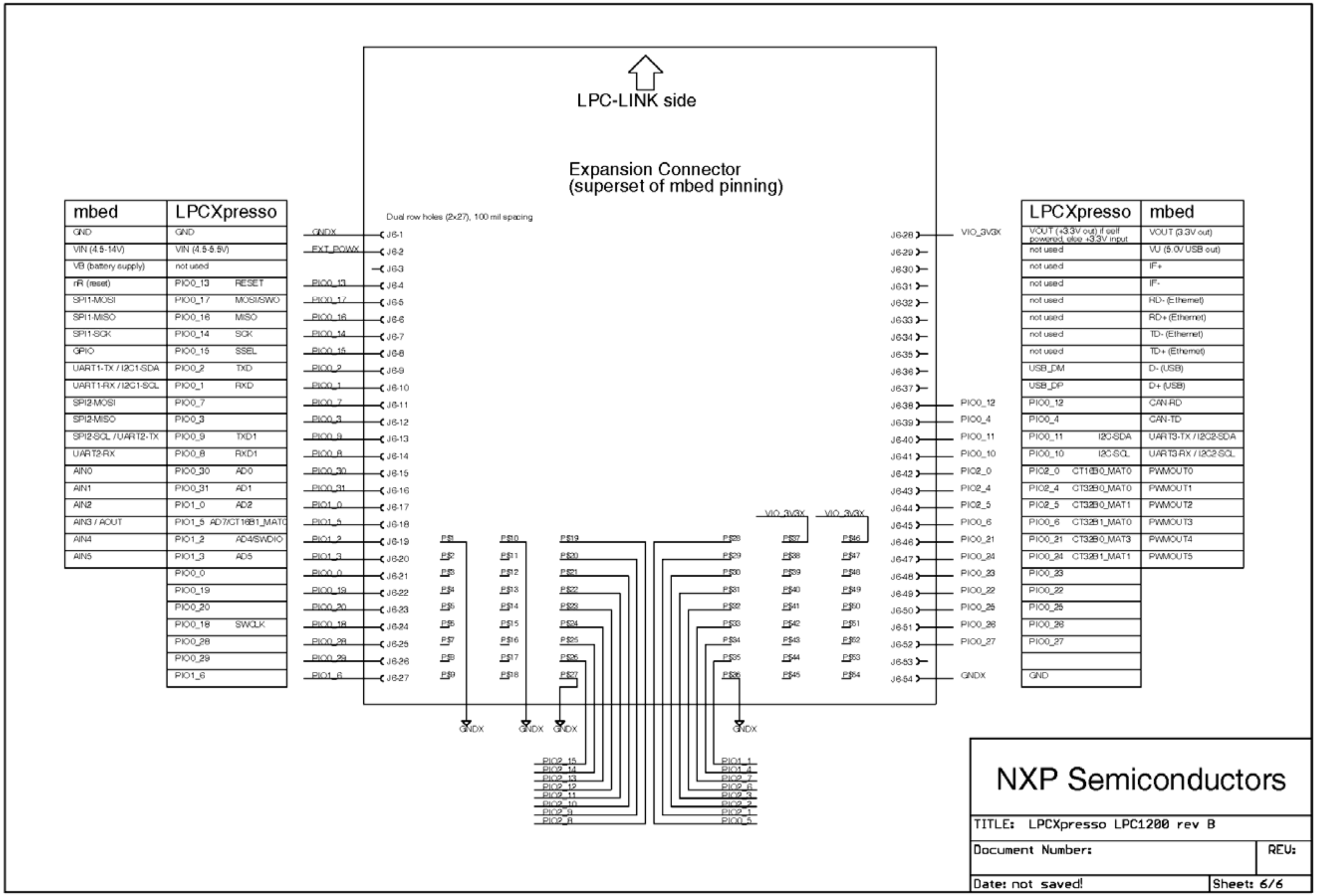


Fig 42. Schematic for the LPCXpresso LPC1200 target side, part 1



mbed	LPCXpresso		Net	Pin
GND	GND		GNDX	J6-1
VIN (4.5-14V)	VIN (4.5-5.5V)		EXT_POWX	J6-2
VB (battery supply)	not used			J6-3
nR (reset)	PIO0_13	RESET	PIO0_13	J6-4
SPI1-MOSI	PIO0_17	MOSI/SWO	PIO0_17	J6-5
SPI1-MISO	PIO0_16	MISO	PIO0_16	J6-6
SPI1-SCK	PIO0_14	SCK	PIO0_14	J6-7
GPIO	PIO0_15	SSEL	PIO0_15	J6-8
UART1-TX / I2C1-SDA	PIO0_2	TXD	PIO0_2	J6-9
UART1-RX / I2C1-SCL	PIO0_1	RXD	PIO0_1	J6-10
SPI2-MOSI	PIO0_7		PIO0_7	J6-11
SPI2-MISO	PIO0_3		PIO0_3	J6-12
SPI2-SCL / UART2-TX	PIO0_9	TXD1	PIO0_9	J6-13
UART2-RX	PIO0_8	RXD1	PIO0_8	J6-14
AIN0	PIO0_30	AD0	PIO0_30	J6-15
AIN1	PIO0_31	AD1	PIO0_31	J6-16
AIN2	PIO1_0	AD2	PIO1_0	J6-17
AIN3 / AOUT	PIO1_5	AD7/CT16B1_MATC	PIO1_5	J6-18
AIN4	PIO1_2	AD4/SWDIO	PIO1_2	J6-19
AIN5	PIO1_3	AD5	PIO1_3	J6-20
	PIO0_0		PIO0_0	J6-21
	PIO0_19		PIO0_19	J6-22
	PIO0_20		PIO0_20	J6-23
	PIO0_18	SWCLK	PIO0_18	J6-24
	PIO0_28		PIO0_28	J6-25
