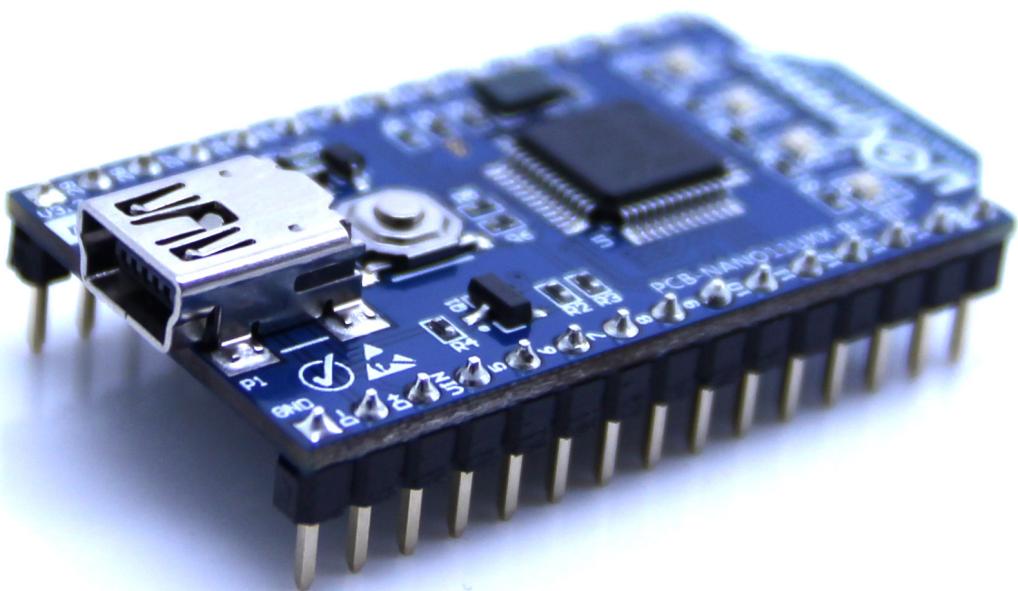


Getting Started with uCXpresso.BLE



2014/3/14
v1.0.1

History:

2014/3/14 Add Overview

2014/3/12 Add “Create A New Project”, “Serial Terminal & Debug”

2014/3/11 First Edition

Overview

First of all, an espresso is necessary,



and the uCXpresso.BLE framework have to work on the LPCXpresso IDE, you need to install the LPCXpresso IDE in your system first. The LPCXpresso IDE are available for Windows, Linux and Mac OS/X, also you can download the LPCXpresso IDE from www.lpcware.com with a free license after register. About LPCXpresso IDE, please refer to

<http://www.lpcware.com/lpcxpresso/download>

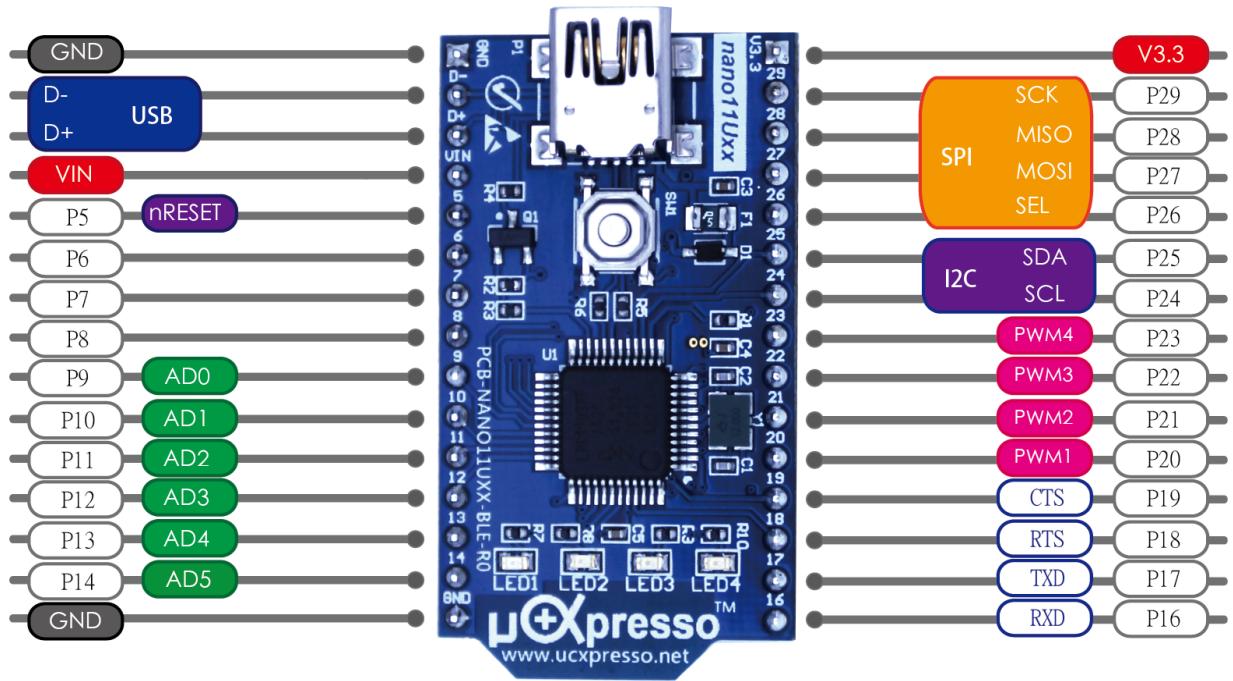
The uCXpresso.BLE framework is a RTOS C++ framework that port for the ARM Cortex-M0 MCU and BLE (Bluetooth Low Energy), and provides many features such as Peripherals (ADC, PWM, I2C, SPI, PIO, Interrupt, Timer....), BLE Services (Serial, Proximity, Heart Rate Meter...) and Multi-Thread (RTOS Thread, Semaphore, Mutex, MailBox...).

Easy to use with C++ framework:

Ex. A LED on the pin 19 of nano11U37

```
CPin led(P19);           // connect led object to P19 pin
led.output();             // set P19 as an output pin
led = HIGH;               // set P19 to HIGH (turn on LED)
```

Multiple Pin Definition:



Ex. Set P20 as an input pin

```
CPin key(P20);           // connect key object to P20
key.input();              // set key as an input pin with internal Pull-Up.
if ( key==LOW ) {        // check key PIN LEVEL
    ...
}
```

Ex. Set P20 as a PWM output

```
CPwm::period(0.02);      // set global PWM period time to 20ms
CPwm servo(PWM1);        // connect servo object to PWM1 (P20)
servo.enable();            // enable the servo PWM to output
servo.dutyCycle(0.8);     // set servo duty-cycle to 80%
servo = 0.005;            // set servo PWM period to 5ms
```

Multi-Thread

uCXpresso.BLE framework also base on an advanced Real-Time operating system (RTOS) that called the FreeRTOS. The multi thread will simplify your design, coding and maintenance.

Final,

let uCXpresso.BLE C++ Framework to shorten the developer's time , improves the qualities and enjoys the development.

1. Upgrade Your LPCXpresso IDE

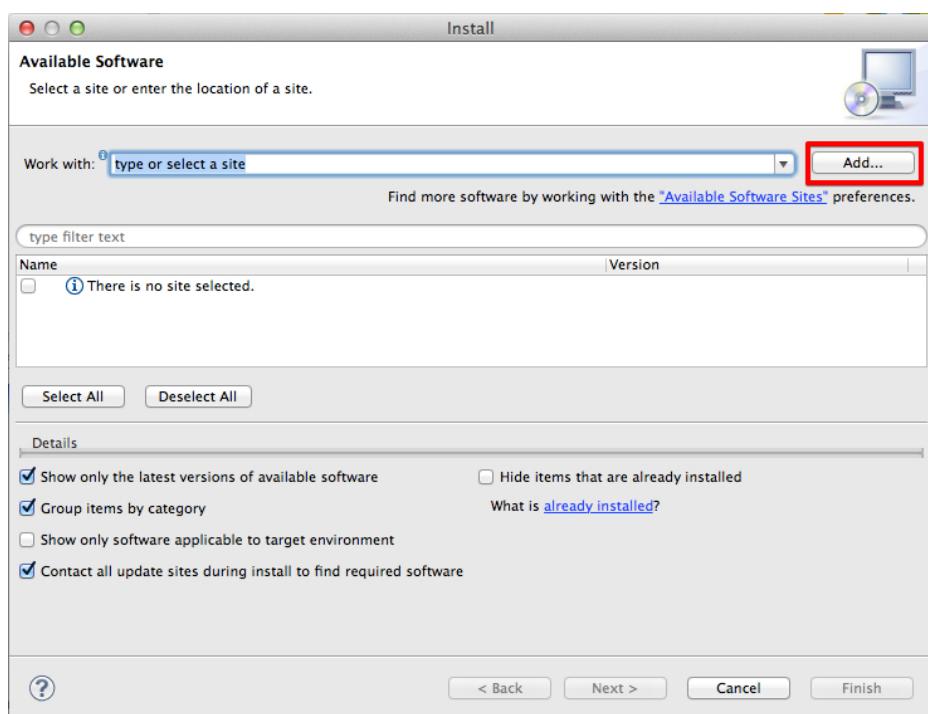
1.1. Increase VM memory for Windows System

- a. Close LPCXpresso if executing.
- b. Open c:/nxp/LPCXpresso_x.x/lpcxpresso/lpcxpresso.ini
- c. Modify MaxPermSize to 512

1.2 Install new software (Eclipse Plug-in)

LPCXpresso Main Menu > Help > Install New Software...

