

## Laboratory 02

### Introduction to LPCXpresso LPC1769

---

#### 1. Introduction

LPCXpresso is a MCU development platform ecosystem. It includes

1. **LPCXpresso Board** which composes of two parts: **the development board**, a target ARM based LPC1769 MCU board and **the LPC-Link debug probe**, an integrated IEEE1149.1 Standard JTAG (Joint Test Action Group) debugger

2. **LPCXpresso IDE** which is a software development environment. LPCXpresso is a, 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.

3. **LPCOpen** which is an extensive array of software drivers and libraries such as peripheral drivers and some meaningful examples.

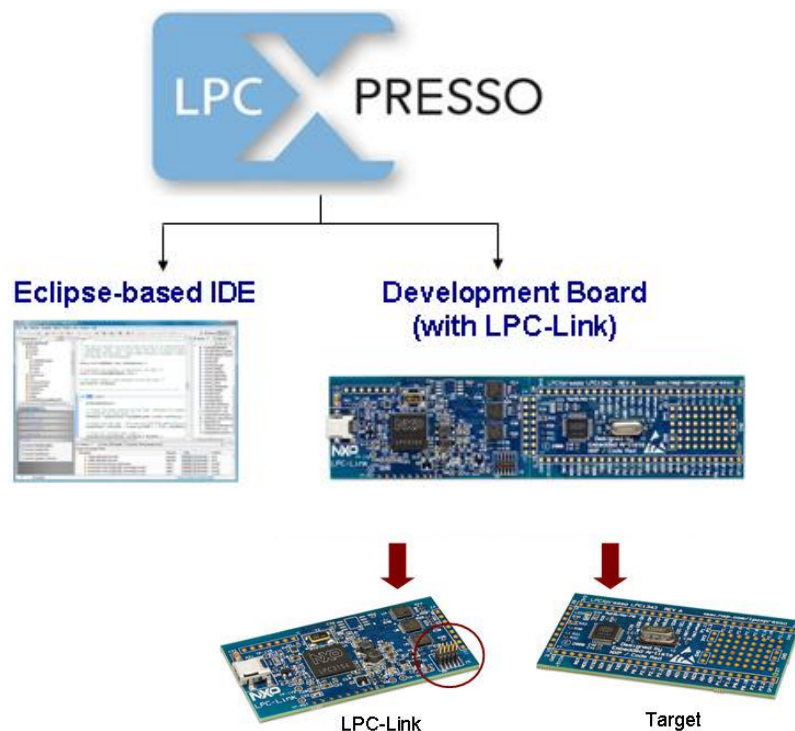


Fig. 1 LPCXpresso

## 2. Overview of LPCXpresso IDE V7

The LPCXpresso IDE is based on the Eclipse IDE. It includes the industry standard ARM GNU tools. Its debugger supports both SWD (Serial Wire Debug) and JTAG debugging, and features direct download to on-chip flash. In our Lab we use a free edition which has a **256 kB** download limit.

## 3. LPCXpresso IDE Tutorial

### 3.1 Activating LPCXpresso free edition

To activate your installation with a Free Edition license, from within LPCXpresso IDE:

1. Go to the menu entry **Help->Activate->Create Serial number and register (Free Edition)...**

- Your product's serial number will be displayed
- Copy it into the clipboard.

2. Press **OK** and a web browser will be opened on the Activations page

- If you are already logged in to the website, the serial number will be completed for you.
- If you are not logged in, you will need to login, navigate to <http://www.lpcware.com/lpcxpresso/activate>, and enter the product's serial number.

3. Press the button to Register LPCXpresso

- Your LPCXpresso Activation code will be generated and displayed.

4. Go to the menu entry **Help->Activate->Activate (Free Edition)...**

5. Enter your activation code and Press **OK**.

- This activates your product. The license type will be displayed and you will be able to use all the features of LPCXpresso, with a debug download limit of 256KB.

### 3.2 Tutorial

#### 3.2.1 Workspaces

A workspace is simply a directory that is used to store the projects you are currently working on. Each workspace can contain multiple projects, and you can have multiple workspaces on your computer. The LPCXpresso IDE can only have a single workspace open at a time, although it is possible to run multiple instances in parallel — with each instance accessing a different workspace.

