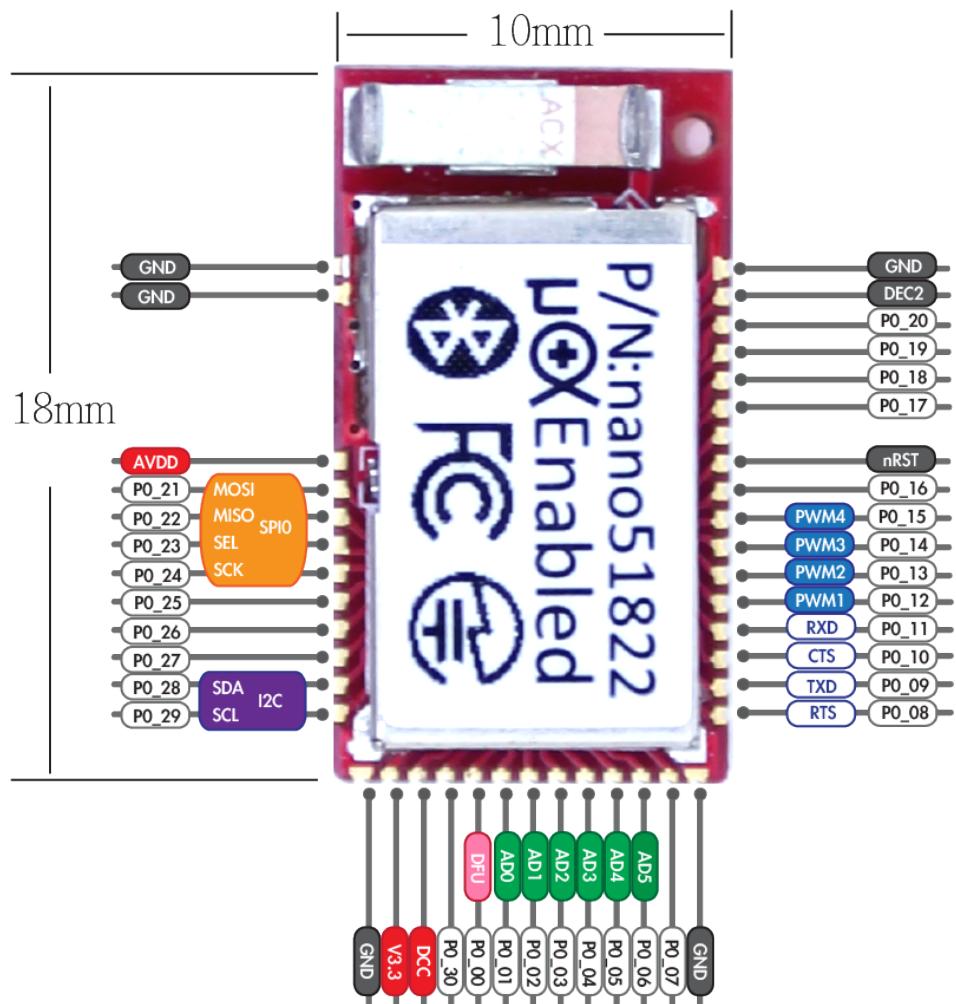


Getting Started with uCpresso.NRF



2014/10/21

Beta-1b

uCpresso.NRF

History:

2014/10/20 First Edition

Overview

First of all, an espresso is necessary before the getting started.



The uCpresso 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/lpcpresso/download>

The uCpresso.NRF is a RTOS C/C++ framework that port for the nano51822 SoC BLE module (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...).

Multi-Thread

The uCpresso.NRF framework is based on an advanced Real-Time Operating System (RTOS), By using the RTOS and the multi-tasking feature, you will simplify your concept, coding, and maintenance.

How uCpresso.NRF RTOS works

When you type the ‘task’ on the debug console of serial-terminal, you can see the all tasks, which runs on the nano51822.

```

***** Welcome to uCpresso.NRF *****
* 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:/>task
Name      Stat. Prio. Remain Num.
IDLE      R      0      10     2
debug     R      0      16     3
main      R      0      34     1
bleDev    B      2      32     4

uCpresso:/>ver
uCpresso.NRF V1.0.0 rc0 (2014/10/18)
Kernel (RTOS): V8.1.2
SoftDevice V7 / FWID:005A
DFU V141020-00

uCpresso:/>

```

- The “bleDev” task is the “[SoftDevice](#)” driver and run in the background to receive and dispatch the ble events.
- The “main” task is the “main-routine” for application entry and main loop.
- The “debug” task is handler the DBG and debug console task.

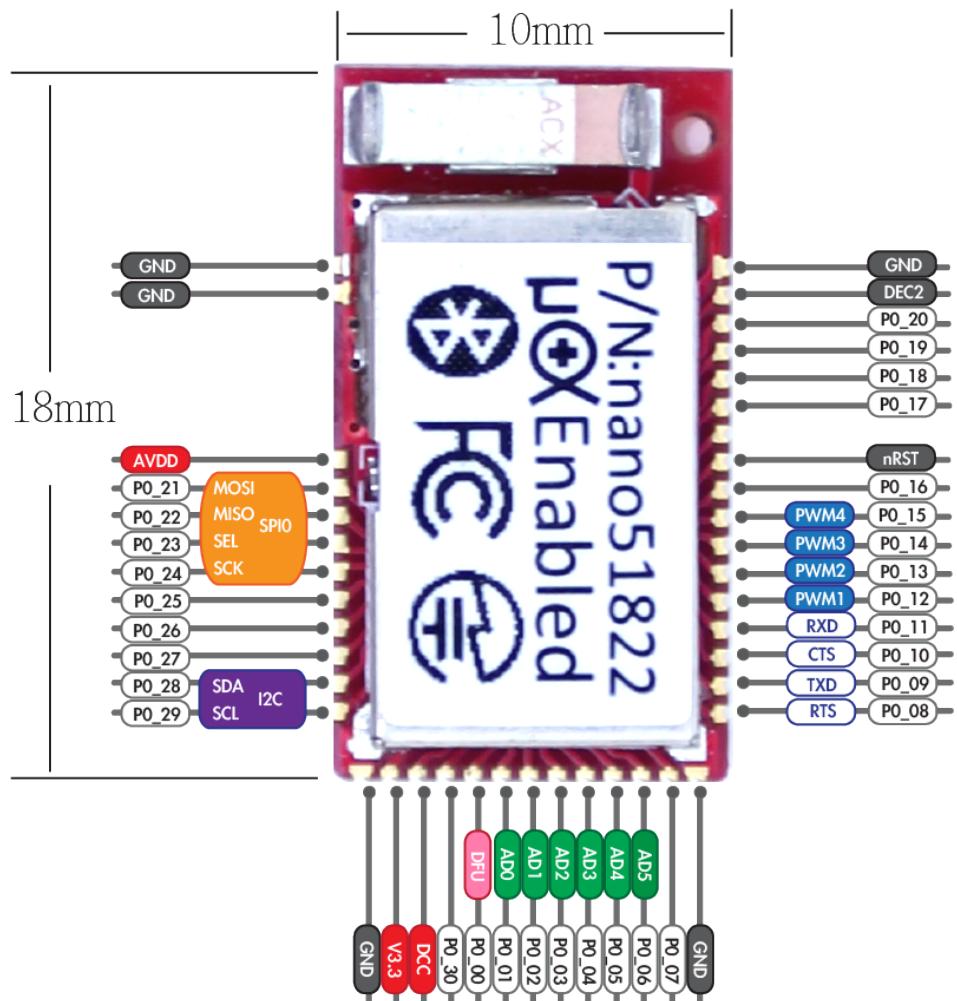
Easy to use with C++ framework:

Ex. A LED on the pin 19 of nano11U37

```

CPin led(18);           // connect led object to P0.18 pin
led.output();           // set led as an output pin
led = HIGH;             // set led to HIGH (turn on LED)

```

Default Pin Definition:**DFU pin on P0.0 and active low (internal pull-up):**

The nano51822 will be entered into the DFU mode when DFU pin is low.

If your application is crash, you also can force to enter into the DFU mode, and just put the DFU pin to low before power-on (or reset) the nano51822.

Ex. Set P0.20 as an input pin

```
CPin key(20);           // connect key object to P0.20
key.input();             // set key as an input pin with internal Pull-Up.
if ( key==LOW ) {       // check key PIN LEVEL
    ...
}
```

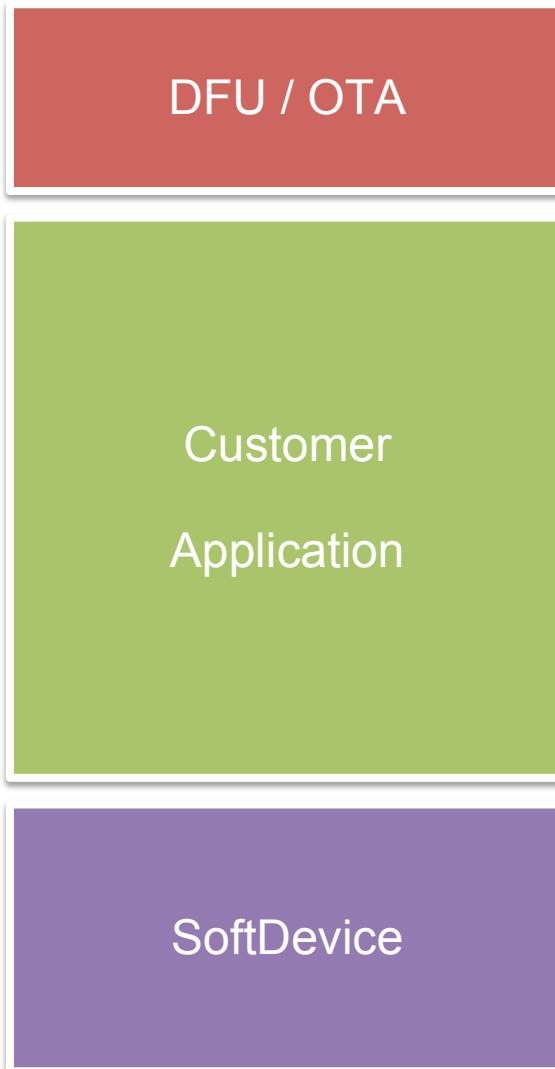
Ex. Set P0.20 as a PWM output

```
swPwm servo;
servo.period(0.02);     // set global PWM period time to 20ms
```

uCpresso.NRF

```
servo.add(20);           // add P0.20 to servo (pwm) module  
servo.enable();          // enable the servo PWM to output  
servo.dutyCycle(0, 0.8); // set servo duty-cycle to 80%
```

The blocks of nano51822



Final,

Let uCpresso RTOS C/C++ framework decreases your development time and increases your joy and creativity!