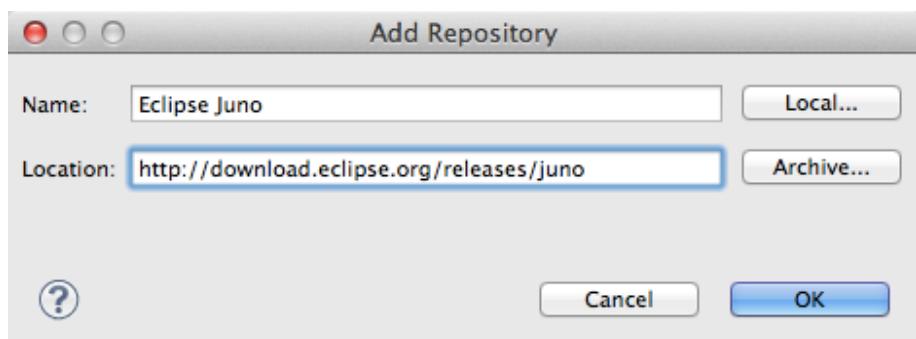
Click [Add]



1.3 Install Eclipse Plug-In

Name: Eclipse Juno

Location: <http://download.eclipse.org/releases/juno>



In General Purpose Tools section, select “Marketplace Client”

Name	Version
ACTF Visualization Extension for PDT Feature	1.0.2.R201302130546
ACTF Visualization Extension for WST Feature	1.0.2.R201302130546
ACTF Visualization Feature	1.0.2.R201302130546
ACTF Visualization SDK Feature	1.0.2.R201302130546
ChangeLog Management Tools	2.8.0.201302051708
ChangeLog Management Tools for C/C++	2.8.0.201302051708
ChangeLog Management Tools for Java	2.8.0.201302051708
Dynamic Languages Toolkit - Core Frameworks	4.0.0.201206120848
Dynamic Languages Toolkit - Remote Development Support	4.0.0.201206120848
Eclipse Plug-in Development Environment	3.8.2.v20130116-091538-7c7wfj0FFt6Z...
Local Terminal (Incubation)	0.2.200.201301070737
m2e - Maven Integration for Eclipse	1.3.0.20130129-0926
m2e - slf4j over logback logging (Optional)	1.3.0.20130129-0926
Marketplace Client	1.1.1.I20110907-0947
Memory Analyzer	1.2.1.201211051250

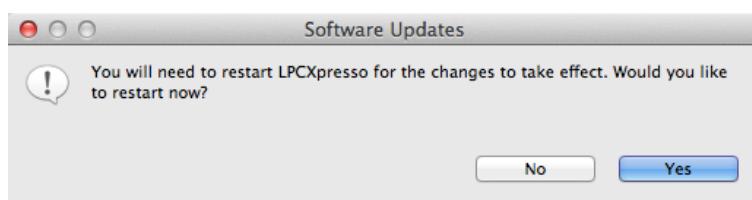
2 items selected

In Mobile and Device Development section, select “Target Management Terminal”

Name	Version
Mobile and Device Development	
C/C++ GCC Cross Compiler Support	1.1.0.201302132326
C/C++ GDB Hardware Debugging	7.0.0.201302132326
C/C++ Memory View Enhancements	2.2.0.201302132326
C/C++ Remote Launch	6.0.0.201302132326
Remote System Explorer End-User Runtime	3.4.1.201302122026
Remote System Explorer User Actions	1.1.400.201301240456
 Target Management Terminal	3.3.2.201301080822
TCF C/C++ Debugger	1.0.1.201212201401
TCF Remote System Explorer add-in	1.0.0.201212201401
TCF Target Explorer	1.0.0.201212201401
Modeling	
Programming Languages	
SOA Development	
Testing	
Web XML, Java EE and OSGi Enterprise Development	

2 items selected

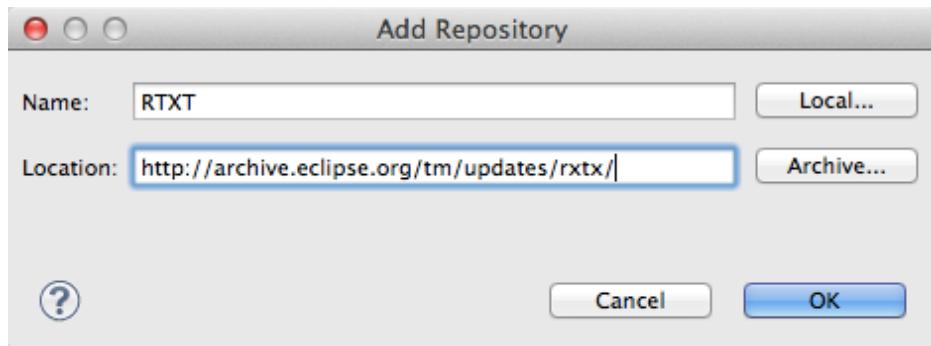
Click [Next] to start the installation. In progress, you need to [Confirm >] the Selected Features, and [accept] the licenses of software, and click [OK] to confirm any security-warning message, click [Yes] to restart and finish the installation if need.



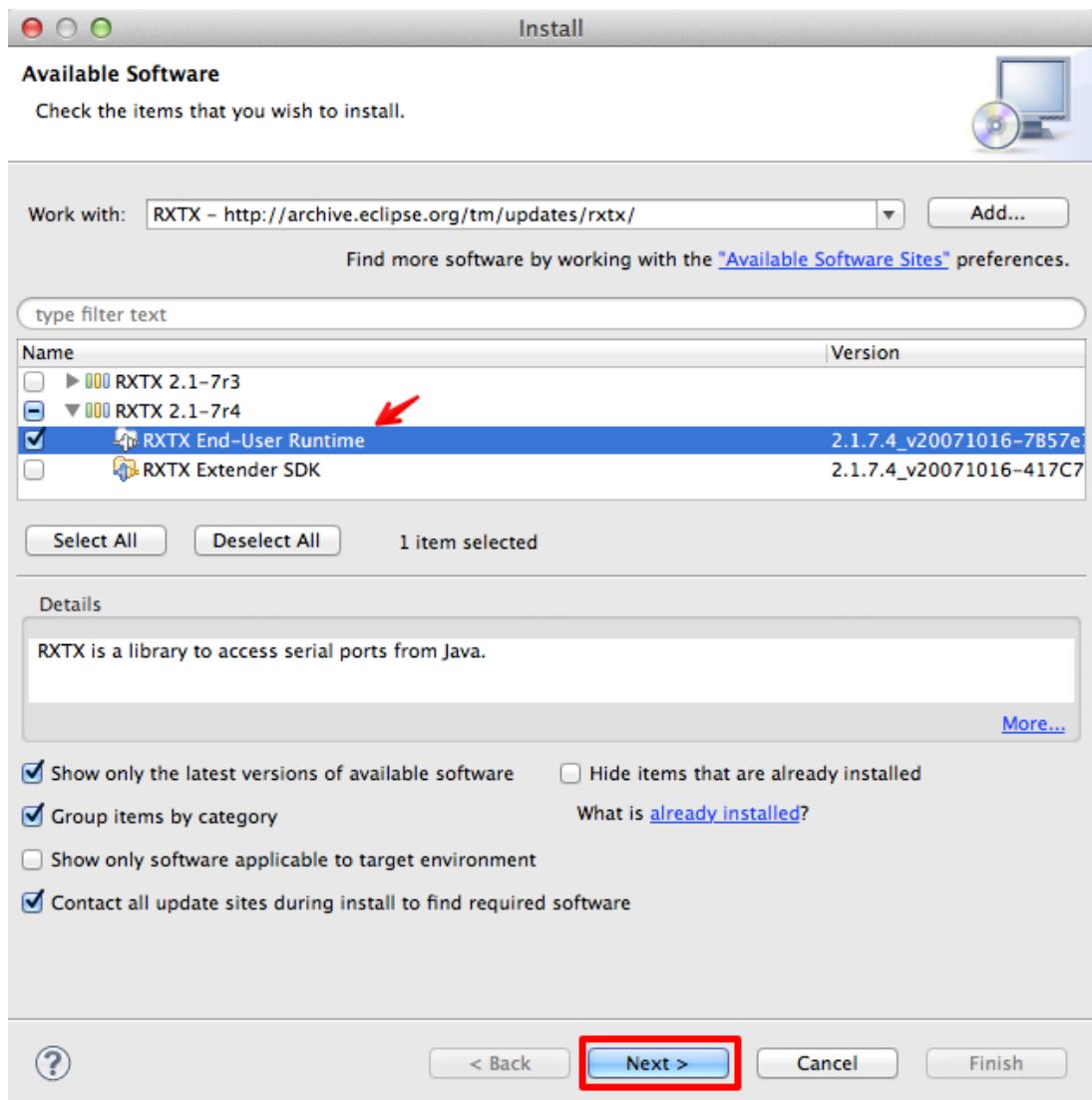
1.4. Install RXTX Plug-in

Name: RXTX

Location: <http://archive.eclipse.org/tm/updates/rxtx/>



Select latest version “RXTX End-User Runtime”



Click [Next] to start the installation.

Remark:

For Mac OS/X user, you have to follow below steps to enable the USB CDC virtual COM. Port:

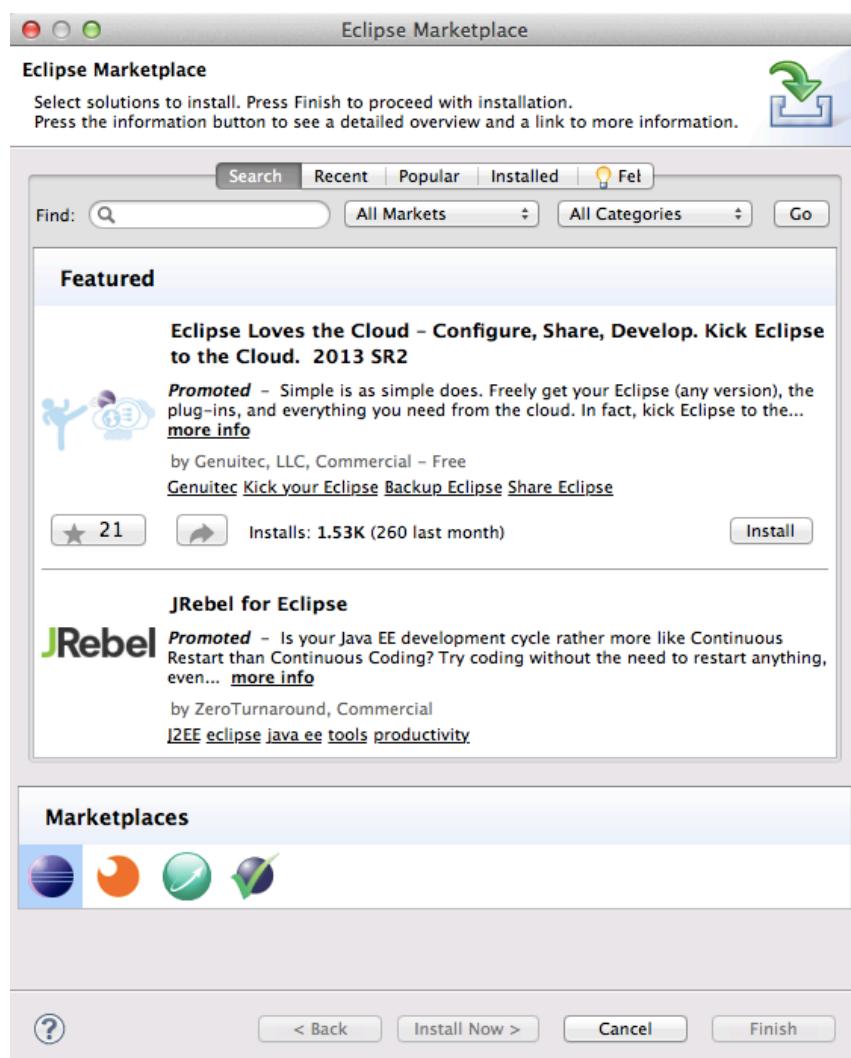
1. Open Terminal App of OS/X
2. Type :
sudo mkdir /var/lock
sudo chmod a+rwx /var/lock

In Windows System, The USB CDC driver can be download from below link:

for NANO11Uxx : http://www.embeda.com.tw/tw/wp-content/uploads/2014/01/nano11Uxx_usb_driver.zip
(unzip and copy INF file to desktop, and USBCOM port driver direct to the INF file.)