If you tick the **Use this as the default and do not ask again** option, then the LPCXpresso IDE will always start up with the chosen workspace opened. Otherwise you will always be prompted to choose a workspace.

It is also possible to change workspace whilst running the LPCXpresso IDE, using

**File -> Switch Workspace**

When you first launch LPCXpresso IDE, you will be asked to select a Workspace, as shown in Figure 2

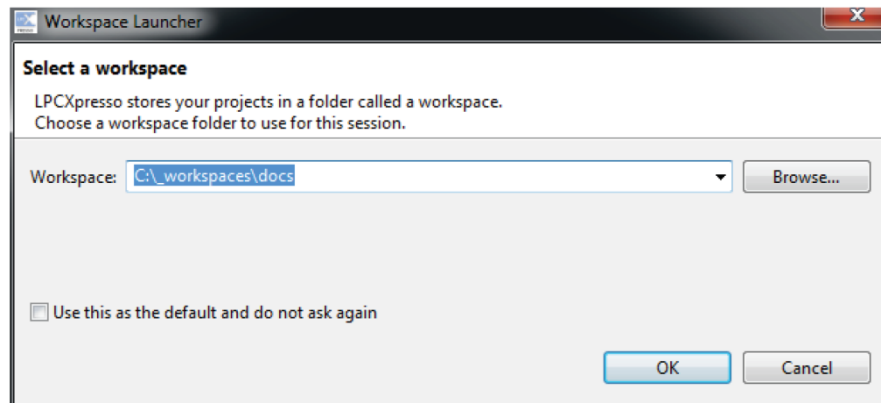


Fig. 2 Workspace Launcher

### 3.2.2 Perspectives and Views

**Perspective** is the overall layout of the main LPCXpresso IDE window. There are two modes of perspective to operate.

**Views** are sub-windows within Perspective. A View displays a particular set of data in the LPCXpresso environment. For example, this data might be source code, hex dumps, disassembly, or memory contents. Views can be opened, moved, docked, and closed, and the layout of the currently displayed Views can be saved and restored.

If a View is accidentally closed, it can be restored by selecting it from:

**Window -> Show View dialog**

### 3.2.3 Modes of Perspective

**Develop Perspective:** Both code development and debug sessions operate as shown in Figure 3

**Dual Perspective:** C/C++ perspective is used for developing and navigating around your code and the debug perspective is used when debugging your application.

**Switch between perspectives:** Using the Perspective icons in the top right of the LPCXpresso IDE window, as in Figure 4