1. Upgrade Your LPCXpresso IDE

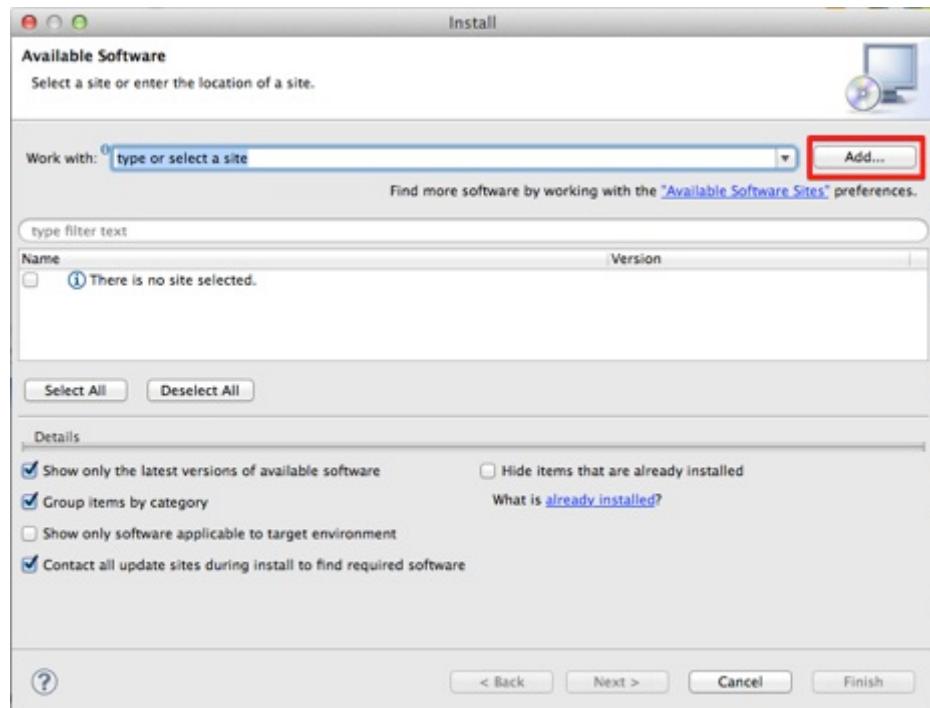
1.1. Increase VM memory for Windows System

- 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...

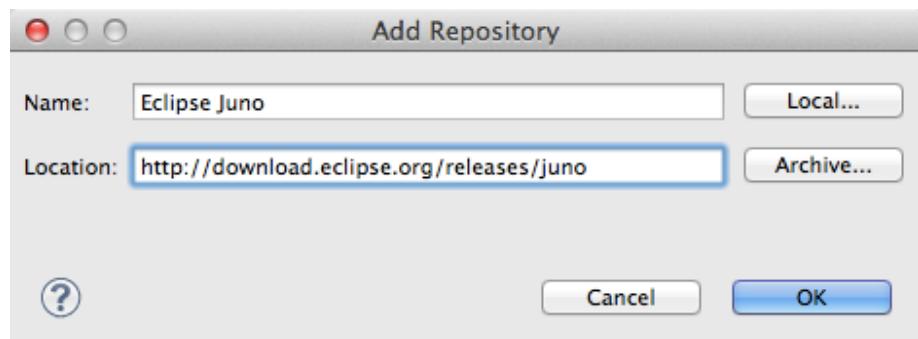
Click [Add]



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
General Purpose Tools	
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
<input checked="" type="checkbox"/> Marketplace Client	1.1.1.I20110907-0947
Memory Analyzer	1.2.1.201211051250

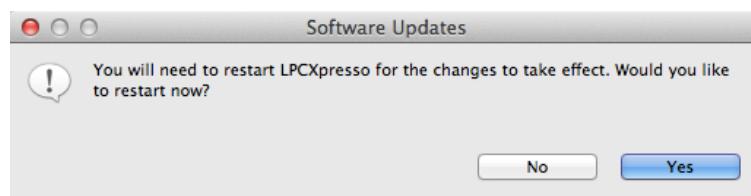
Select All Deselect All 2 items selected

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
<input checked="" type="checkbox"/> 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	

Select All Deselect All 2 items selected

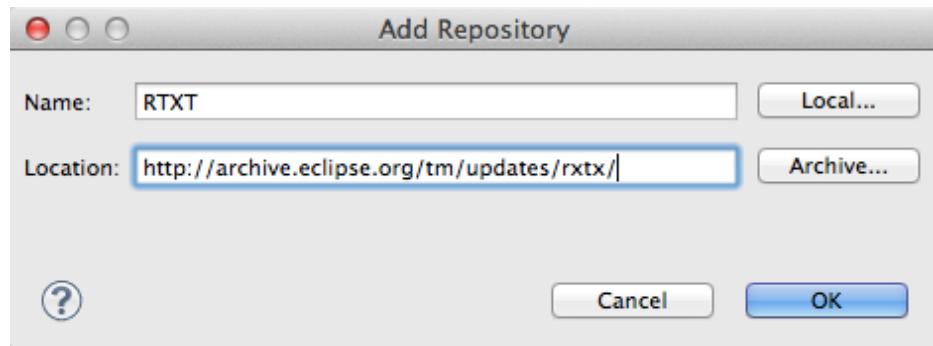
Click [Next] to start the installation. In progress, you need to [Confirm >] the Selected Features, and [accept] the licenses of software, and click [OK] to confirm any security-warning message, click [Yes] to restart and finish the installation if need.



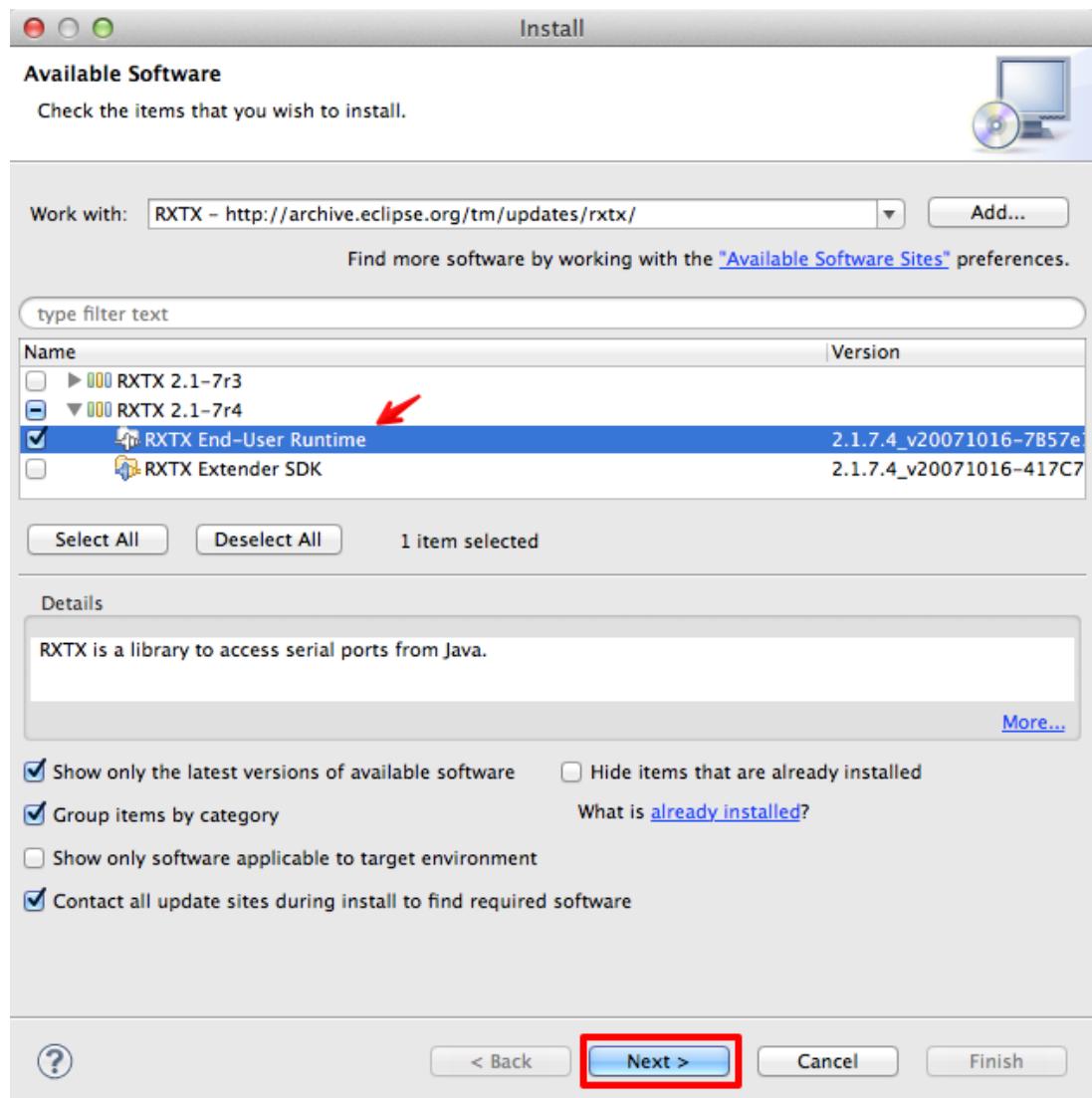
1.4. Install RXTX Plug-in

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:

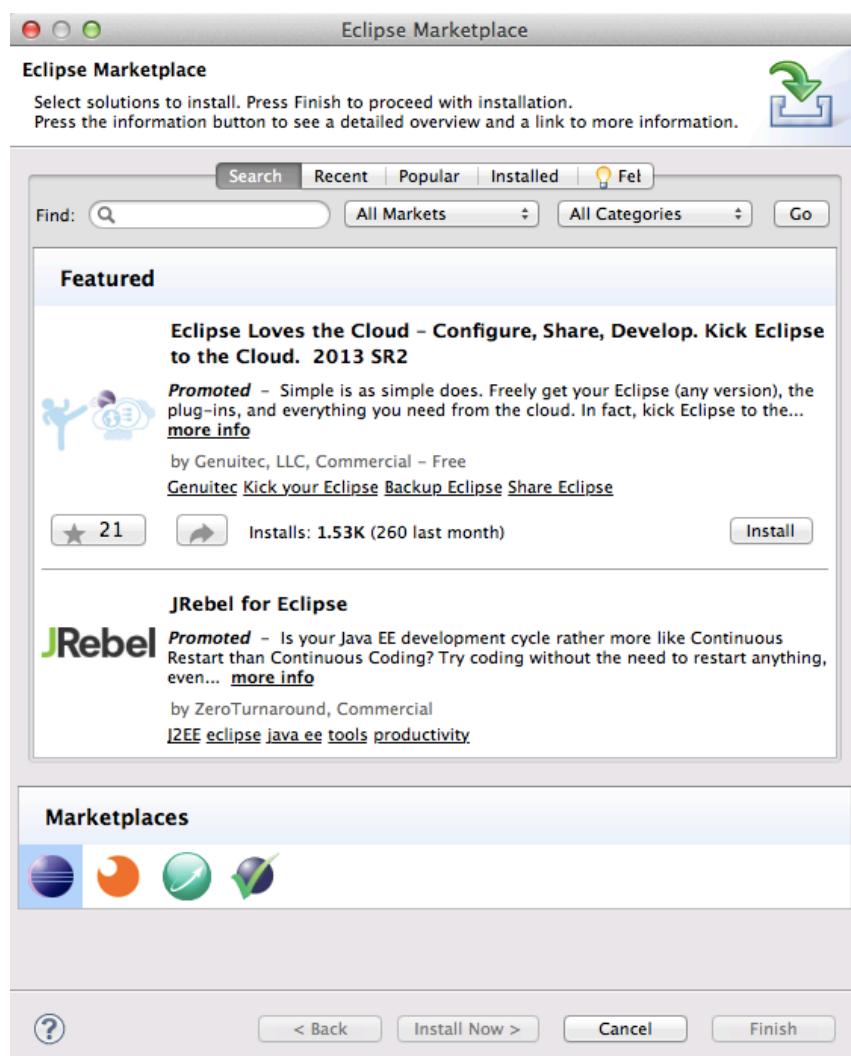
1. Open Terminal App of OS/X
2. Type :
sudo mkdir /var/lock
sudo chmod a+rwx /var/lock

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

for NANO11Uxx : http://www.embeda.com.tw/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.)