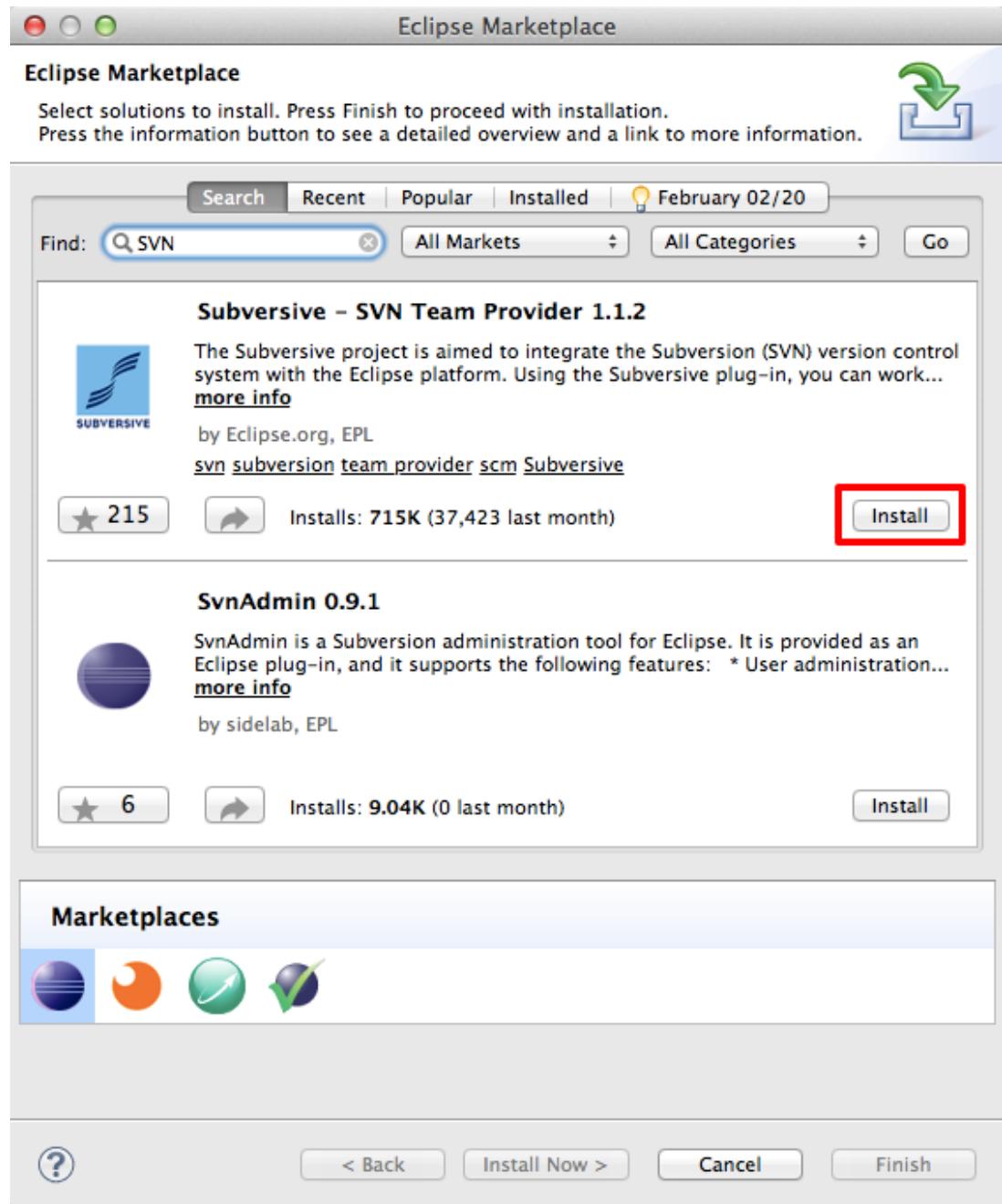
1.5. Install new software from Eclipse Marketplace

Click LPCXpresso Main Menu > Help > Marketplace...

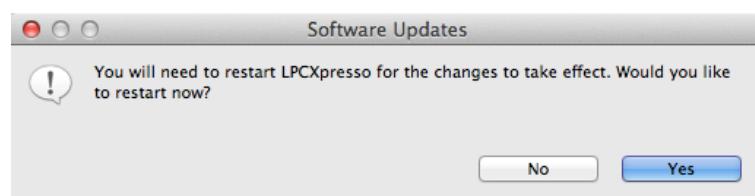


1.6 Install Subversion – SVN Team Provider (Plug-in)

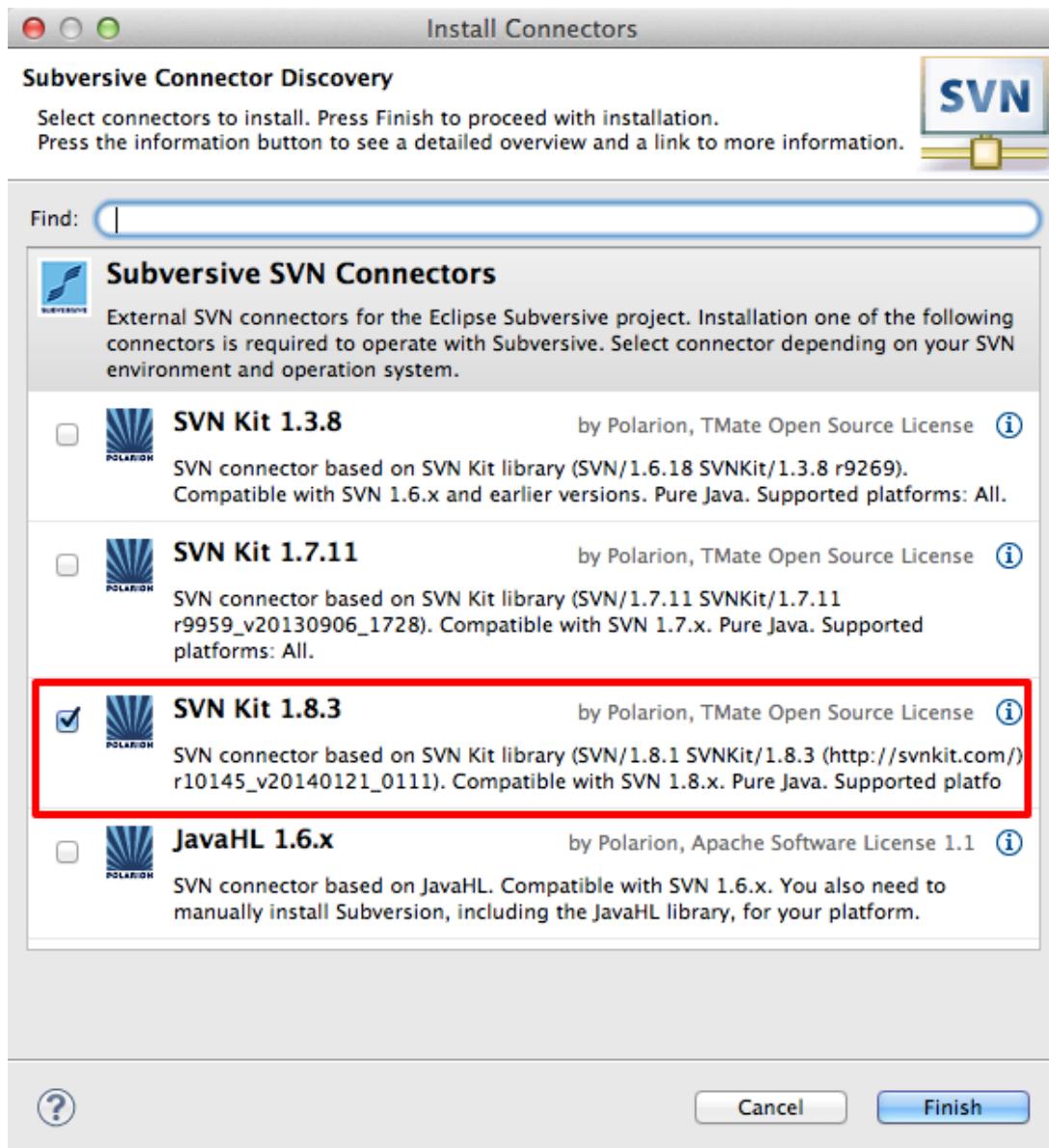
In Find, input “SVN” and click [Go]



Click [Install] to start the installation. In progress, you need to [Confirm >] the Selected Features, and [accept] the licenses of software, and click [OK] to confirm any security-warning message. Click [Yes] to restart and finish the installation if need.