	PIO0_29		PIO0_29	J6-26
	PIO1_6		PIO1_6	J6-27

Pin	Net	LPCXpresso		mbed
J6-28	VIO_3V3X	VOUT (+3.3V out) if self powered, else +3.3V input		VOUT (3.3V out)
J6-29		not used		VU (5.0V USB out)
J6-30		not used		IF+
J6-31		not used		IF-
J6-32		not used		RD- (Ethernet)
J6-33		not used		RD+ (Ethernet)
J6-34		not used		TD- (Ethernet)
J6-35		not used		TD+ (Ethernet)
J6-36		USB_DM		D- (USB)
J6-37		USB_DP		D+ (USB)
J6-38	PIO0_12	PIO0_12		CAN-RD
J6-39	PIO0_4	PIO0_4		CAN-TD
J6-40	PIO0_11	PIO0_11	I2C-SDA	UART3-TX / I2C2-SDA
J6-41	PIO0_10	PIO0_10	I2C-SCL	UART3-RX / I2C2-SCL
J6-42	PIO2_0	PIO2_0	CT16B0_MAT0	PWMOUT0
J6-43	PIO2_4	PIO2_4	CT32B0_MAT0	PWMOUT1
J6-44	PIO2_5	PIO2_5	CT32B0_MAT1	PWMOUT2
J6-45	PIO0_6	PIO0_6	CT32B1_MAT0	PWMOUT3
J6-46	PIO0_21	PIO0_21	CT32B0_MAT3	PWMOUT4
J6-47	PIO0_24	PIO0_24	CT32B1_MAT1	PWMOUT5
J6-48	PIO0_23	PIO0_23		
J6-49	PIO0_22	PIO0_22		
J6-50	PIO0_25	PIO0_25		
J6-51	PIO0_26	PIO0_26		
J6-52	PIO0_27	PIO0_27		
J6-53				
J6-54	GNDX	GND		