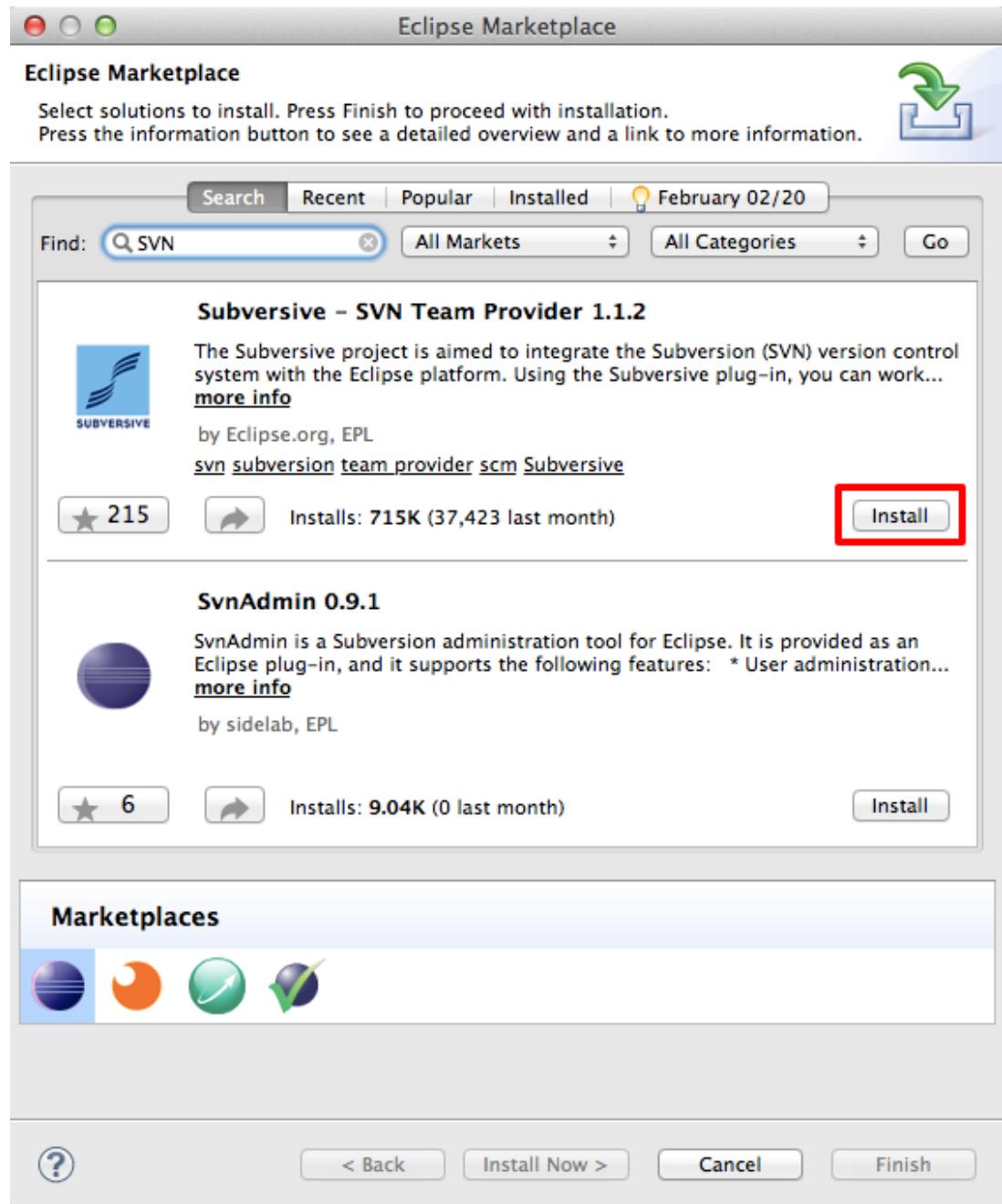
1.5. Install new software from Eclipse Marketplace

Click LPCXpresso Main Menu > Help > Marketplace...

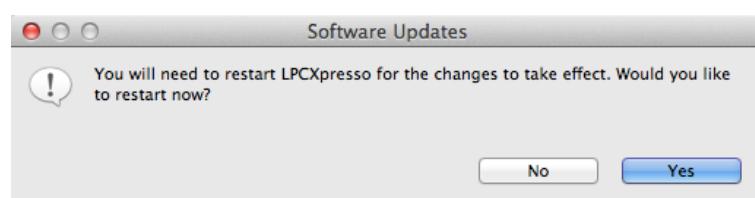


1.6 Install Subversion – SVN Team Provider (Plug-in)

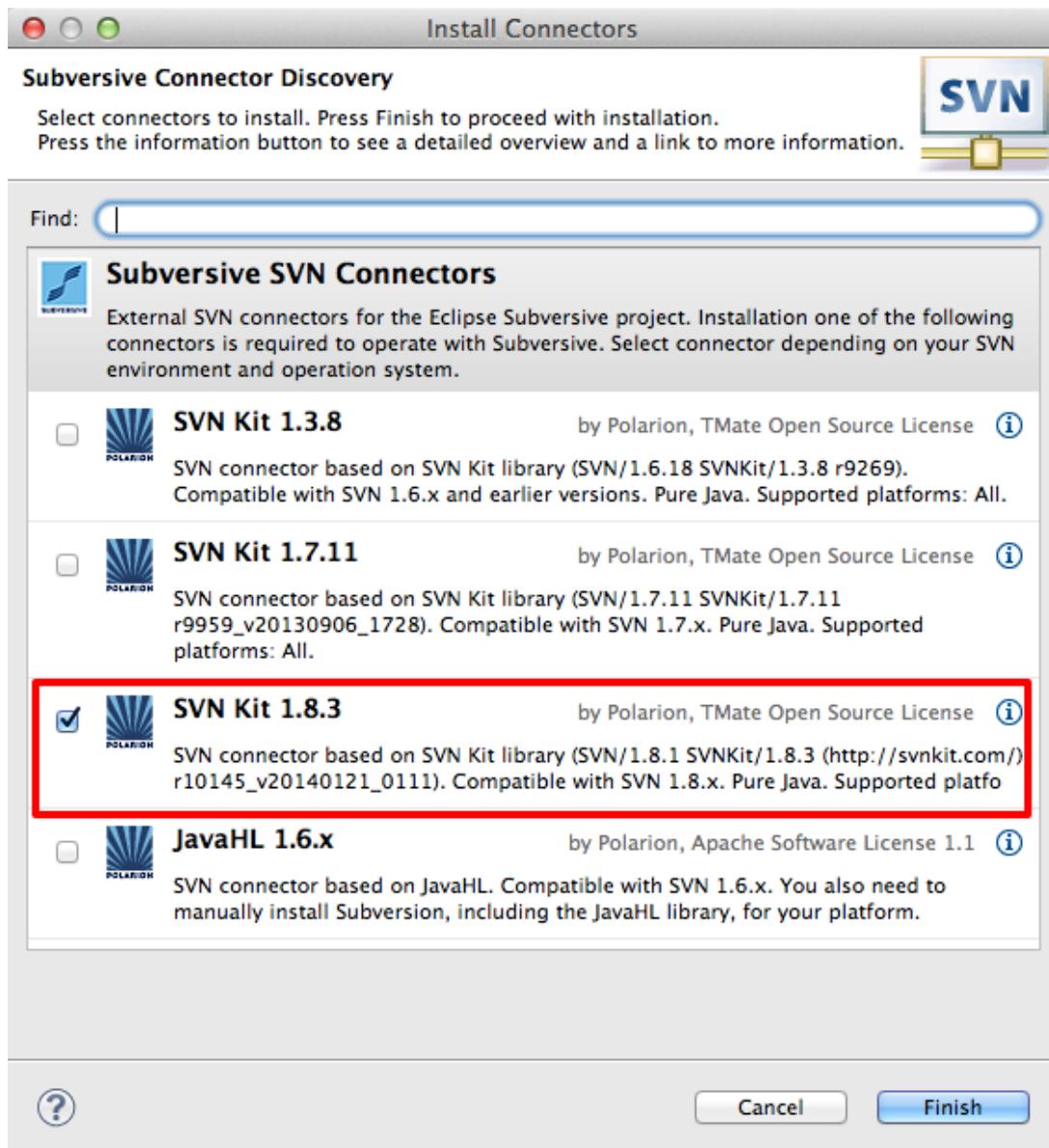
In Find, input “SVN” and click [Go]



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:

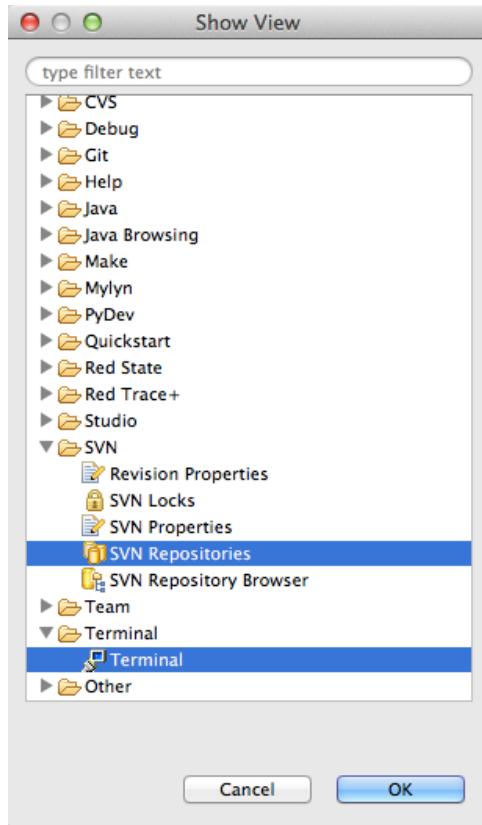


Select the latest version “SVN Kit x.x.x”, and click [Finish] to start the installation. Confirm and click [Next], and [accept] the licenses of software, and click [OK] to confirm any security-warning message. Click [Yes] to restart and finish the installation if need.