After restart, you may need to install the SVN connector kit:

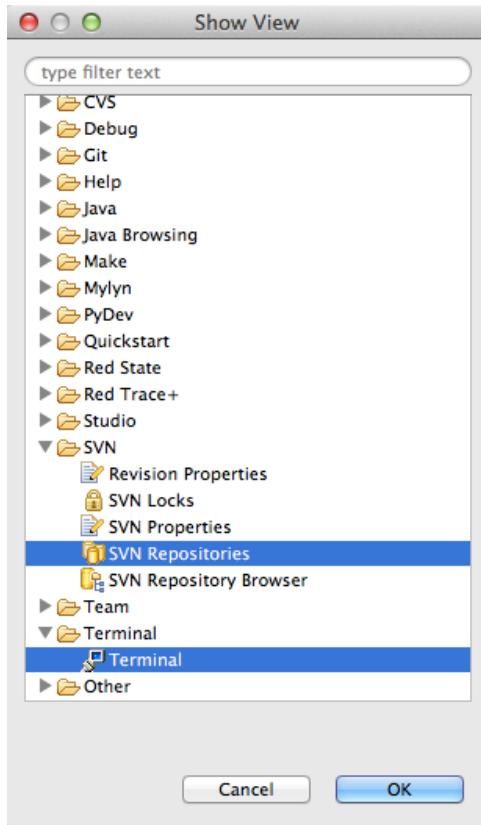


Select the latest version “SVN Kit x.x.x”, and click [Finish] to start the installation. Confirm and click [Next], and [accept] the licenses of software, and click [OK] to confirm any security-warning message. Click [Yes] to restart and finish the installation if need.



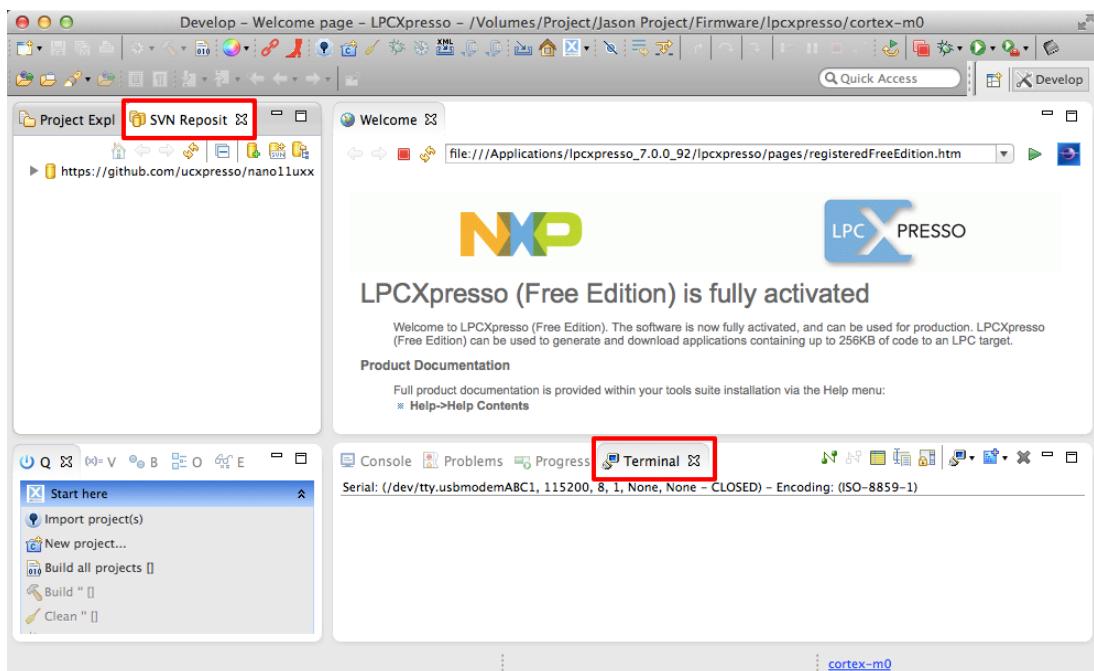
1-7. Show the SVN and Terminal Views

Click LPCXpresso Main Menu > Window > Show View > Other...



Select the “SVN Repositories” and “Terminal”, click [OK]

Move the “SVN Repositories” Tab to “Project Explorer” right side (optional)

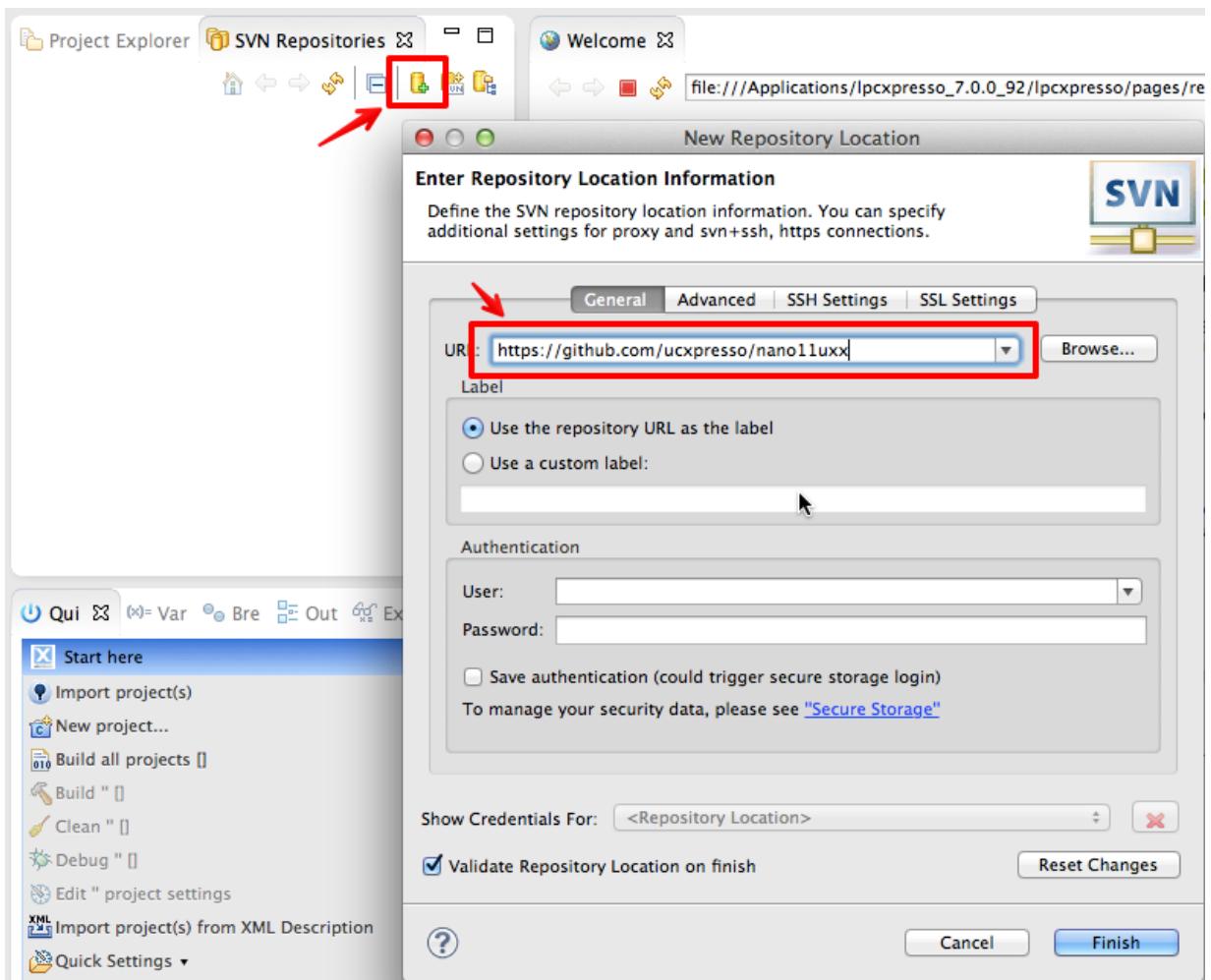


2. Install uCXpresso.BLE Framework

2.1 Using GitHub Repositories



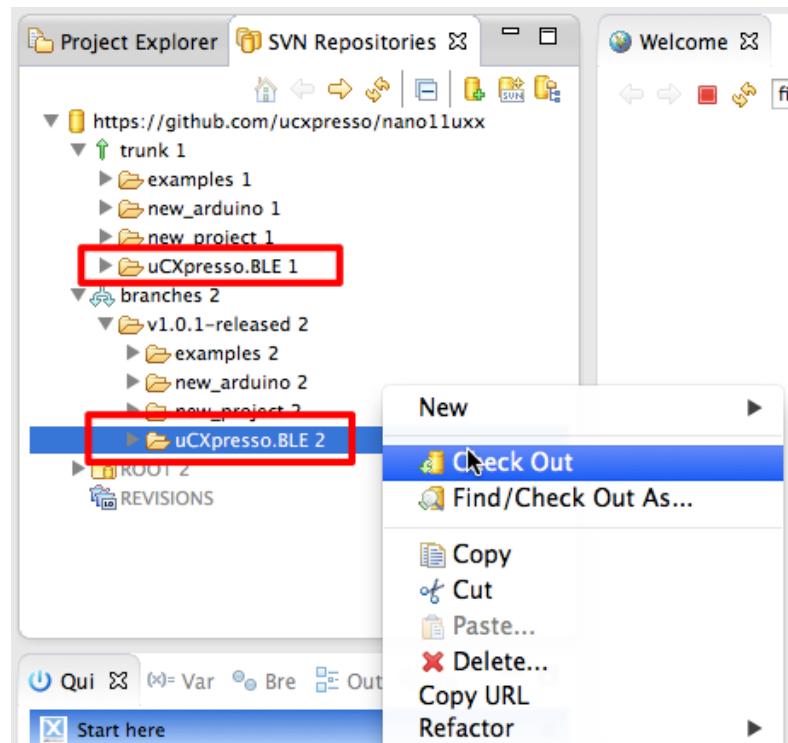
In “SVN Repositories” Tab, Click “New Repository Location”,