Fig 43. Schematic for the LPCXpresso LPC1200 target side, part 2

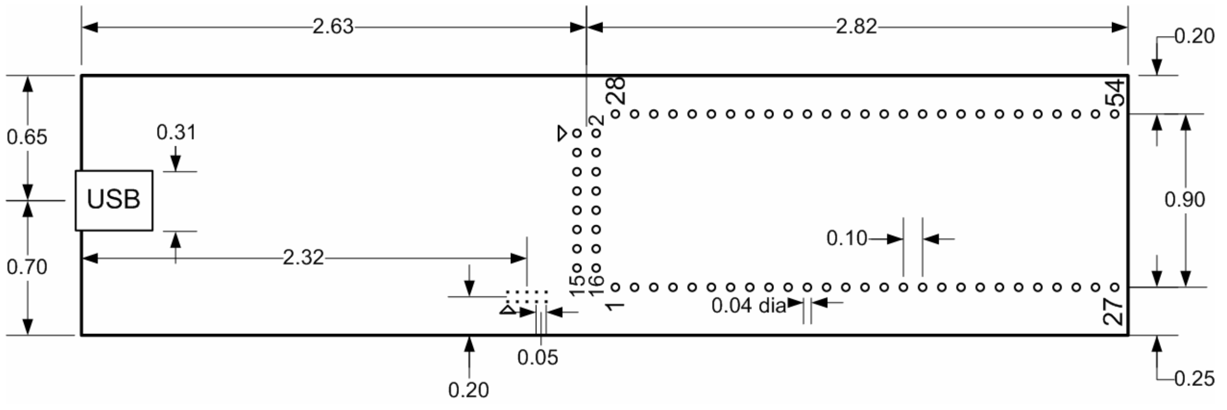


Fig 44. Dimensioned drawing of LPCXpresso LPC1227, LPC1343, LPC1114/301, LPC1114/302, LPC11C24, LPC1769, LPC1768