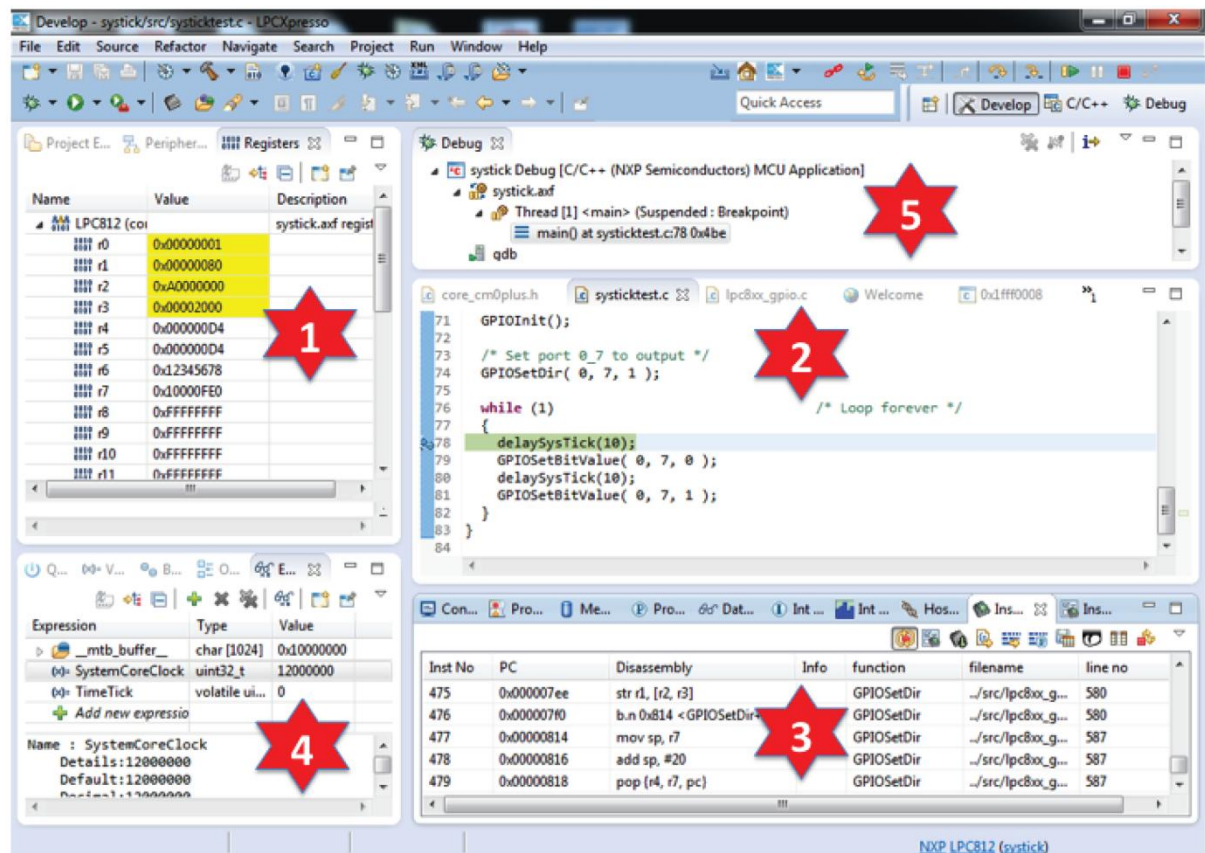


Fig. 3 Develop Perspective (whilst debugging)

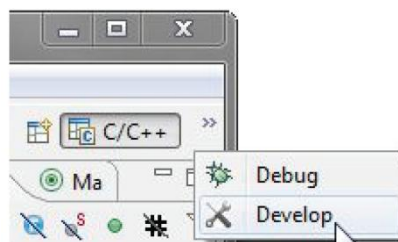


Fig 4. Perspective selection

### 3.2.4 Major components of the Develop Perspective

#### 1. Project Explorer / Peripherals / Registers Views

- The Project Explorer gives you a view of all the projects in your current 'Workspace'.
- When debugging, the Peripherals view allows you to display the registers with-in Peripherals.

- When debugging, the Registers view allows you to display the registers within the CPU of your MCU.

## 2. Editor

- The editor, which allows modification and saving of source code. When debugging, it is here that 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.

## 3. Console / Problems / Red Trace Views

- The Console View displays status information on compiling and debugging, as well as semihosted program output. The Problem View (available by changing tabs) shows all compiler errors and will allow easy navigation to the error location in the Editor View.
- Located in parallel with the Console View are the various views that make up the Red Trace functionality of LPCXpresso IDE. The Red Trace views allow you to gather and display runtime information using the SWV technology that is part of Cortex-M3/M4 based parts. In addition, for some MCUs, you can also view instruction trace data downloaded from the MCU's Embedded Trace Buffer (ETB) or Micro Trace Buffer (MTB). The example here shows instruction trace information downloaded from an LPC812's MTB.

## 4. Quick Start / Variables / Breakpoints / Expressions Views

- The **Quickstart Panel** has fast links to commonly used features. This is the best place to go to find options such as Build, Debug, and Import.
- The **Variable** view allows you to see the values of local variables.
- The 'Quickstart' view, the **Breakpoint** view allows you to see and modify currently set breakpoints.
- The 'Quickstart' view, the **Expressions** view allows you to add global variables and other expressions so that you can see and modify their values.

## 5. Debug View

- The Debug view appears when you are debugging your application. This shows you the stack trace. In the 'stopped' state, you can click on any particular function and inspect its local variables in the Variables tab (which is located parallel to the **Quickstart Panel**).

### 3.2.5 Importing and Debugging example projects

#### Software drivers and examples download

First we need to download example projects from **LPCOpen** website

In the **Quickstart Panel**

Click on **Start Here > Import project(s)**.