and URL Input: `https://github.com/ucxpresso/nano11uxx`

2.2 Check Out uCXpresso.BLE framework

Expand the uCXpresso.BLE SVN repository

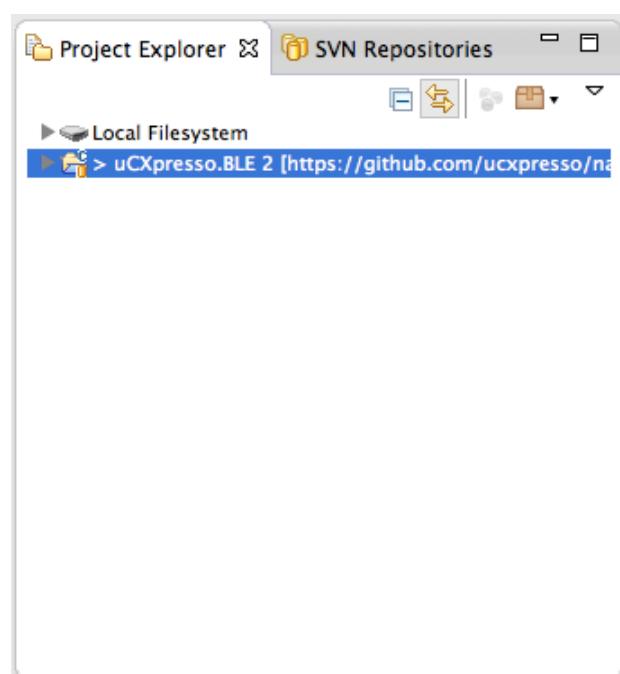


trunk: uCXpresso.BLE RC version

branches: All released version

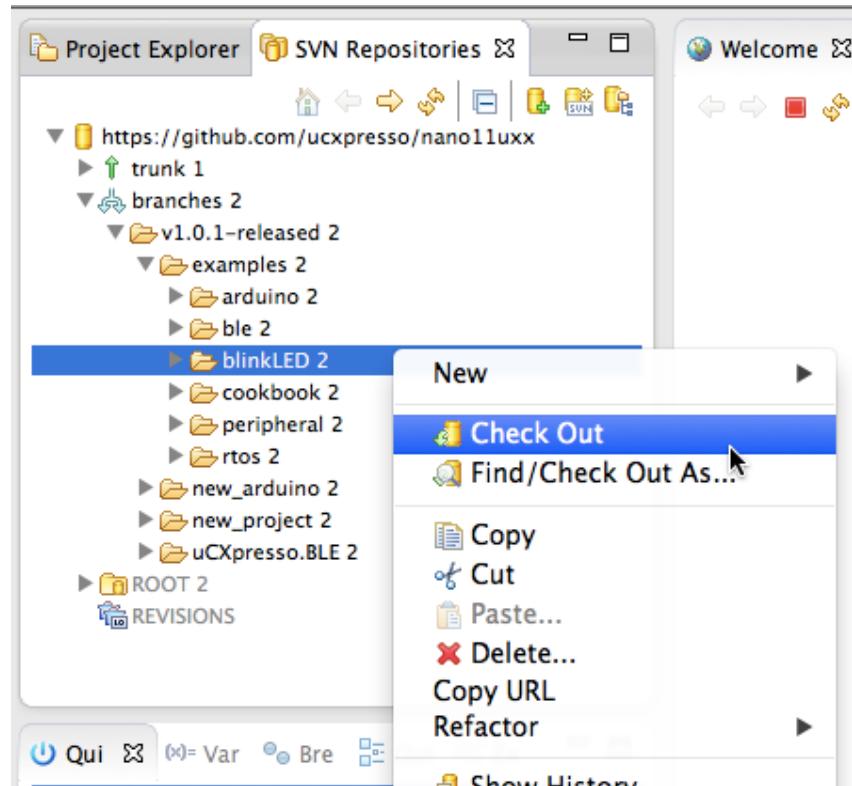
Select “uCXpresso.BLE” folder, and click right button of mouse, and click “Check Out” in drop down menu.

The uCXpresso.BLE framework show in your workspace after check out.



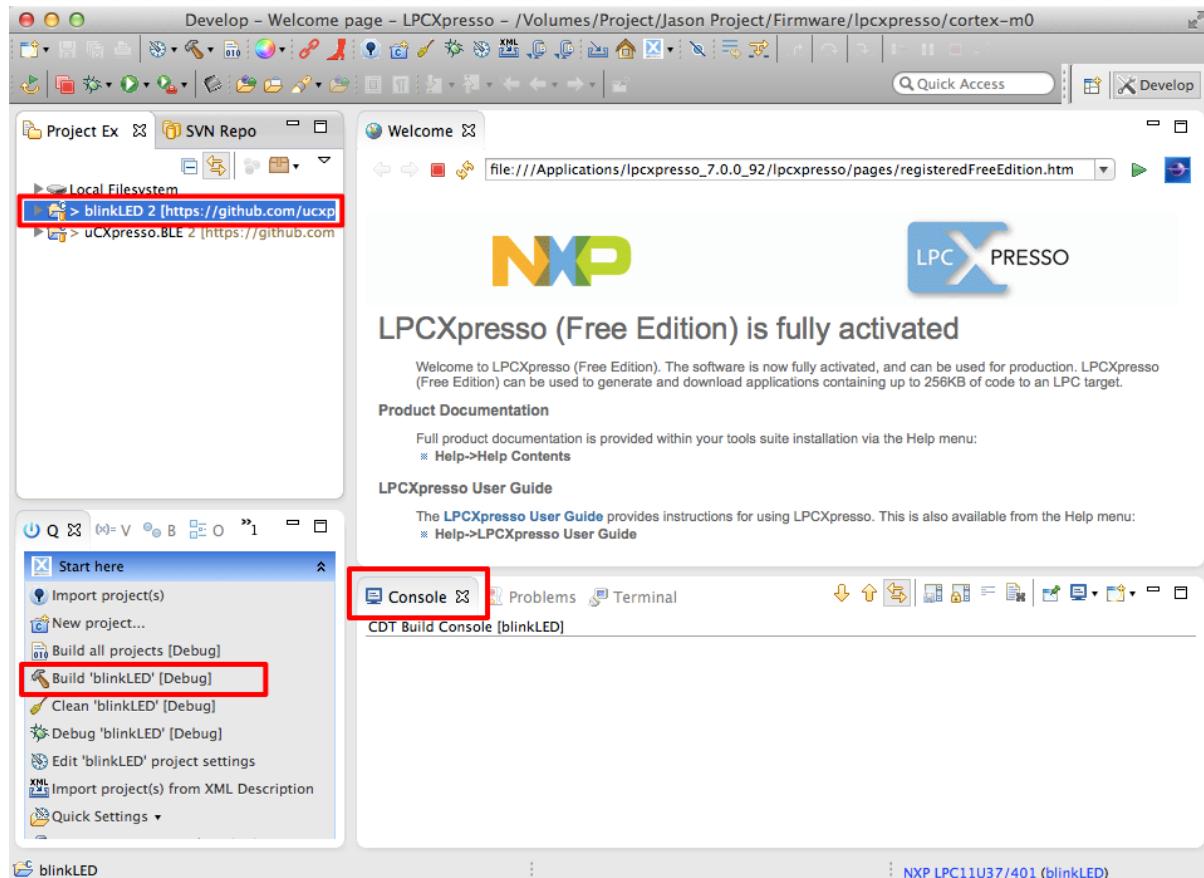
2.3 Check Out example “blinkLED”

In SVN Repositories Tab, select the “blinkLED” folder in “/examples”, then Check Out it.



2.4 Build the “blinkLED” example project

1. Click “blinkLED” Project on Project Explorer View.
2. Click “Console” View
3. Click “Build ‘blinkLED’ [Debug] on Quick Start View.



In Console View, the compiler will create a “blinkLED.bin” image file.

```

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size blinkLED.axf; arm-none-eabi-objcopy -O binary blinkLED.axf blinkLED.bin ;
    text     data     bss     dec      hex filename
  25272       20    7268   32560    7f30 blinkLED.axf
Created checksum 0xffff7d15 at offset 0x1c in file blinkLED.bin
12:00:54 Build Finished (took 363ms)

```

Note:

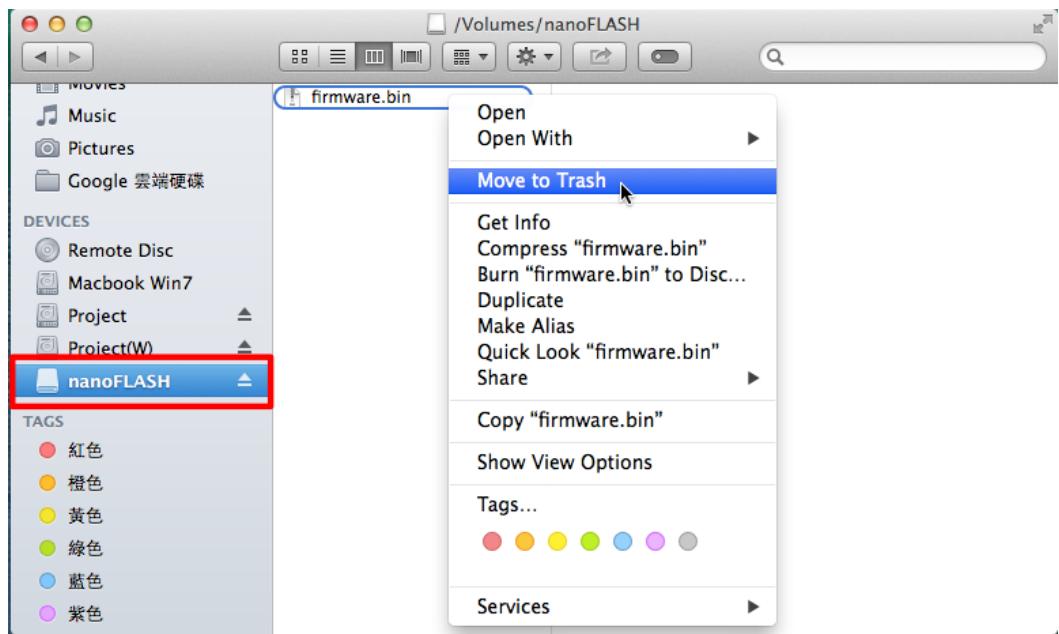
code size = text + data = 25292 bytes

ram size = data + bss = 7288 bytes

3. Download Image File

3.1 Erase Flash Code Memory

First, you need to erase the flash memory for new binary image. Click the “ISP” button on the nano11U37-BLE. In Finder (or File Manager) will creates an USB Disk and called “nanoFLASH”



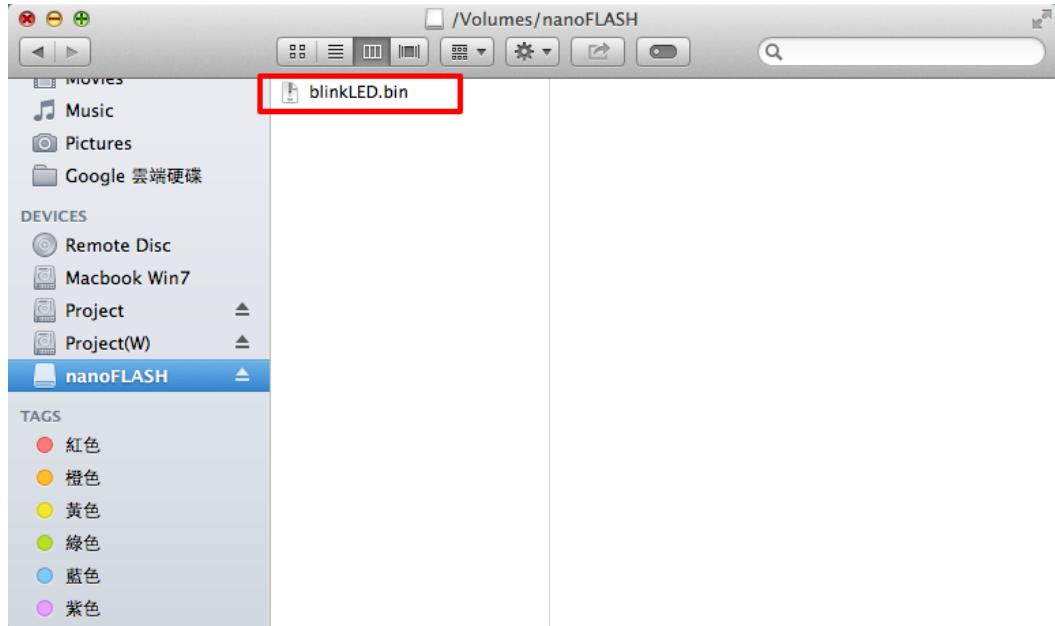
Remove the “firmware.bin” and move to trash.