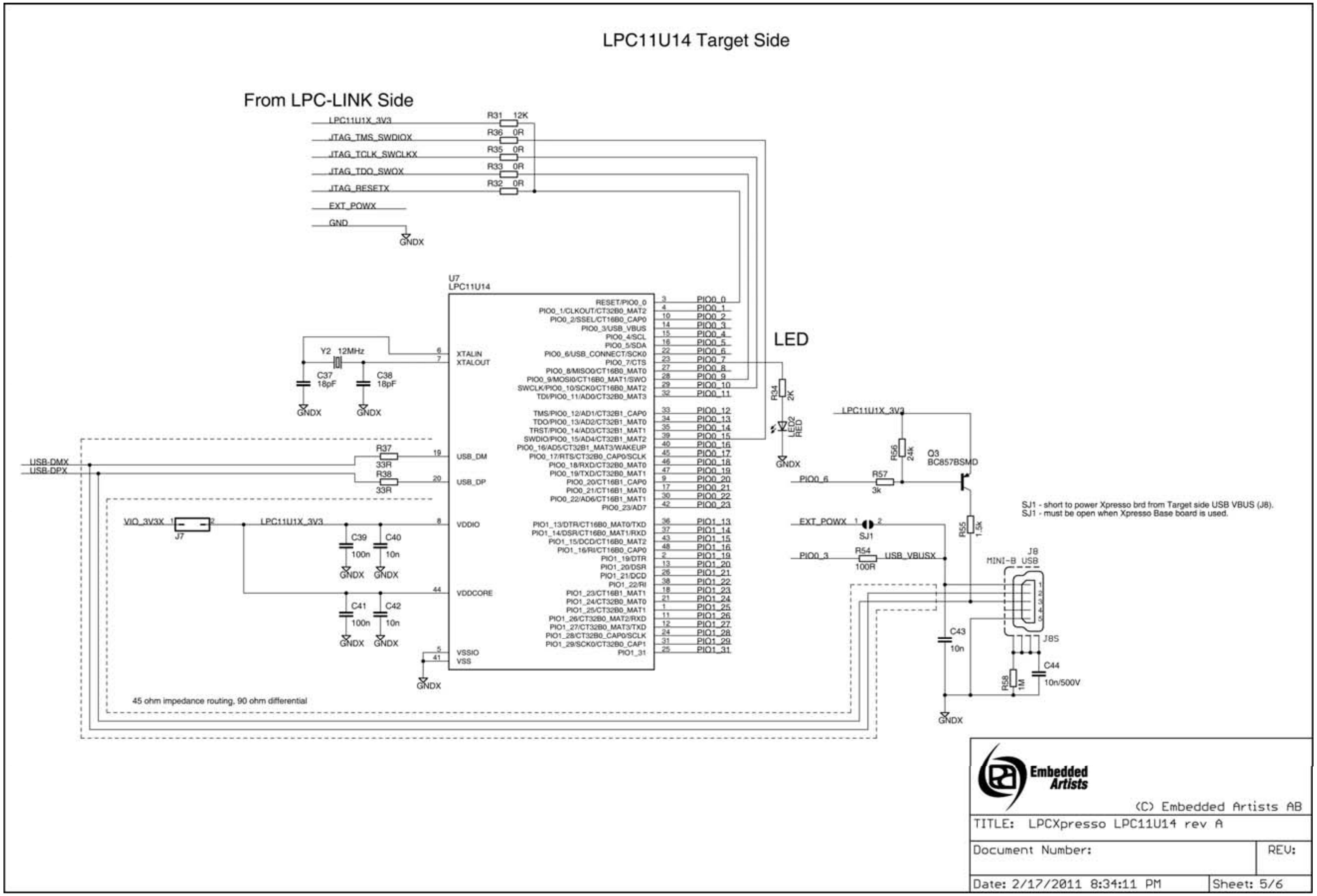
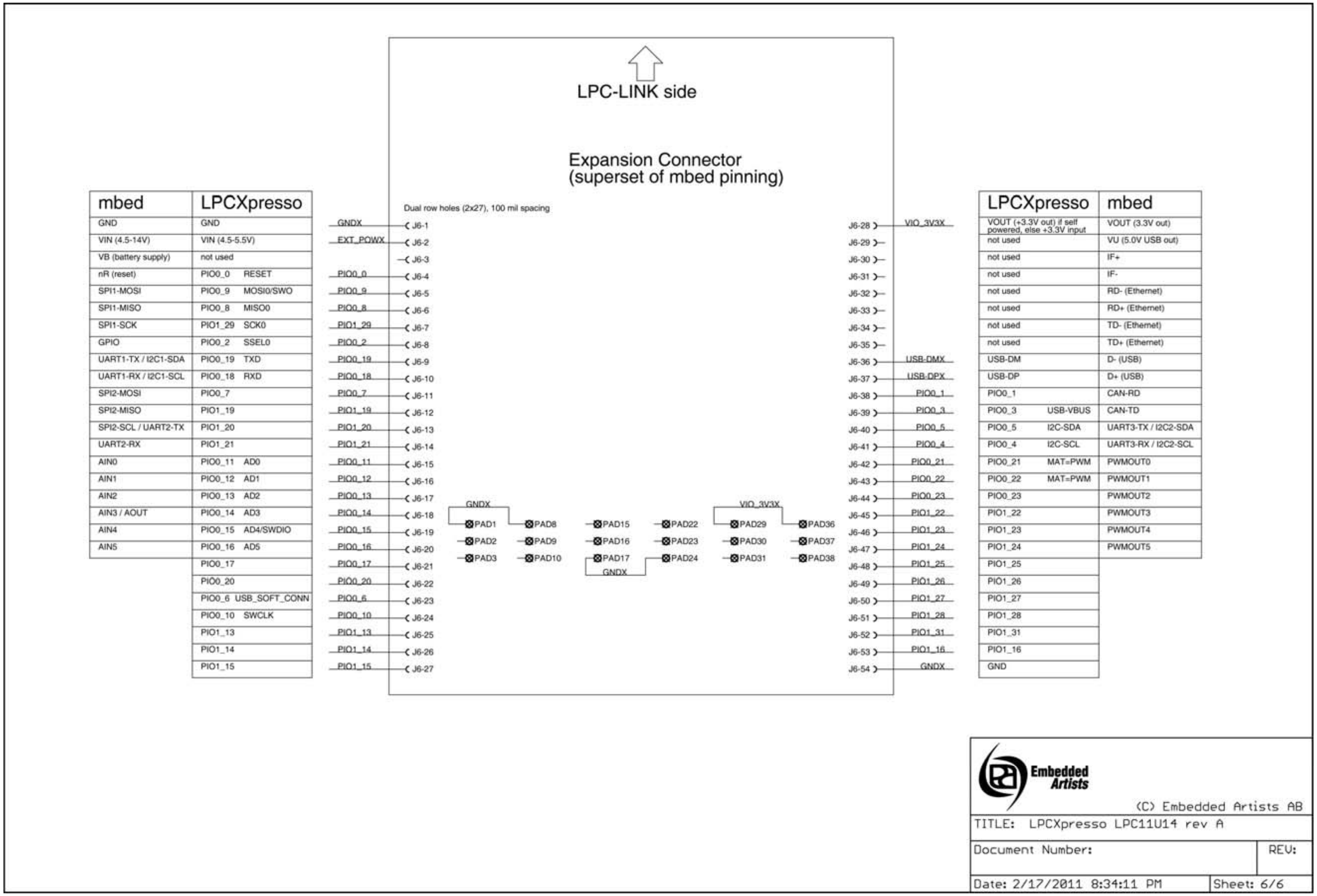