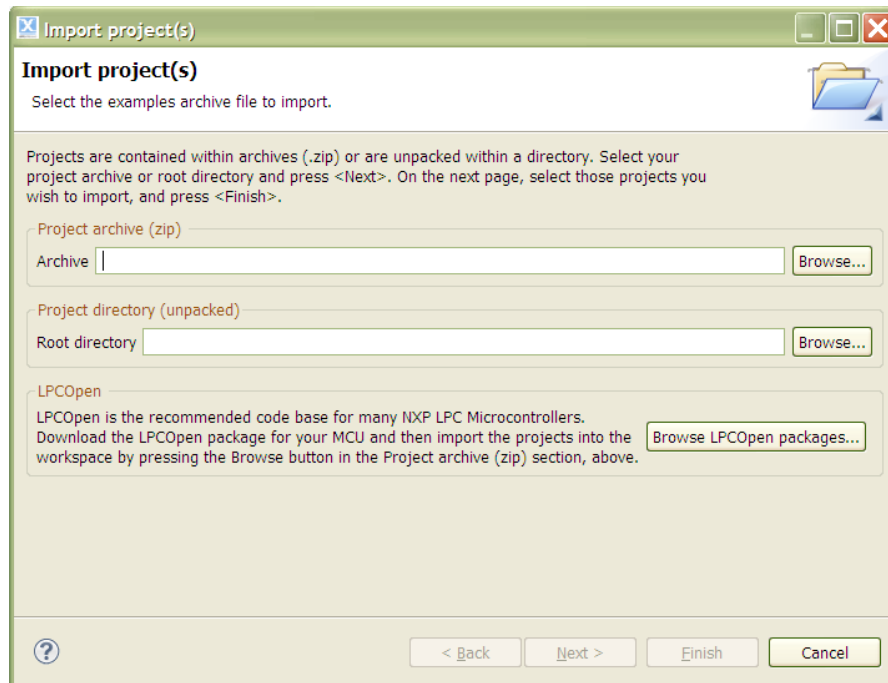


Fig. 5 Import project(s) Dialog

Click on : **Browse LPCOpen packages**

It will open a web browser link page on the NXP LPCware website

Click on : **Download LPCOpen Packages**

In the new page, select: **LPCOpen v2.xx for LPC17xx family devices**

In table, select : **LPCXpresso LPC1769 board > v2.10 (LPCXpresso v7.0.2\_102)**

You will download “**lpcopen\_2\_10\_lpcxpresso\_nxp\_lpcxpresso\_1769.zip**”

### 3.2.6 Importing examples for the LPCXpresso 1769

Once the package has downloaded, return to the Import Project(s) dialog and click on the **Browse** button next to **Project archive (zip)** and locate the **lpcopen\_2\_10\_lpcxpresso\_nxp\_lpcxpresso\_1769.zip** package archive previously downloaded.

Select the archive, click **Open** and then click **Next**.

You will then be presented with a **list of projects** within the archive, as shown in Figure 6.