Note:

In ISP (bootloader) mode: The LED1, LED2, LED3 and LED4 are blink together.

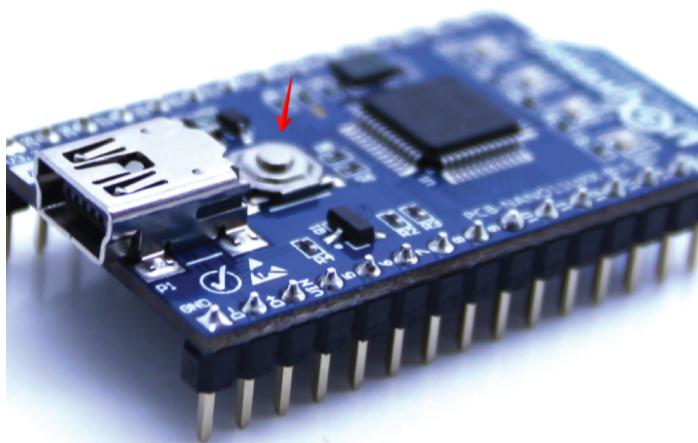
3.2 Program Flash Code Memory

Copy the “blinkLED.bin” from “blinkLED\Debug” folder to nanoFLASH



Eject the “nanoFLASH” disk and click the “ISP” button of NANO11U37-BLE again to execute your binary code. (In App Mode)

Note:

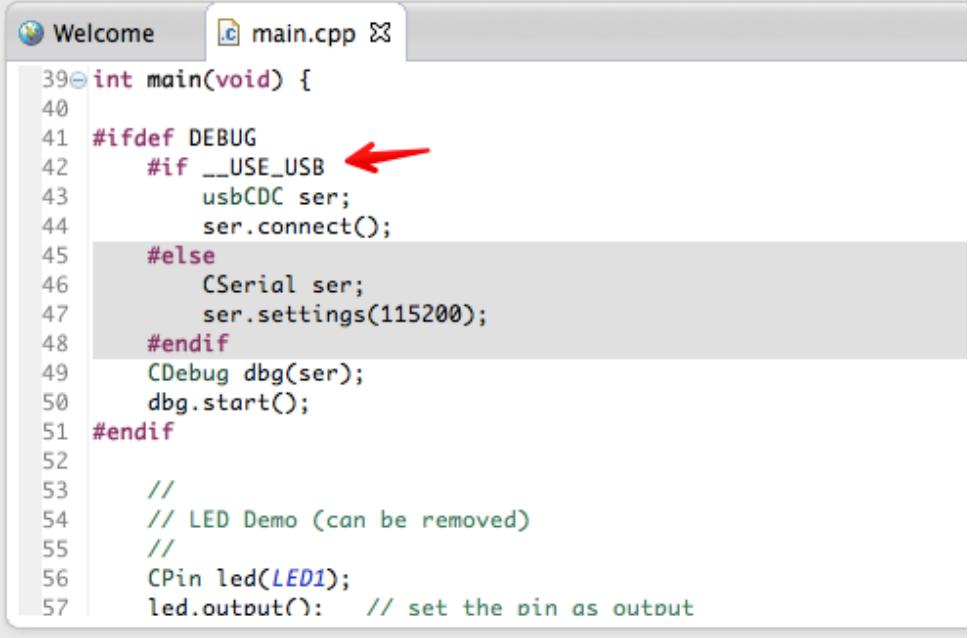


The “ISP” button.

4. Serial Terminal & Debug

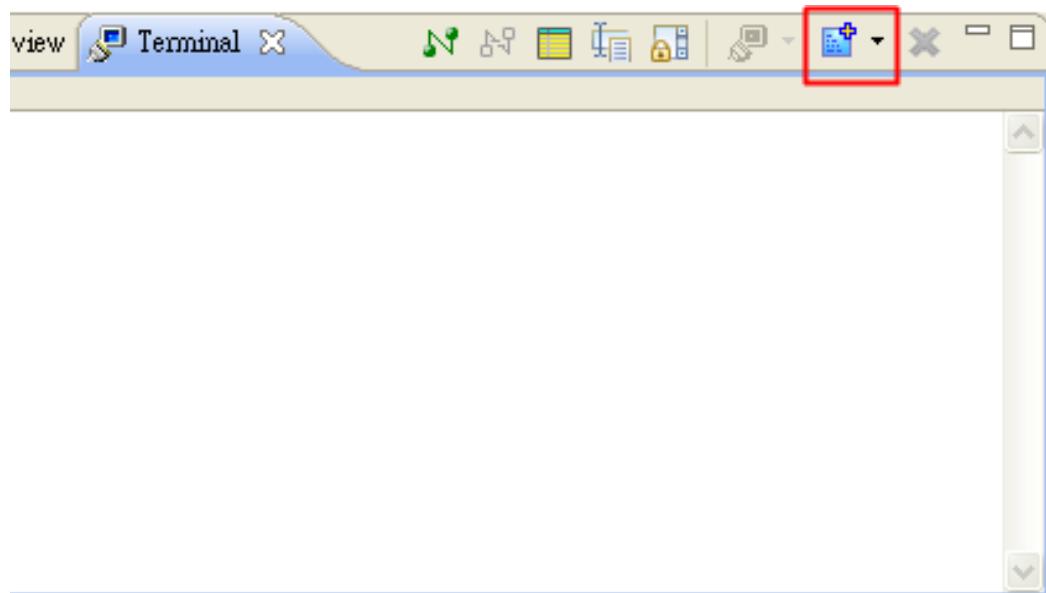
4.1 Serial Terminal (Use USB CDC Virtual COM. Port)

In main.cpp of blinkLED, the usbCDC will be enabled.



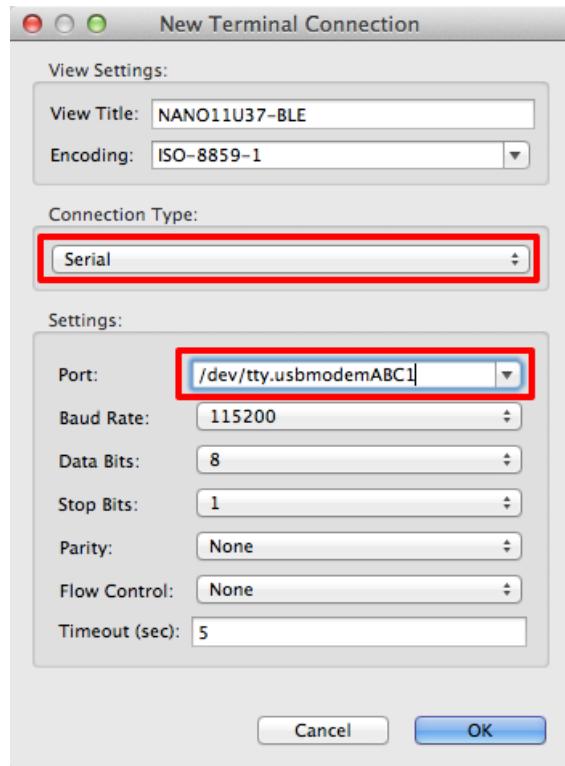
```
39 int main(void) {
40
41 #ifdef DEBUG
42 #if __USE_USB ←
43     usbCDC ser;
44     ser.connect();
45 #else
46     CSerial ser;
47     ser.settings(115200);
48 #endif
49     CDebug dbg(ser);
50     dbg.start();
51 #endif
52
53 // ...
54 // LED Demo (can be removed)
55 // ...
56 CPin led(LED1);
57 led.setOutput(); // set the pin as output
```

In Terminal View, click “New Terminal Connection” to add a new connection.



In Terminal Settings:

1. View Title: NANO11U37-BLE
2. Connection Type: Serial
3. Port : /dev/tty.usbmodemABC1 (for Mac OS/X)



The blinkLED serial shell will run in the Terminal View

```
Serial: (/dev/tty.usbmodemABC1, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)
*****
*      Welcome to uCXpresso.BLE      *
*      http://www.ucpresso.net        *
* (C)2012-2014 Embeda International Inc.* 
*****
ver      : get kernel version
task     : get tasks list
heap     : get heap available size
clear    : clear terminal screen
debug    : enter to debug mode
help or ? to show the command list.

uCpresso:/>
```

```

Serial: (/dev/tty.usbmodemABC1, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)
*****
*      Welcome to uCpresso.BLE      *
*      http://www.ucpresso.net       *
* (C)2012-2014 Embeda International Inc.*
*****
ver   : get kernel version
task  : get tasks list
heap   : get heap available size
clear  : clear terminal screen
debug  : enter to debug mode
help or ? to show the command list.

uCpresso:/>ver
uCpresso.BLE V1.0.1 (released)
FreeRTOS: V8.0.0
USBoot : 0.0.2

uCpresso:/>heap
Heap available size: 3544 bytes

uCpresso:/>task
Name      Stat.  Prio.  Remain  Num.
debug     R      0      32      4
IDLE      R      0      25      2
main      B      0      126     1
usbCDC   B      4      22      3

uCpresso:/>

```

Command:

ver, to show the kernel & module version.

heap, to check the available size of heap memory.

task, to check all of tasks in the system.

debug, to enter to debug mode to view the DBG(...) message from user's program.