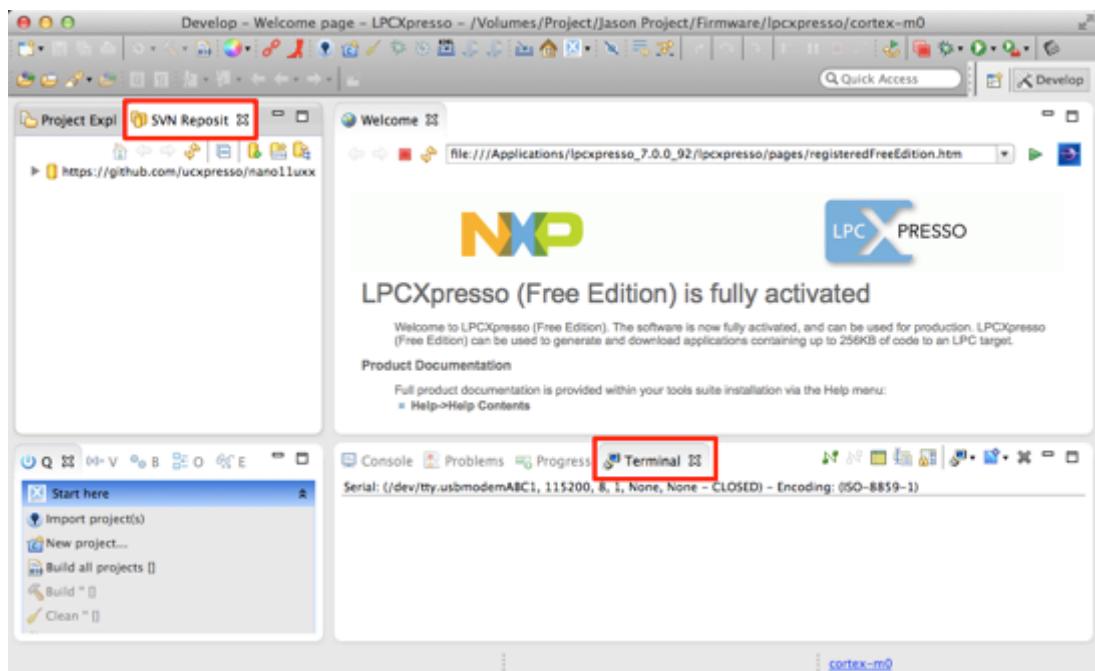
1-7. Show the SVN and Terminal Views

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)

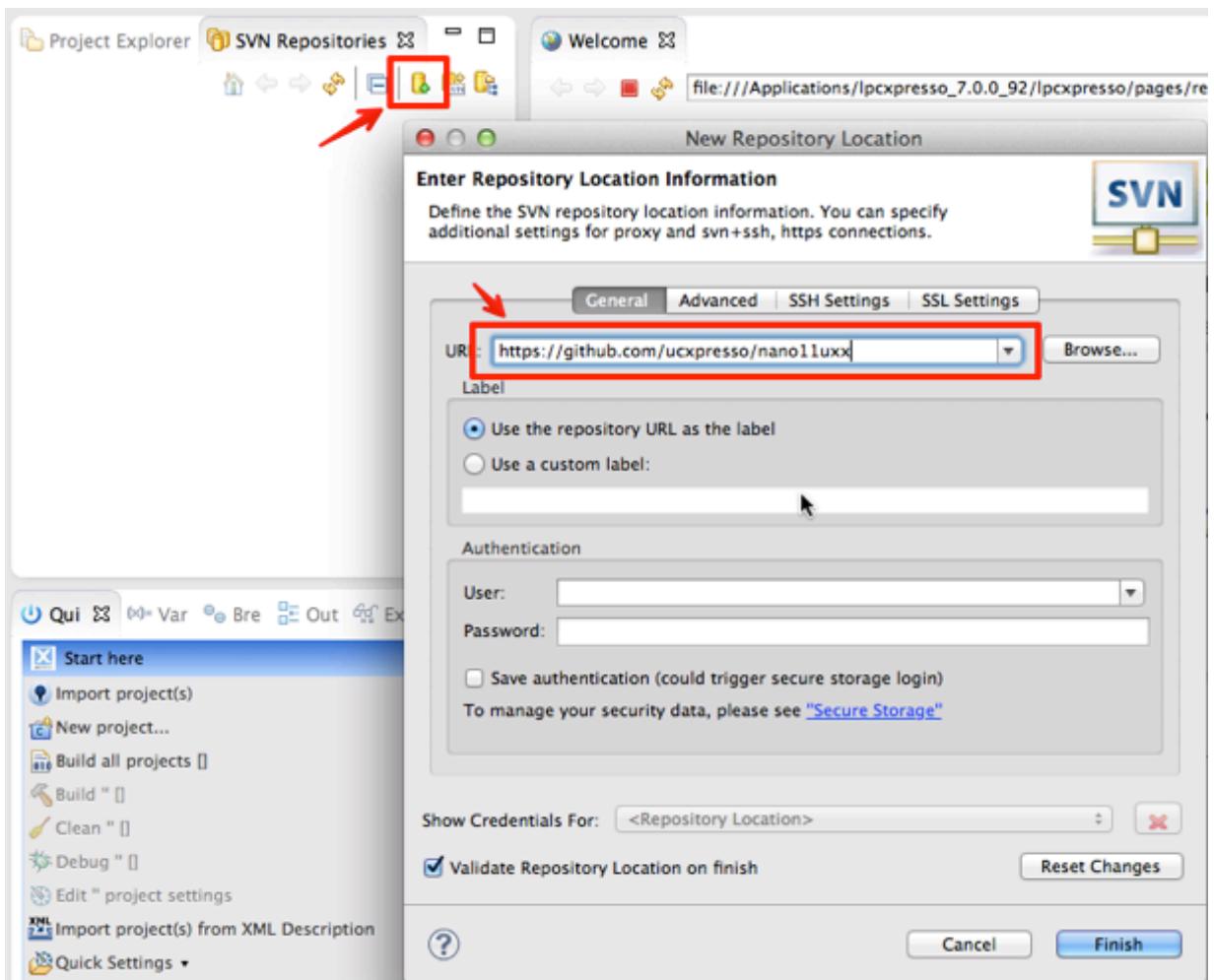


2. Install uCpresso.NRF Framework

2.1 Using GitHub Repositories



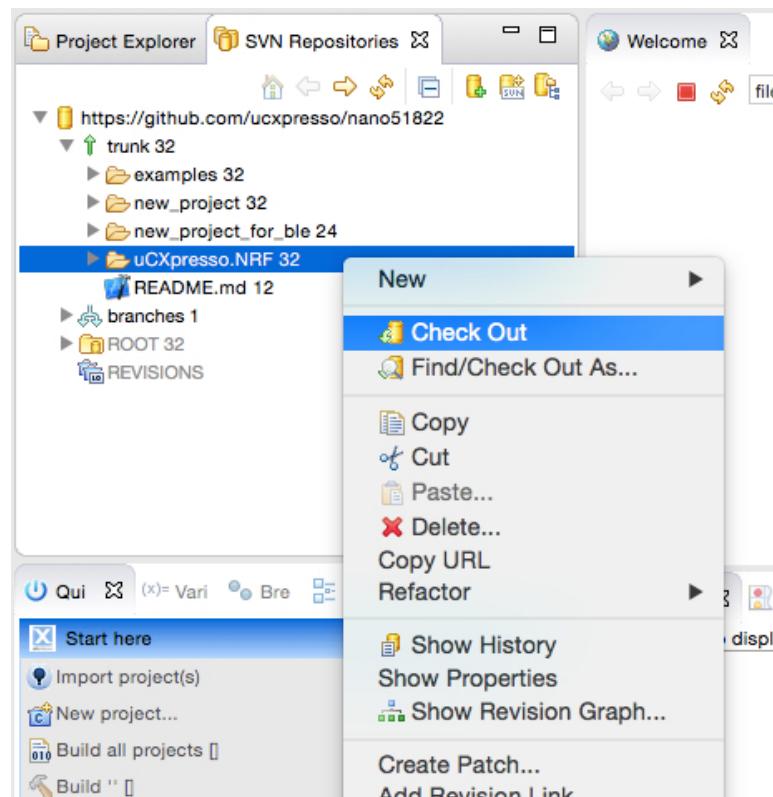
In “SVN Repositories” Tab, Click “New Repository Location”,



and URL Input: <https://github.com/ucpresso/nano51822>

2.2 Check Out uCpresso.NRF framework

Expand the uCpresso.NRF SVN repository

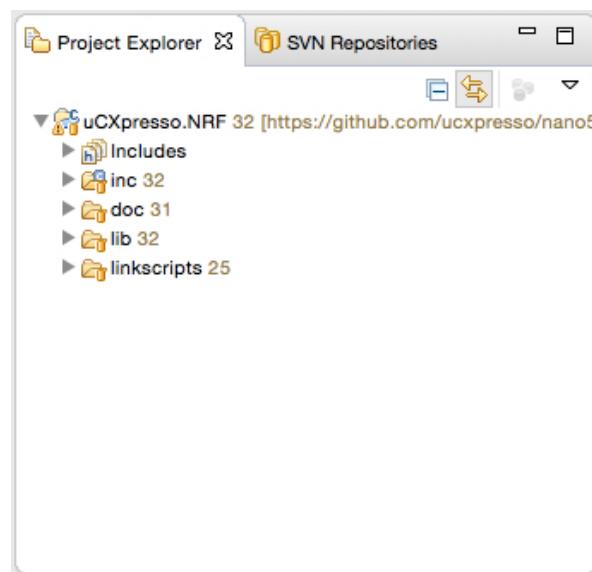


trunk: uCpresso.NRF RC (Release Candidate) version

branches: All released version

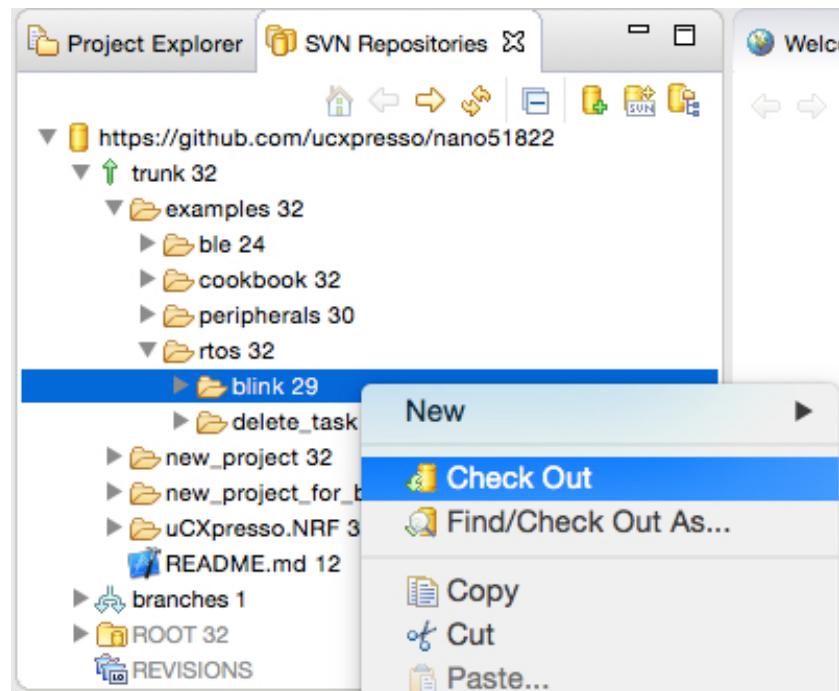
Select “uCpresso.NRF” folder, and click right button of mouse, and click “Check Out” in drop down menu.