Fig 45. Schematic for the LPCXpresso LPC11U14 target side, part 1



mbed	LPCXpresso		Net	Pin
GND	GND		GNDX	J6-1
VIN (4.5-14V)	VIN (4.5-5.5V)		EXT_POWX	J6-2
VB (battery supply)	not used			J6-3
nR (reset)	PIO0_0	RESET	PIO0_0	J6-4
SPI1-MOSI	PIO0_9	MOSI0/SWO	PIO0_9	J6-5
SPI1-MISO	PIO0_8	MISO0	PIO0_8	J6-6
SPI1-SCK	PIO1_29	SCK0	PIO1_29	J6-7
GPIO	PIO0_2	SSEL0	PIO0_2	J6-8
UART1-TX / I2C1-SDA	PIO0_19	TXD	PIO0_19	J6-9
UART1-RX / I2C1-SCL	PIO0_18	RXD	PIO0_18	J6-10
SPI2-MOSI	PIO0_7		PIO0_7	J6-11
SPI2-MISO	PIO1_19		PIO1_19	J6-12
SPI2-SCL / UART2-TX	PIO1_20		PIO1_20	J6-13
UART2-RX	PIO1_21		PIO1_21	J6-14
AIN0	PIO0_11	AD0	PIO0_11	J6-15
AIN1	PIO0_12	AD1	PIO0_12	J6-16
AIN2	PIO0_13	AD2	PIO0_13	J6-17
AIN3 / AOUT	PIO0_14	AD3	PIO0_14	J6-18
AIN4	PIO0_15	AD4/SWDIO	PIO0_15	J6-19
AIN5	PIO0_16	AD5	PIO0_16	J6-20
	PIO0_17		PIO0_17	J6-21
	PIO0_20		PIO0_20	J6-22
	PIO0_6	USB_SOFT_CONN	PIO0_6	J6-23
	PIO0_10	SWCLK	PIO0_10	J6-24
	PIO1_13		PIO1_13	J6-25
	PIO1_14		PIO1_14	J6-26
	PIO1_15		PIO1_15	J6-27

Pin	Net	LPCXpresso		mbed
J6-28	VIO_3V3X	VOUT (+3.3V out) if self powered, else +3.3V input		VOUT (3.3V out)
J6-29		not used		VU (5.0V USB out)
J6-30		not used		IF+
J6-31		not used		IF-
J6-32		not used		RD- (Ethernet)
J6-33		not used		RD+ (Ethernet)
J6-34		not used		TD- (Ethernet)
J6-35		not used		TD+ (Ethernet)
J6-36	USB-DMX	USB-DM		D- (USB)
J6-37	USB-DPX	USB-DP		D+ (USB)
J6-38	PIO0_1	PIO0_1		CAN-RD
J6-39	PIO0_3	PIO0_3	USB-VBUS	CAN-TD
J6-40	PIO0_5	PIO0_5	I2C-SDA	UART3-TX / I2C2-SDA
J6-41	PIO0_4	PIO0_4	I2C-SCL	UART3-RX / I2C2-SCL
J6-42	PIO0_21	PIO0_21	MAT=PWM	PWMOUT0
J6-43	PIO0_22	PIO0_22	MAT=PWM	PWMOUT1
J6-44	PIO0_23	PIO0_23		PWMOUT2
J6-45	PIO1_22	PIO1_22		PWMOUT3
J6-46	PIO1_23	PIO1_23		PWMOUT4
J6-47	PIO1_24	PIO1_24		PWMOUT5
J6-48	PIO1_25	PIO1_25		
J6-49	PIO1_26	PIO1_26		
J6-50	PIO1_27	PIO1_27		
J6-51	PIO1_28	PIO1_28		
J6-52	PIO1_31	PIO1_31		
J6-53	PIO1_16	PIO1_16		
J6-54	GNDX	GND		

Embedded Artists

(C) Embedded Artists AB

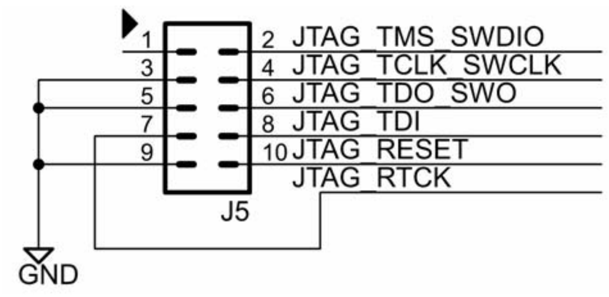
TITLE: LPCXpresso LPC11U14 rev A	
Document Number:	REV:
Date: 2/17/2011 8:34:11 PM	Sheet: 6/6

Fig 46. 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 35 to Fig 46. The LPCXpresso board was designed to be pin compatible with NXP mbed. LPCXpresso can be powered either through the debug mini-USB port, by 3.3 V applied to the board, or by 5 V 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: A dimensioned drawing of LPCXpresso can be found in Fig 44. 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.