4.2 Debug

Add 'key' object in main.cpp of blinkLED project.

```

55     //  

56     CPin led(LED1);  

57     led.output(); // set the pin as output  

58  

59     //  

60     // your setup code here  

61     //  

62     CPin key(P18);  

63     key.input();  

64  

65     while(1) {  

66         //  

67         // LED Demo (can be removed)  

68         //  

69         led = !led;  

70         sleep(500);  

71  

72         //  

73         // your loop code here  

74         //  

75         if (key==LOW) {  

76             dbg.printf("Key down!!\n");  

77         }  

78     }  

79     return 0 ;  

80 }
81

```

And add the key check in while loop.

Build Debug and download blinkLED.bin to nanoFLASH, then execute it.

In Terminal View, input 'debug' command to enter the debug mode:

```

Serial: (/dev/tty.usbmodemABC1, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)
*****
*      Welcome to uCpresso.BLE      *
*      http://www.ucpresso.net       *
* (C)2012-2014 Embeda International Inc.*  

*****  

ver   : get kernel version  

task  : get tasks list  

heap  : get heap available size  

clear : clear terminal screen  

debug : enter to debug mode  

help or ? to show the command list.  

uCpresso:/>debug  

press [ESC] to exit the debug mode  

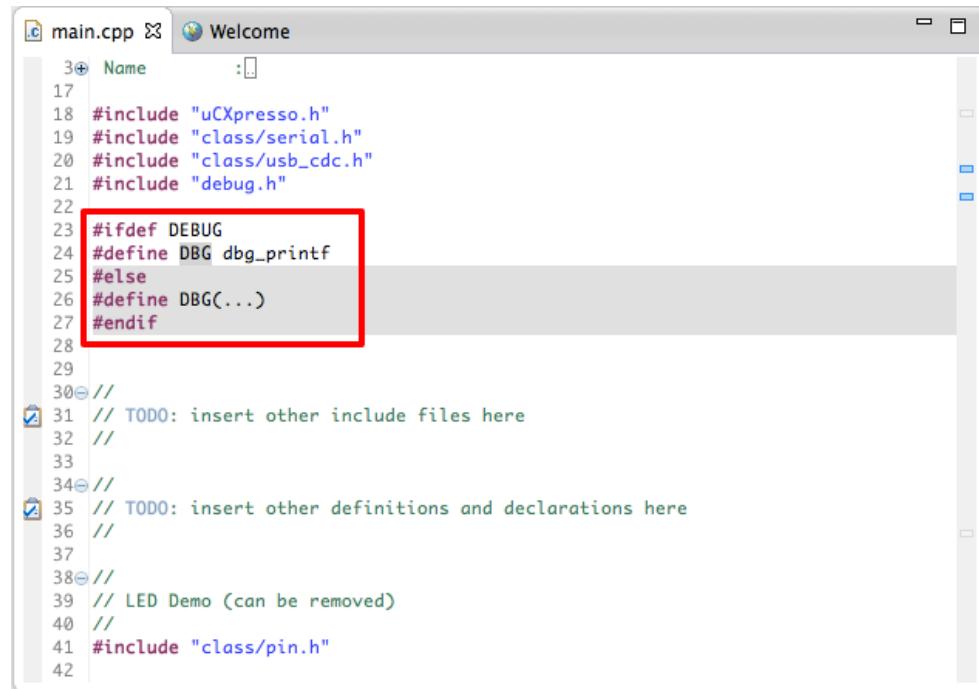
Key down!! ←
Key down!!

```

Try to push the key (pin 18) to LOW, and you can see the debug message to show in the terminal view.

4.3 Made an invalidly debug message code.

Made an invalidly debug message code in [Release Build]



```

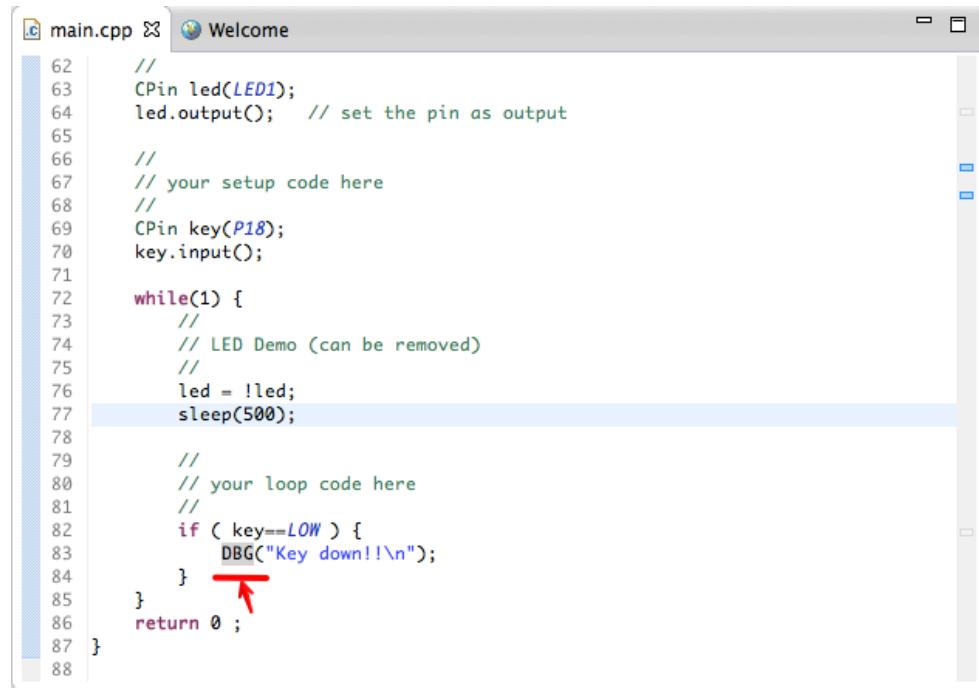
main.cpp  Welcome
3 Name :..
17
18 #include "uCpresso.h"
19 #include "class/serial.h"
20 #include "class/usb_cdc.h"
21 #include "debug.h"
22
23 #ifdef DEBUG
24 #define DBG dbg_printf
25 #else
26 #define DBG(...)
27 #endif
28
29
30 /**
31 // TODO: insert other include files here
32 /**
33
34 /**
35 // TODO: insert other definitions and declarations here
36 /**
37
38 /**
39 // LED Demo (can be removed)
40 /**
41 #include "class/pin.h"
42

```

The code editor shows a conditional compilation block. Lines 23 to 27 are highlighted with a red rectangle. Line 23 contains `#ifdef DEBUG`, line 24 contains `#define DBG dbg_printf`, line 25 contains `#else`, line 26 contains `#define DBG(...)`, and line 27 contains `#endif`. Lines 31 to 41 are marked with checkboxes and contain TODO comments.

Using the `#ifdef DEBUG` to block and define the `DBG` to `dbg_printf`, and `#else` to block and define the `DBG(...)` to nothing.

Rename `dbg.printf(...)` to `DBG(...)` in your program



```

main.cpp  Welcome
62 /**
63 CPin led(LED1);
64 led.output(); // set the pin as output
65
66 /**
67 // your setup code here
68 /**
69 CPin key(P18);
70 key.input();
71
72 while(1) {
73 /**
74 // LED Demo (can be removed)
75 /**
76 led = !led;
77 sleep(500);
78
79 /**
80 // your loop code here
81 /**
82 if (key==LOW) {
83 DBG("Key down!!\n");
84 }
85 return 0 ;
86
87 }
88

```

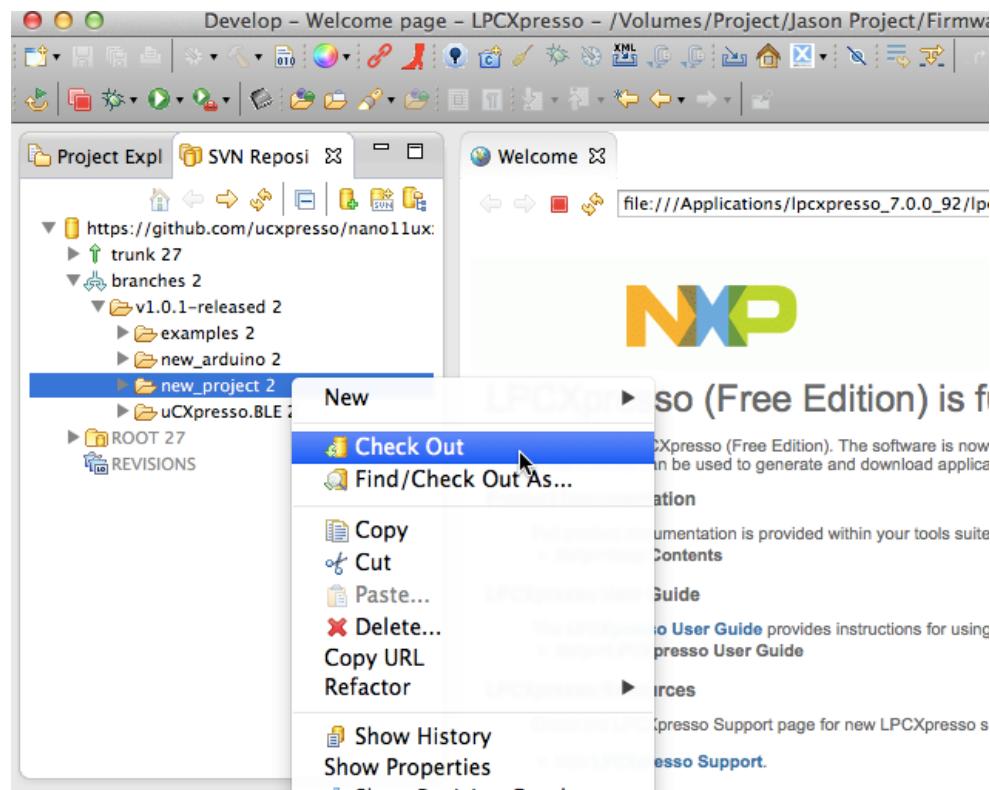
The code editor shows a loop with a sleep call at line 77. A red arrow points to the closing brace at line 85, indicating where the code was modified. The original `dbg.printf("Key down!!\n");` has been replaced by `DBG("Key down!!\n");`.

Now, The `DBG(...)` message code will be create in the [Debug Build] only, not in the [Release Build].