The uCpresso.NRF framework show in your workspace after check out.



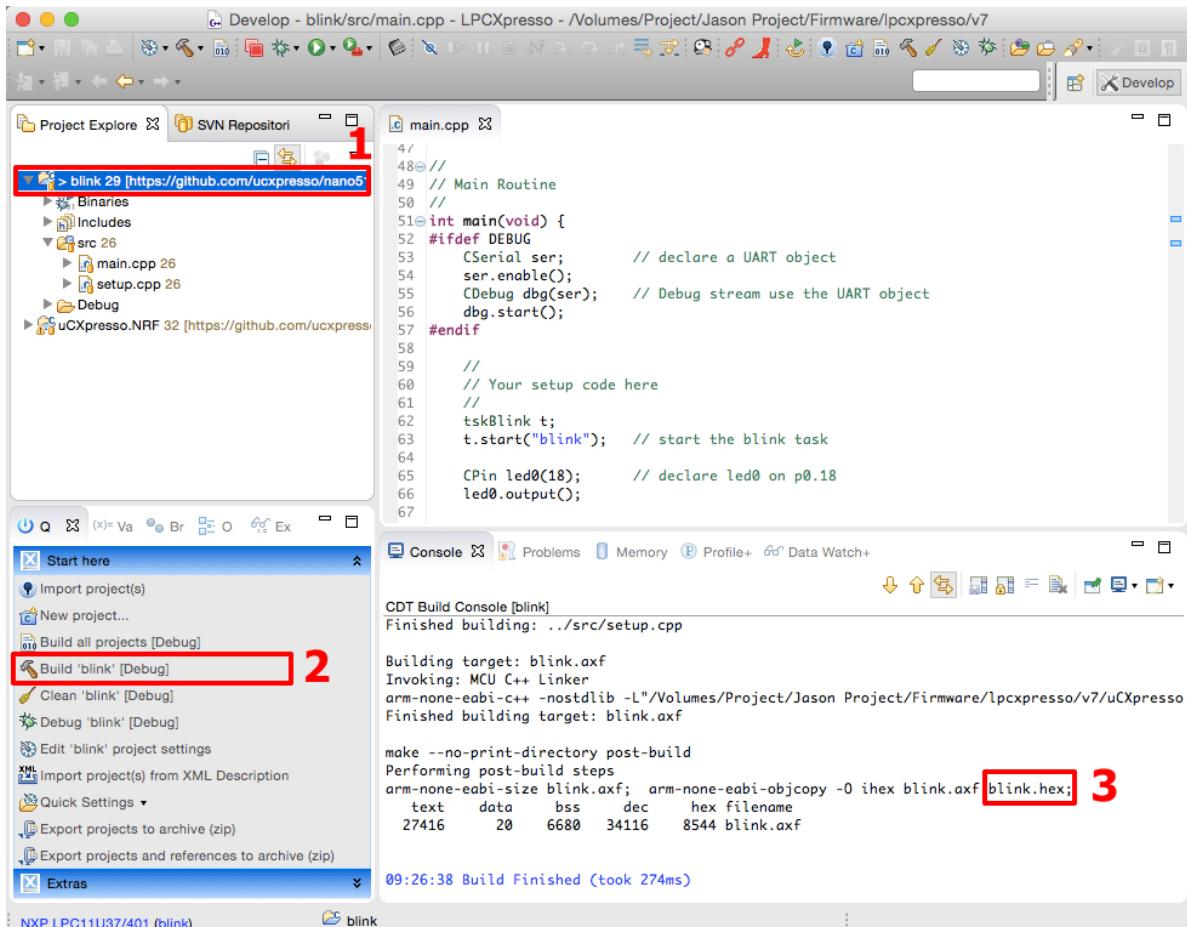
2.3 Check Out “blink” example

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

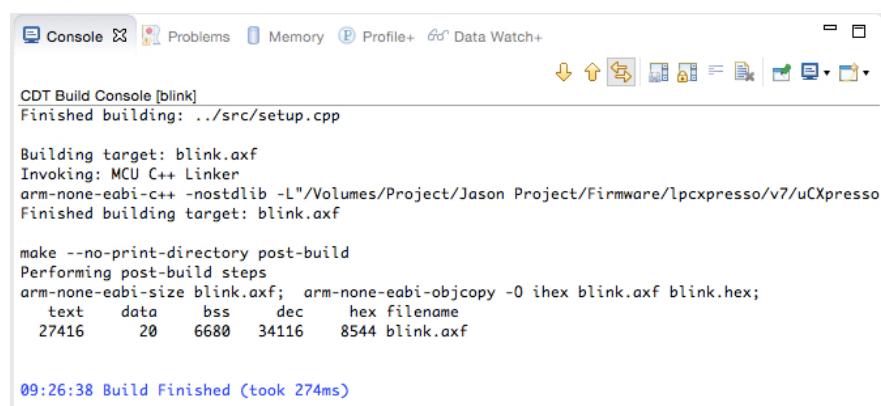


2.4 Build the “blink” example project

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



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



Note:

code size = text + data = 27436 bytes

ram size = data + bss = 6700 bytes

3. Development Firmware Upgrade (DFU) and Over the Air (OTA)

Note:

The nano51822 provide the DFU code in module already.

3.1 Install the “Master Control Panel” (MCP) App for Android

Looking for “[nRF Master Control Panel](#)” on Google Play:

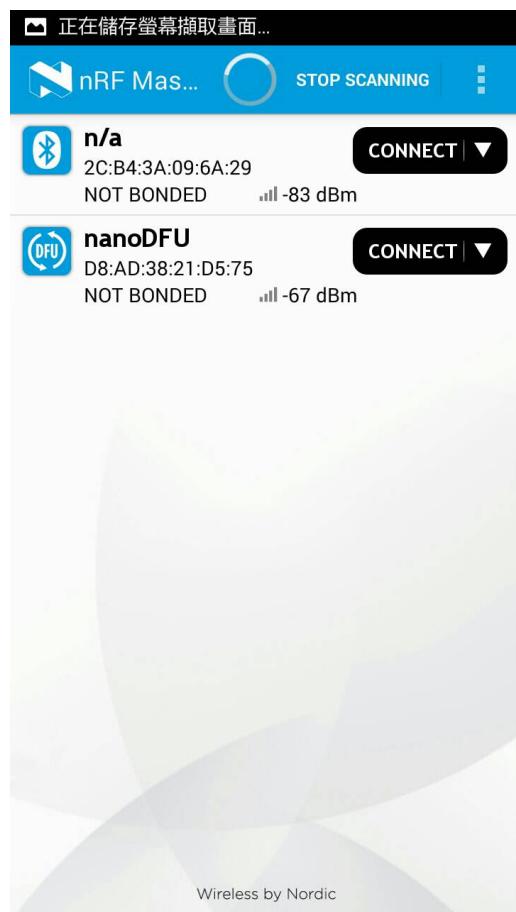
<https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp>

3.1 Prepare your Cloud Driver (Support Google Driver and Dropbox)

For example, copy the [blink.hex](#) to your dropbox that built at chapters 2.4.

3.2 DFU over the air

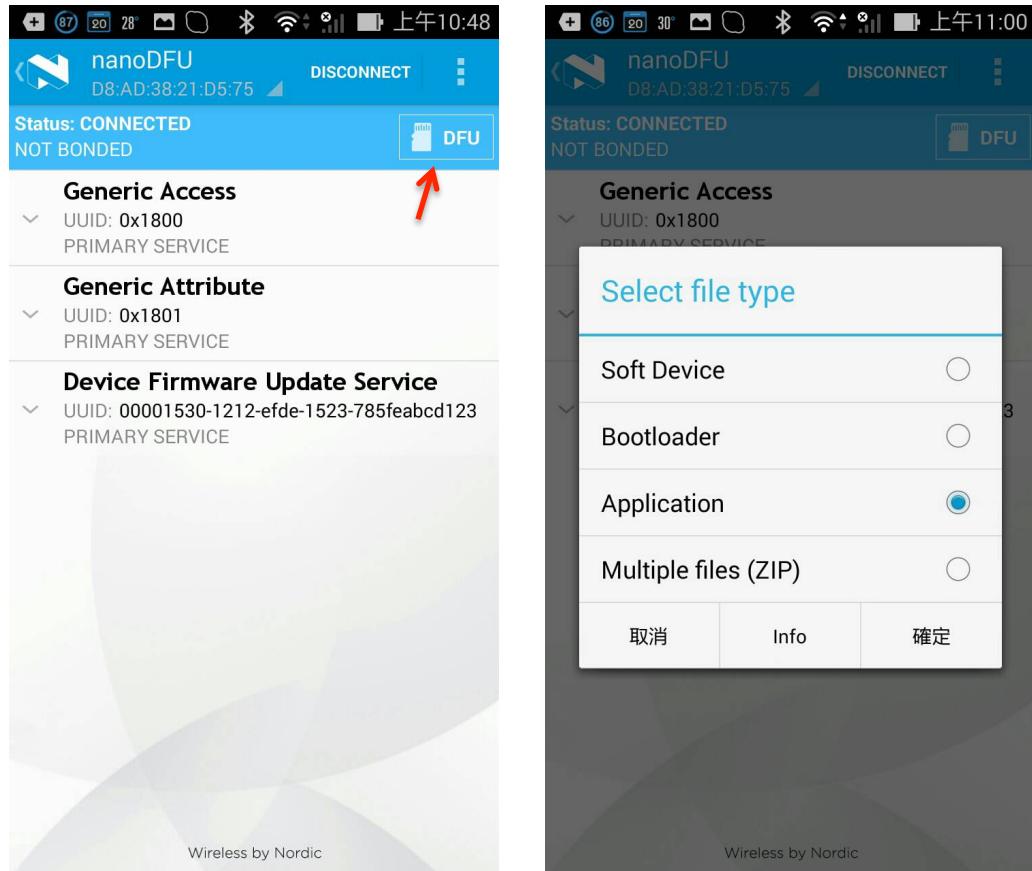
Scan the nano51822 DFU device (nanoDFU) in MCP.



the nanoDFU will show in the scan list.

Click [CONNECT] on nanoDFU.