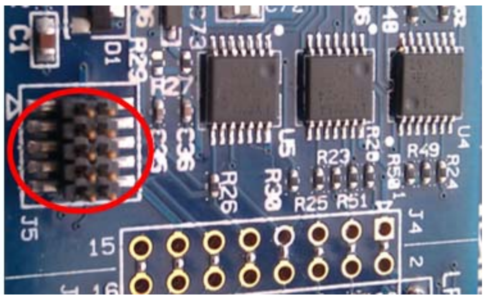


Fig 47. LPCXpresso LPC-LINK JTAG/SWO pinout

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.

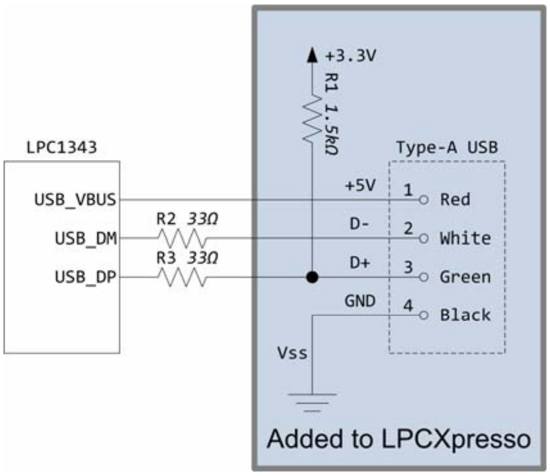


Fig 48. USB retrofit schematic

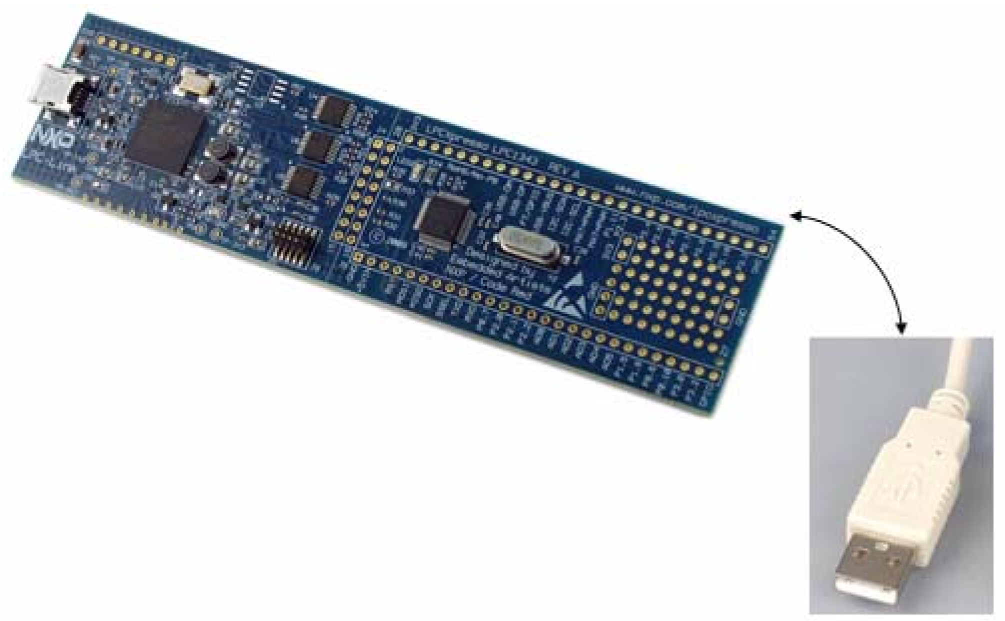


Fig 49. LPC1343 target connected to Type-A USB cable

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^{TM}

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

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] Harbison, S.P. & Steele, G.L. (2002). *C: A Reference Manual (5th Edition)*. Prentice Hall.

[8] Yiu, J. (2007). *The Definitive Guide to the ARM Cortex-M3*. Oxford, UK: Newnes.

[9] Yiu, J. (2011). *The Definitive Guide to the ARM Cortex-M0*. Oxford, UK: Newnes.

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

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

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

[13] 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. Contents

Date of release: 14 June 2011
Document identifier: LPCXpresso