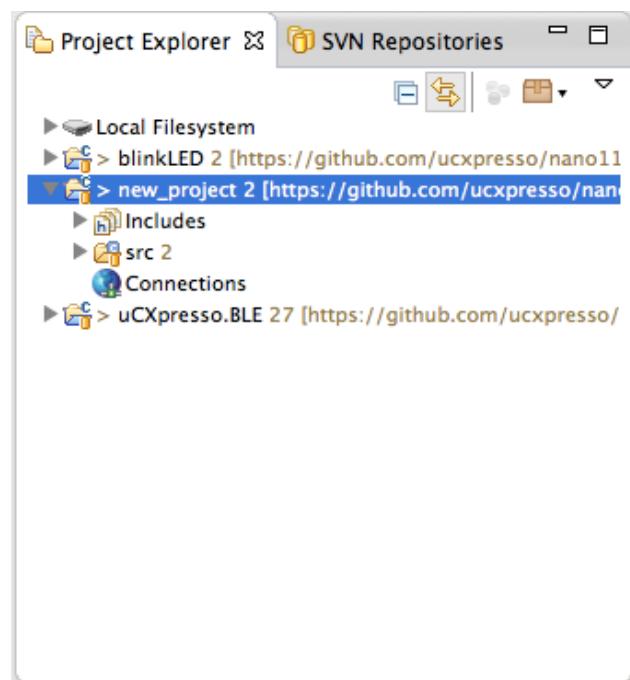
5. Create A New Project

5.1 Check Out the “new_project” template

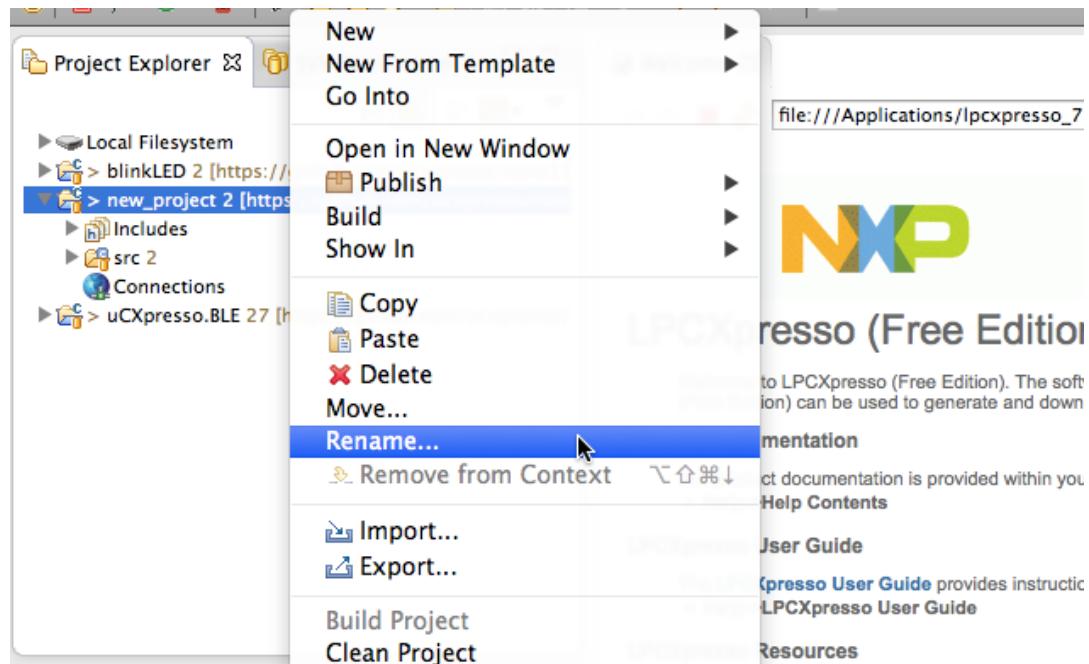
In SVN Repositories View, Expand the three and “Check Out” the “new_project” to workspace.



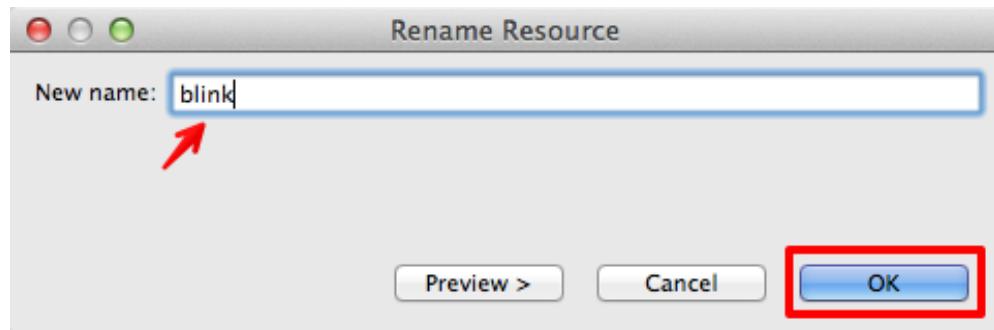
The “new_project” will show in workspace after Check Out.



Rename the “new_project” to new project name what you want.



For example to rename to “blink” as below,



An empty project “blink” is in the workspace.

The screenshot shows the LPCXpresso IDE interface with the following details:

- Title Bar:** Develop - blink/src/main.cpp - LPCXpresso - /Volumes/Project/Jason Project/Firmware/lpcxpresso/cortex-m0
- Project Explorer:** Shows a local filesystem with a project named "blink 2" containing "src 2" and "main.cpp".
- SVN Repositories:** Shows connections to "uCxpresso.BLE" and "nanollux".
- Welcome View:** Displays the main.cpp file content.
- Code Editor:** The main.cpp file contains the following code:

```
61  ****
62  *
63  *          your setup code here
64  *
65  ****
66
67  // LED Demo (can be removed)
68  //
69  DBG("Hello I'm in debug mode\n");
70  uint8_t i = 0;
71  CBus port(LED1, LED2, LED3, LED4, END);
72  port.output(); // set all pins as output
73
74
75  while(1) {
76      ****
77      *
78      *          your loop code here
79      *
80      ****
81
82  // LED Demo (can be removed)
83  //
84  port = led_scripts[i];
85  i = (i+1) < sizeof(led_scripts) ? i+1 : 0;
86  sleep(100);
87
88
89
90 }
91
92 }
93
94 //
```

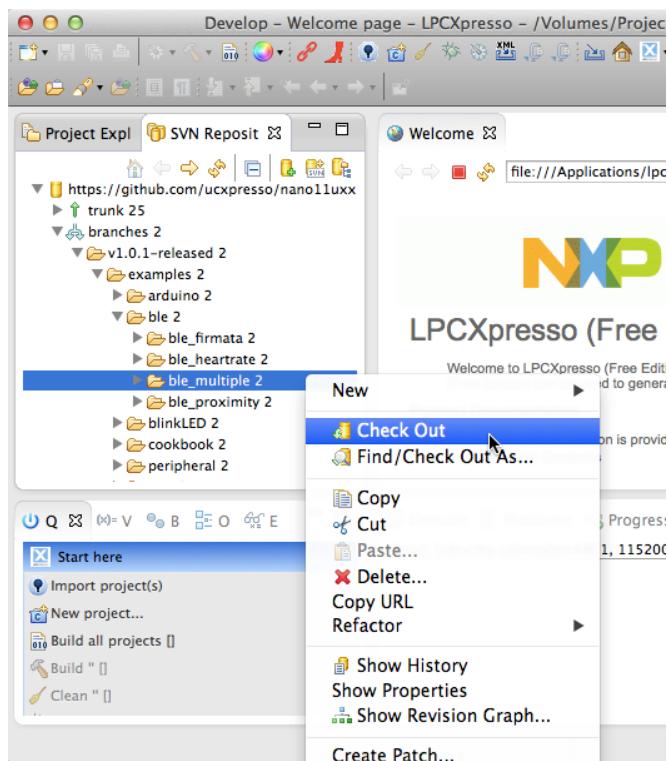
- Console View:** Shows the CDT Build Console [blink] with the message: NXP LPC11U37/401 (blink).
- Bottom Navigation:** Includes icons for File, Open, Save, and Recent Projects.

Expand the 'src' folder of 'blink' project, and start to program your main.cpp.

6. Example ble_multiple Project

The ble_multiple project included:

1. Tickless Power Save Feature
2. BLE Arduino Firmata Test
3. BLE Battery Level Service
4. BLE Health Thermometer Service
5. BLE Proximity Service



iOS App : BLE Arduino

<https://itunes.apple.com/tw/app/ble-arduino/id547628998>



Youtube Firmata Demo: http://youtu.be/7v7_mqfynRA

7. Class Manual

6.1 Online Class Manual

<https://rawgithub.com/ucpresso/nano11uxx/master/uCXpresso.BLE/doc/doxygen/html/index.html>

The screenshot shows a web browser window with the title "uCXpresso.BLE: bleSerial C" and the URL "https://rawgithub.com/ucpresso/nano11uxx/master/uCXpresso.BLE/doc/doxygen/...". The browser's address bar and various icons are visible at the top. The main content area is titled "uCXpresso.BLE v1.0.2" and "RTOS C++ Framework for Bluetooth Low Energy". A navigation bar at the top has tabs for "Main Page", "Modules", and "Classes", with "Classes" being the active tab. Below the navigation bar is a sub-navigation bar with tabs for "Class List", "Class Index", "Class Hierarchy", and "Class Members". A search bar is located at the top right. The left sidebar shows a tree view of the class hierarchy under "uCXpresso.BLE" and "Modules". The "BLE" module is expanded, showing classes like "bleBatteryLevel", "bleHeartRate", "bleHealthThermometer", "bleProximity", and "bleSerial". The "bleSerial" class is selected and highlighted in blue. The main content area on the right is titled "bleSerial Class Reference" and contains a brief description of the class: "bleSerial class is a ble core, and inherits from CStream class to provide the stream virtual functions for serial input and output. the bleSerial class also inherits from the CThread class and can be work in background." It includes a "More..." link, a code snippet starting with "#include <class/ble_serial.h>", and an "Inheritance diagram for bleSerial:" section which includes a UML-like inheritance diagram. The diagram shows "bleSerial" inheriting from "CStream" and "CThread", which both inherit from "CObject". The footer of the page indicates it was generated on "Tue Mar 11 2014 10:13:40" for "uCXpresso.BLE" by "doxygen 1.8.6".