Connect to the nanoDFU device:

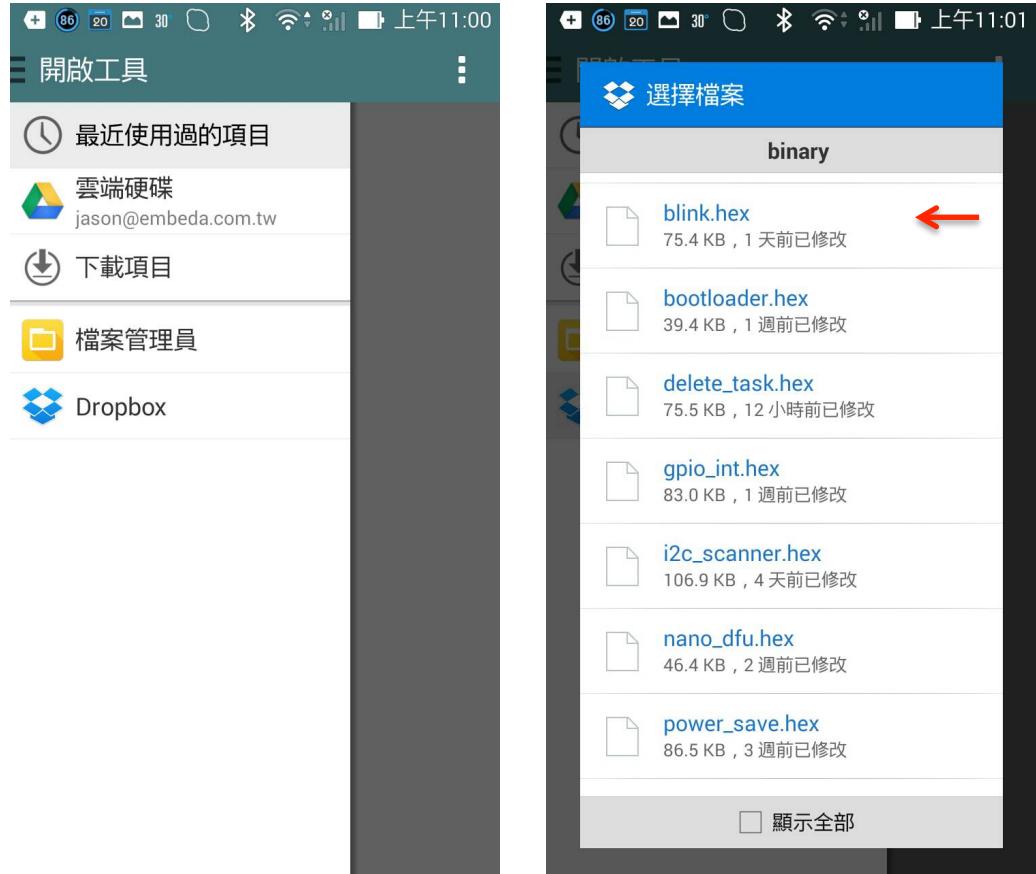


Click [DFU] and Select file type to “Application”

Warning:

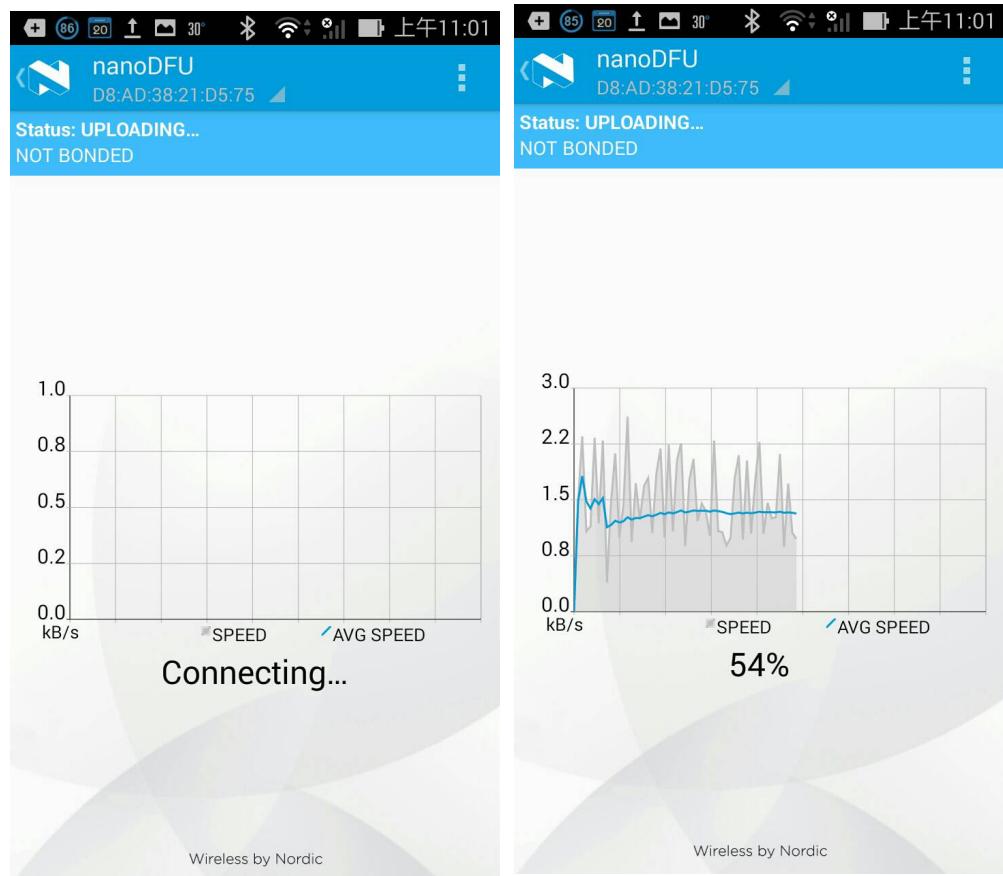
You have to select the “Application” section for your application updated in nano51822.

Select cloud driver (Google or Dropbox)



Select the “blink.hex” which you want to upload to DFU device.

Start to upload the “blink.hex” firmware to nanoDFU device.

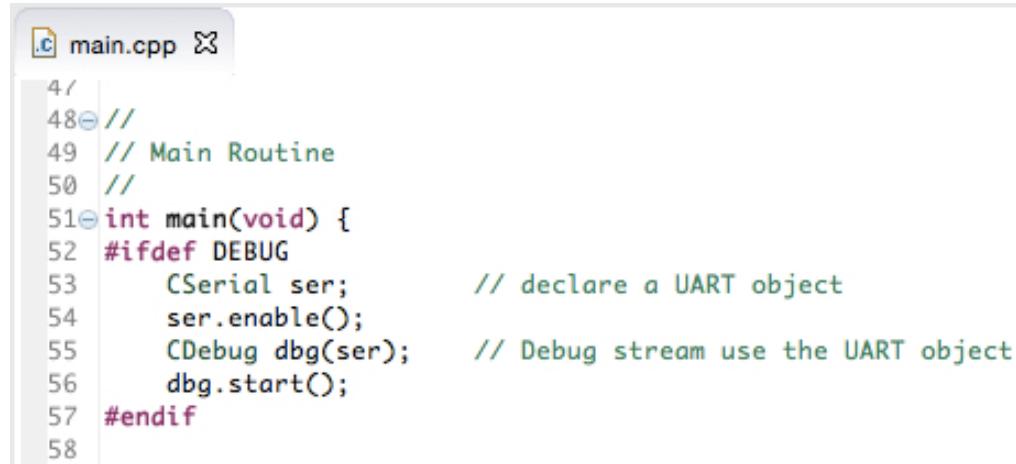


After finished, the nanoDFU will run your application automatically.

4. Serial Terminal & Debug

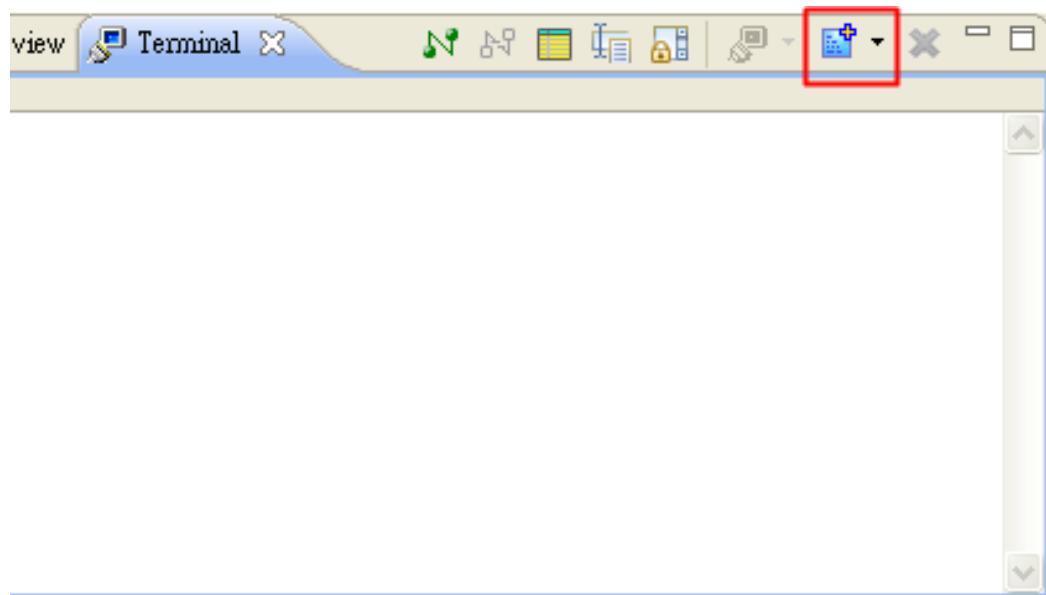
4.1 Serial Terminal (You need an USB to UART convertor)

In main.cpp of blink, the serial debug console will be enabled in DEBUG build.



```
main.cpp
47
48 // Main Routine
49 // Main Routine
50 //
51 int main(void) {
52 #ifdef DEBUG
53     CSerial ser;          // declare a UART object
54     ser.enable();
55     CDebug dbg(ser);      // Debug stream use the UART object
56     dbg.start();
57 #endif
58 }
```

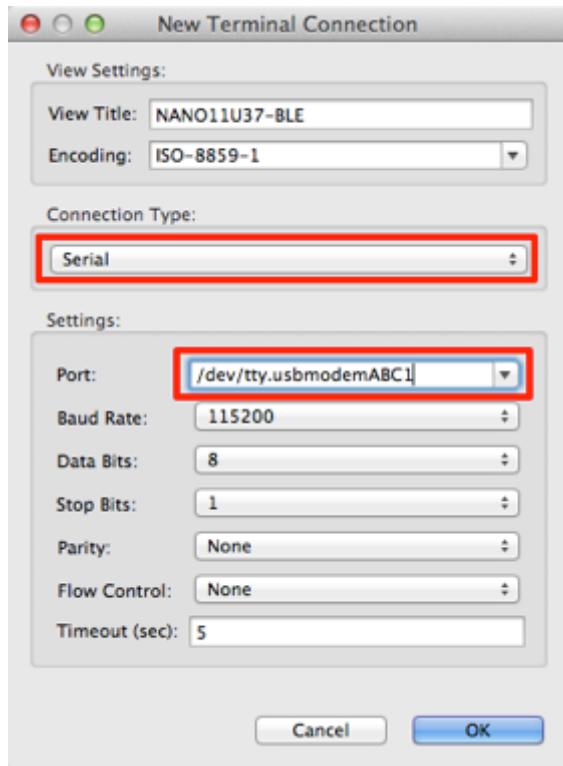
In Terminal View, click “New Terminal Connection” to add a new connection.