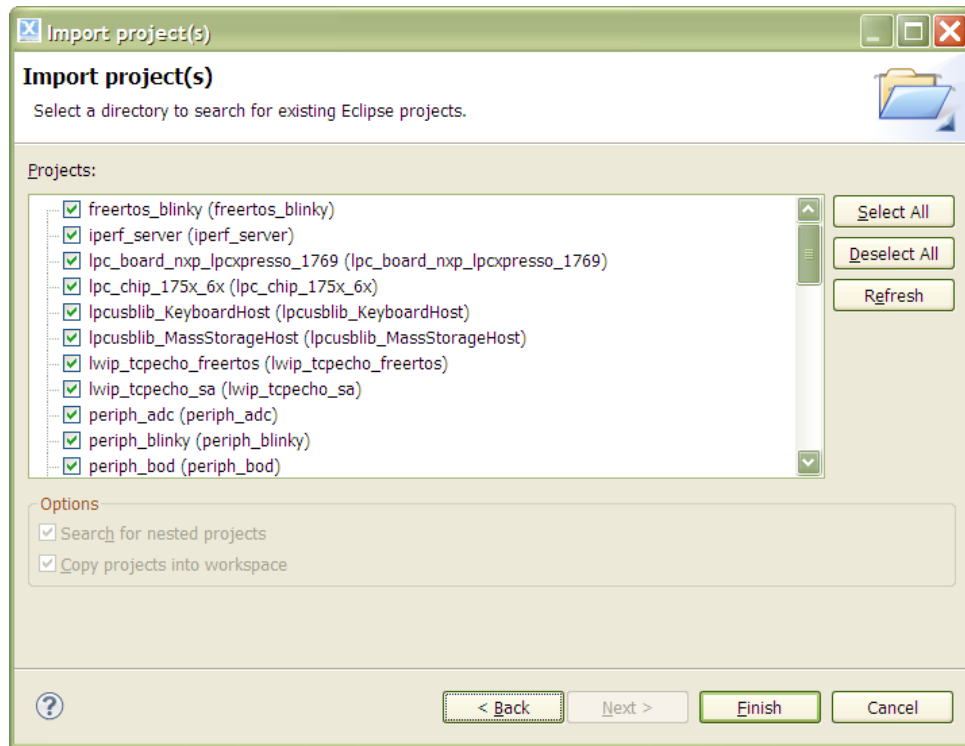


Fig. 6. A list of project to import

Select the projects you want to import (for this lab **select all**) and then click **Finish**. The examples will be imported into your workspace.

### 3.2.7 Building projects

There are two ways to build the projects.

1. Build a single project: In Project Explorer select a project and in Quickstart Panel click **Build** 'the corresponding project name'
2. Build all project: In **Quickstart Panel** click **Build all projects**. It will build all the imported projects.

#### Build configurations

There are two configurations to chose

**[Debug]** : it compiles code with optimization disable.

**[Release]**: it compiles code with optimizing for minimum code size

In this lab, it is set to debug configuration

### 3.2.8 Debugging a project

When debug is started, the program is automatically downloaded to the target and is programmed into FLASH memory, a default breakpoint is set on the first instruction in main(), the application is started (by simulating a processor reset), and code is executed until the default breakpoint is hit.