uCpresso.NRF

In Terminal Settings:

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

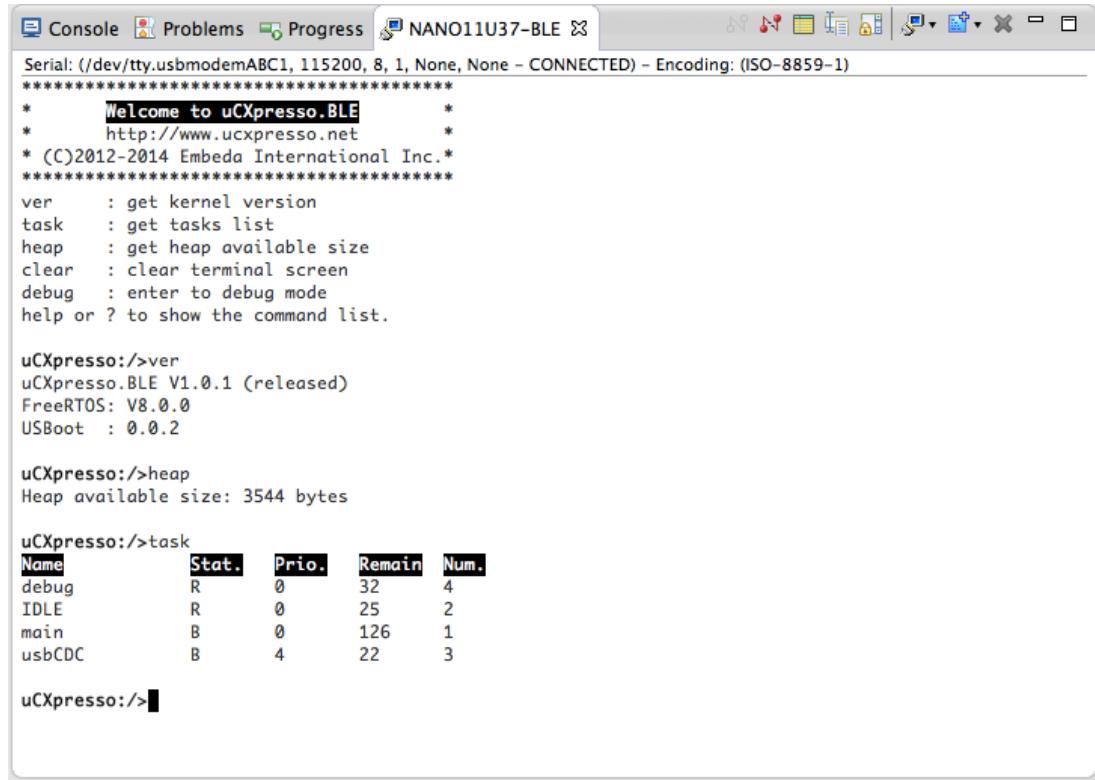


The blink serial shell will run in the Terminal View

```
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:/>
```

uCpresso.NRF



The screenshot shows the Eclipse IDE interface with the "Console" tab selected. The title bar indicates "NANO11U37-BLE" and "Serial: (/dev/tty.usbmodemABC1, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)". The console window displays the following output:

```
*****
*      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 and view the DBG(...) message from user's program.

4.2 Debug

Add 'key' object in main.cpp of blink 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 blink.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  

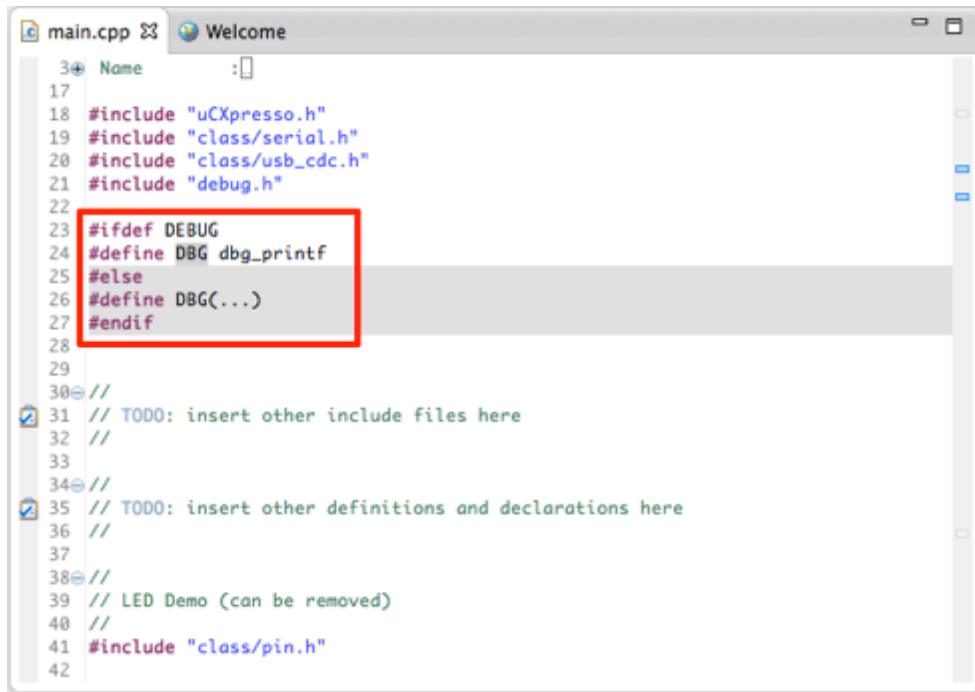
Key down!! ←

```

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]



```

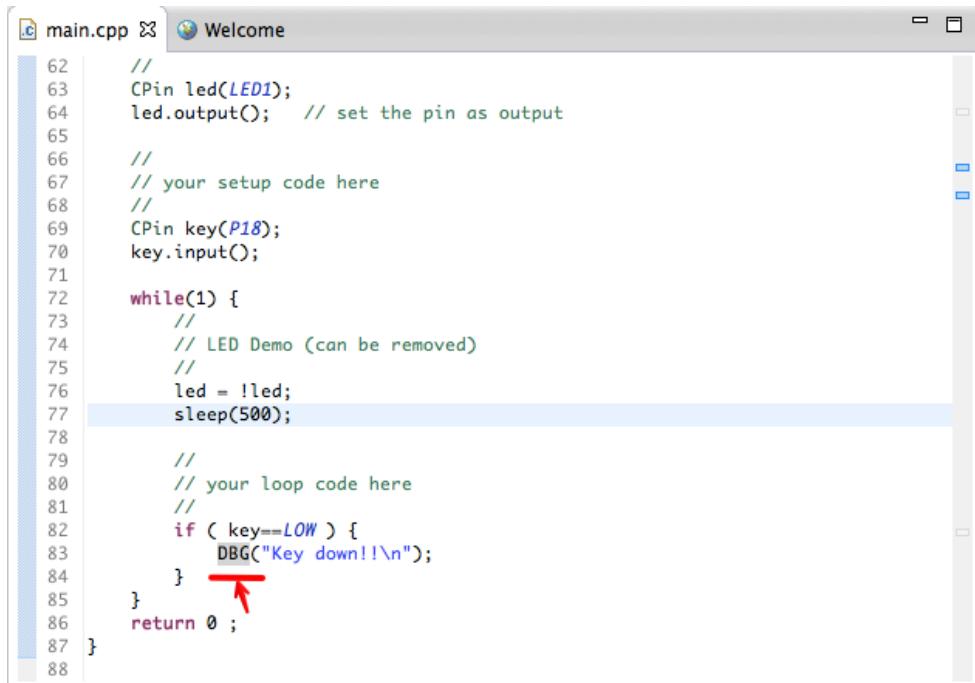
main.cpp  Welcome
Name : main.cpp

3 Name : main.cpp
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