#### Example 1. Debugging a project named `periph_blinky`

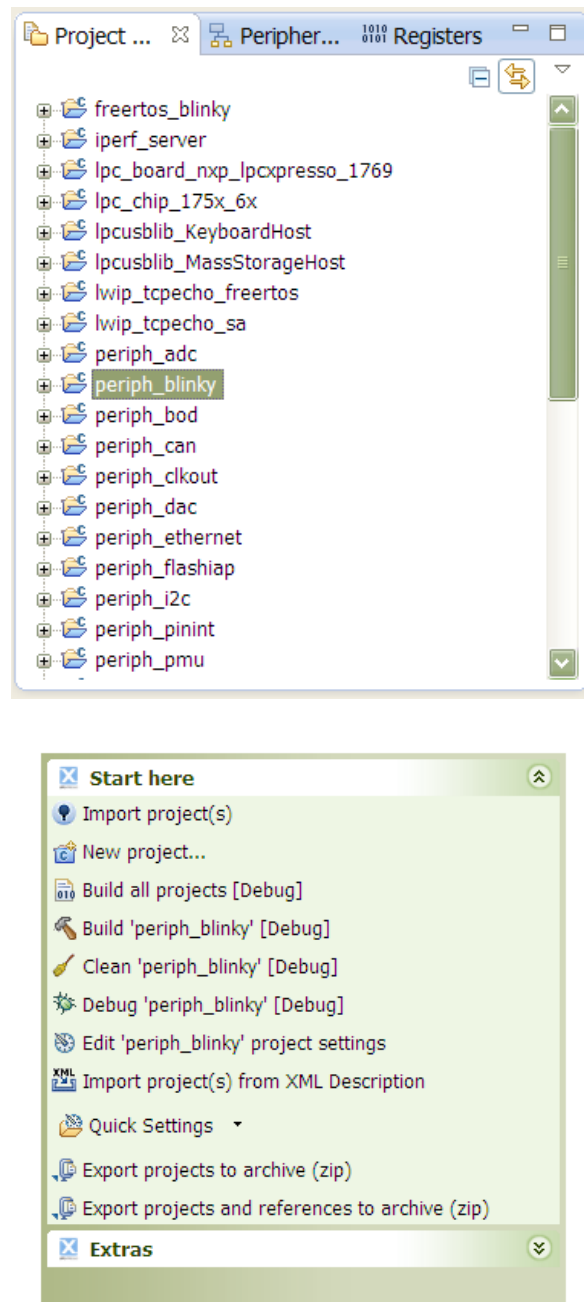


Fig.7



To start debugging **periph\_blinky** on your target, simply highlight the project in the **Project Explorer** and then in the **Quickstart Panel** click on **Start Here** and select **Debug 'periph\_blinky'**, as in Figure 7. The LPCXpresso IDE will first build and then start debugging the application. Click on **OK** to continue with the download and debug of the 'Debug' build of your project.

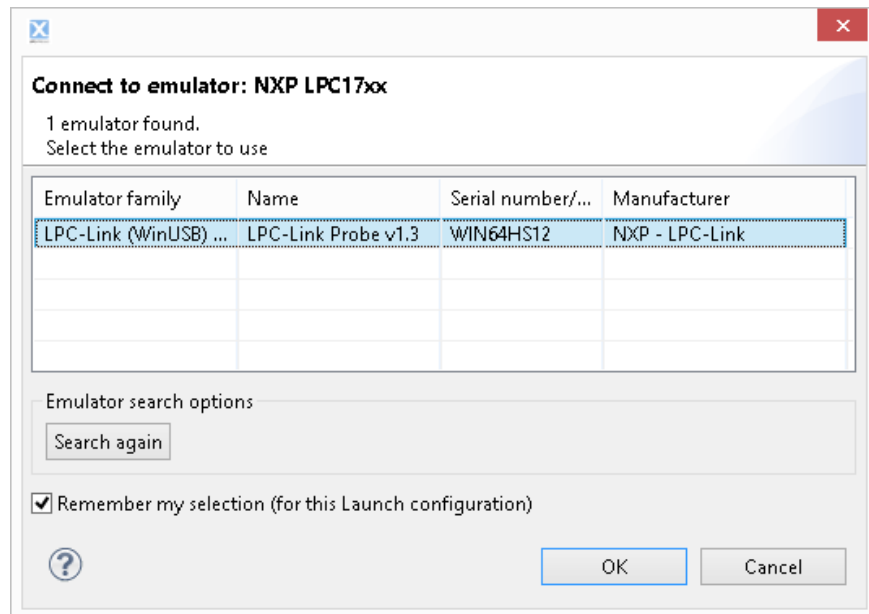


Fig 7. Connect to Emulator dialog box

In first run, LPCExpresso will show the configuration dialog box to select the board emulator to use as in Figure 7. Click on **OK** to continue, the debugger will download your object code to the board. LED1 will flash during the operation. When it finishes, the Debugger will start the break point on 'main()' function of your code (Figure 8). You can press **F8** to resume the code or **CTRL+F2** to terminate your code. (Figure 9).

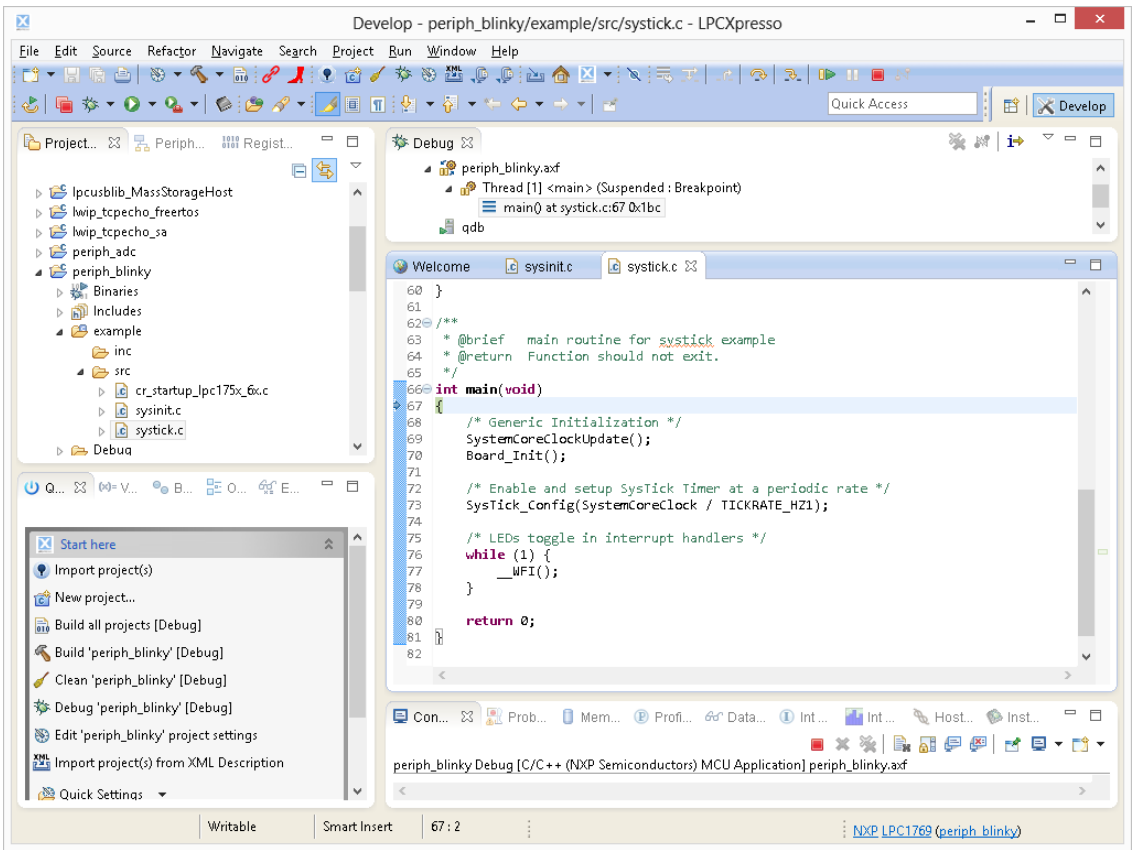


Fig 8. The Debug View







	Run the program.
	Step Over C/C++ line.
	Step into a function.
	Stop the debugger.
	Pause Execution of the running program.
	Restart the debug session

Fig 9. Debug Buttons

You may also enter debug mode by clicking the bug icon on the top LPCXpresso tool-bar. (Fig 10)



Fig 10. 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. (Figure 11)

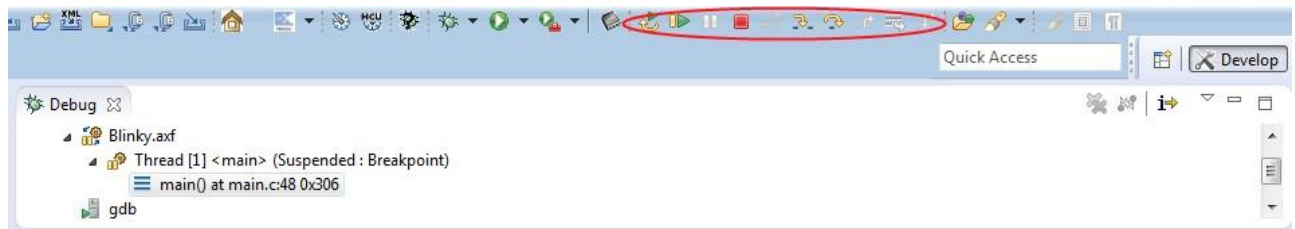


Fig 11. Debug toolbar

## 4. Experiment

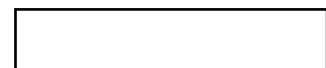
Modified the example code to turn on LED the Target Board and then turn it off without using SysTick Timer.

```
int main(void)
{
    /* Generic Initialization */
    SystemCoreClockUpdate();
    Board_Init();

    /* Enable and setup SysTick Timer at a periodic rate */
    SysTick_Config(SystemCoreClock / TICKRATE_HZ1);

    /* LEDs toggle in interrupt handlers */
    while (1) {
        __WFI();
    }

    return 0;
}
```



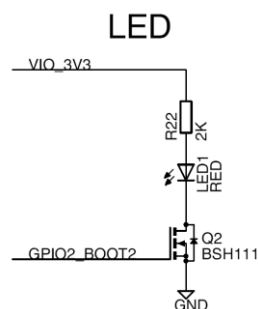
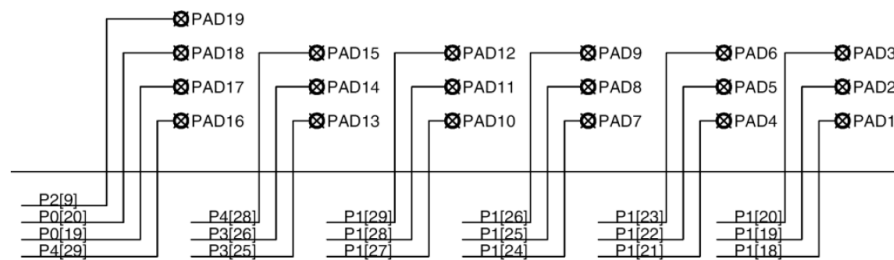
## 5. References

1. NXP, LPCXpresso v7 User Guide, Rev. 7.3 — 30 June, 2014

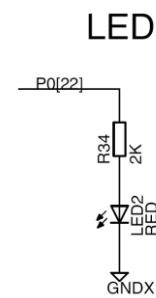
## 6. Appendix

Pin map of LPCXpresso 1769 Target Board

GND	— GNDX	↔ J6-1	J6-28	— VIO_3V3X	VOUT (+3.3V out) if self powered, else +3.3V input
VIN (4.5-5.5V)	— EXT_POWX	↔ J6-2	J6-29		not used
VB (battery supply)	— VB	↔ J6-3	J6-30		not used
RESET_N	— RESET_N	↔ J6-4	J6-31		not used
P0.9 MOSI1	— P0[9]	↔ J6-5	J6-32	— RD-	RD-
P0.8 MISO1	— P0[8]	↔ J6-6	J6-33	— RD+	RD+
P0.7 SCK1	— P0[7]	↔ J6-7	J6-34	— TD-	TD-
P0.6 SSEL1	— P0[6]	↔ J6-8	J6-35	— TD+	TD+
P0.0 TXD3/SDA1	— P0[0]	↔ J6-9	J6-36	— USB-D-	USB-D-
P0.1 RXD3/SCL1	— P0[1]	↔ J6-10	J6-37	— USB-D+	USB-D+
P0.18 MOSI0	— P0[18]	↔ J6-11	J6-38	— P0[4]	P0.4 CAN_RX2
P0.17 MISO0	— P0[17]	↔ J6-12	J6-39	— P0[5]	P0.5 CAN_TX2
P0.15 TXD1/SCK0	— P0[15]	↔ J6-13	J6-40	— P0[10]	P0.10 TXD2/SDA2
P0.16 RXD1/SSEL0	— P0[16]	↔ J6-14	J6-41	— P0[11]	P0.11 RXD2/SCL2
P0.23 AD0.0	— P0[23]	↔ J6-15	J6-42	— P2[0]	P2.0 PWM1.1
P0.24 AD0.1	— P0[24]	↔ J6-16	J6-43	— P2[1]	P2.1 PWM1.2
P0.25 AD0.2	— P0[25]	↔ J6-17	J6-44	— P2[2]	P2.2 PWM1.3
P0.26 AD0.3/AOUT	— P0[26]	↔ J6-18	J6-45	— P2[3]	P2.3 PWM1.4
P1.30 AD0.4	— P1[30]	↔ J6-19	J6-46	— P2[4]	P2.4 PWM1.5
P1.31 AD0.5	— P1[31]	↔ J6-20	J6-47	— P2[5]	P2.5 PWM1.6
P0.2	— P0[2]	↔ J6-21	J6-48	— P2[6]	P2.6
P0.3	— P0[3]	↔ J6-22	J6-49	— P2[7]	P2.7
P0.21	— P0[21]	↔ J6-23	J6-50	— P2[8]	P2.8
P0.22	— P0[22]	↔ J6-24	J6-51	— P2[10]	P2.10
P0.27	— P0[27]	↔ J6-25	J6-52	— P2[11]	P2.11
P0.28	— P0[28]	↔ J6-26	J6-53	— P2[12]	P2.12
P2.13	— P2[13]	↔ J6-27	J6-54	— GNDX	GND



LED1 on LPC-Link Board



LED2 on Target Board