```

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
Name : main.cpp

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

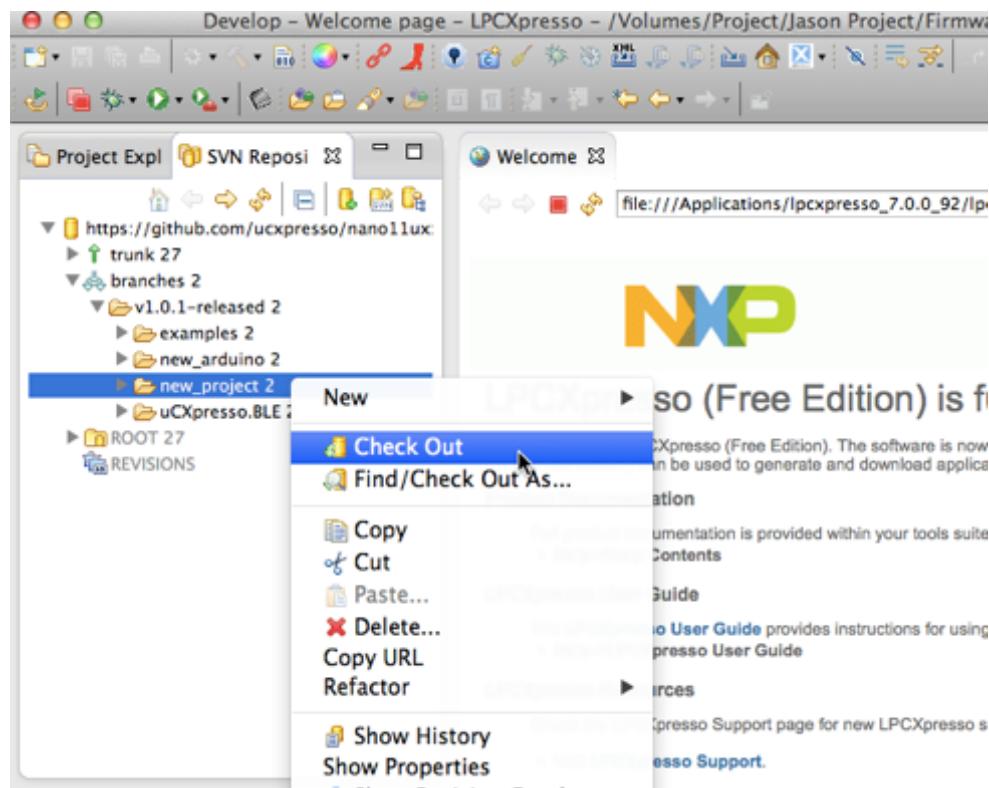
```

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

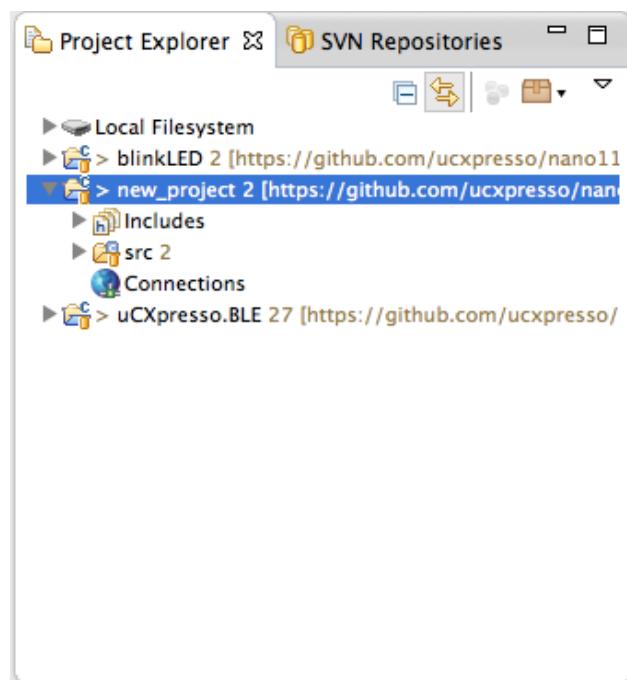
5. Create A New Project

5.1 Check Out the “new_project” template

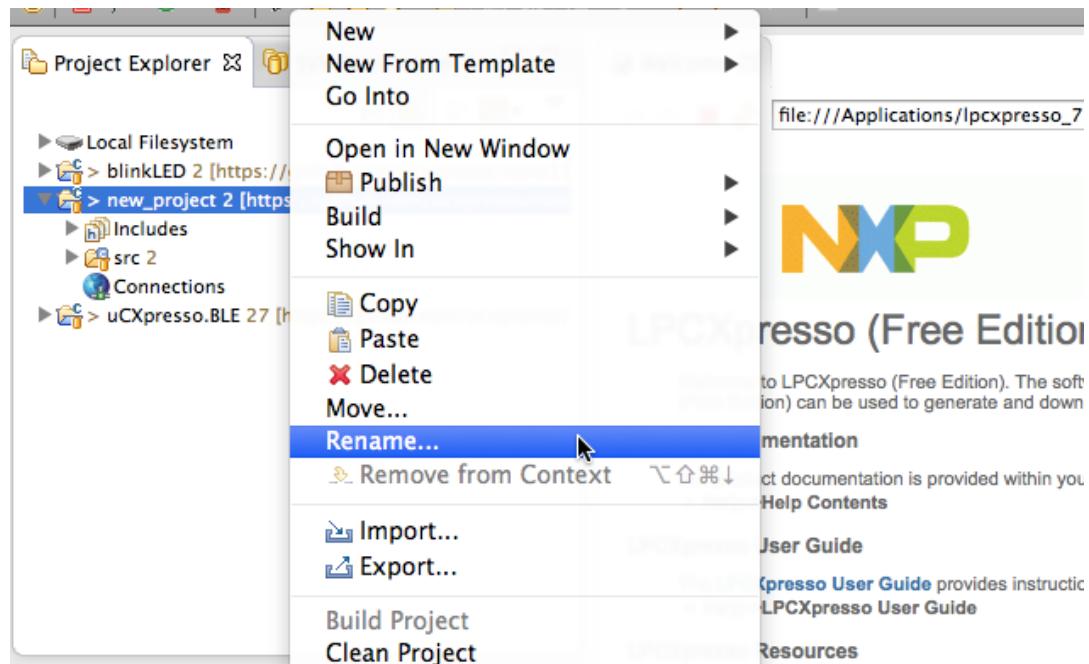
In SVN Repositories View, expand the three and “Check Out” the “new_project” to workspace.



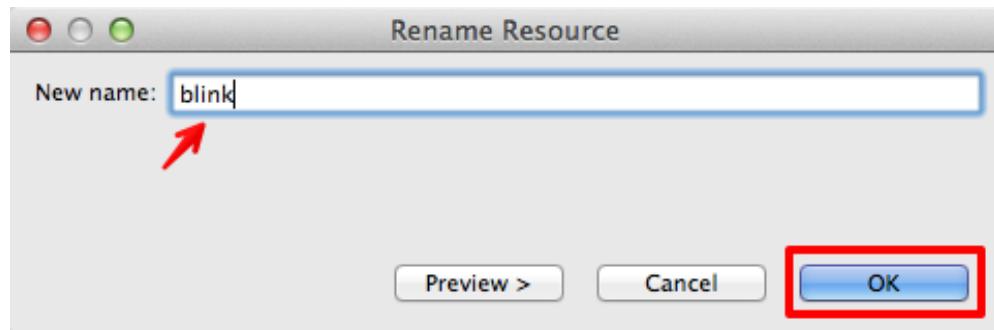
The “new_project” will show in workspace after Check Out.



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



For example to rename to “blink” as below,



uCpresso.NRF

An empty project “blink” is in the workspace.

The screenshot shows the LPCXpresso IDE interface. The top bar displays "Develop - blink/src/main.cpp - LPCXpresso - /Volumes/Project/Jason Project/Firmware/lpcxpresso/cortex-m0". The left sidebar has "Project Explorer" and "SVN Repositories" tabs, with "Local Filesystem" expanded to show a "blink 2" project from a GitHub repository. The main workspace shows the "main.cpp" file with the following code:

```
b1
62* ****
63*          your setup code here
64*
65* ****
66
67// LED Demo (can be removed)
68//
69DBG("Hello I'm in debug mode\n");
70uint8_t i = 0;
71C8us port(LED1, LED2, LED3, LED4, END);
72port.output(); // set all pins as output
73
74
75while(1) {
76    ****
77*          your loop code here
78* ****
79
80    //
81    // LED Demo (can be removed)
82    //
83    port = led_scripts[i];
84    i = (i+1) < sizeof(led_scripts) ? i+1 : 0;
85    sleep(100);
86
87}
88
89
90}
91
92}
93
94//
```

The bottom left shows a toolbar with icons for Quic, Variables, Breakpoints, Output, and Error. The bottom right shows a "Console" tab with "CDT Build Console [blink]" and a "NXP LPC11U37/40 (blink)" message.

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

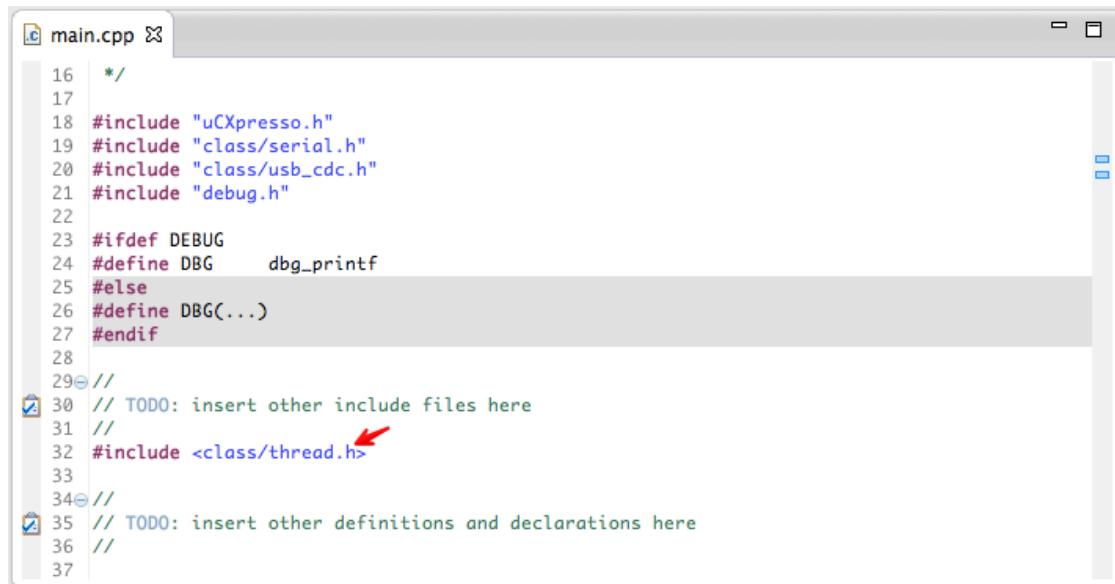
6. RTOS Thread (need to modify for nano51822)

6.1 Overview

This is an example of how to make four threads to control the LED1, LED2, LED3 and LED4 respectively.

6.2 Make your thread class

Includes the CThread class to provide the thread features.

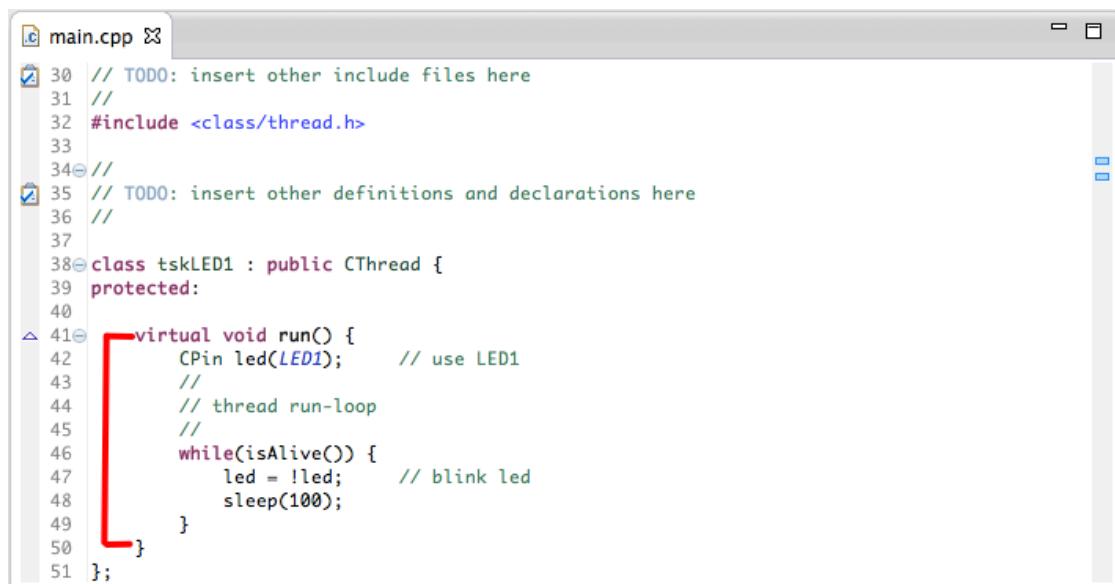


```

16 */
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 // TODO: insert other include files here
31 //
32 #include <class/thread.h> ←
33
34 //
35 // TODO: insert other definitions and declarations here
36 //
37

```

Creates a “tskLED1” thread and inherits from CThread class.



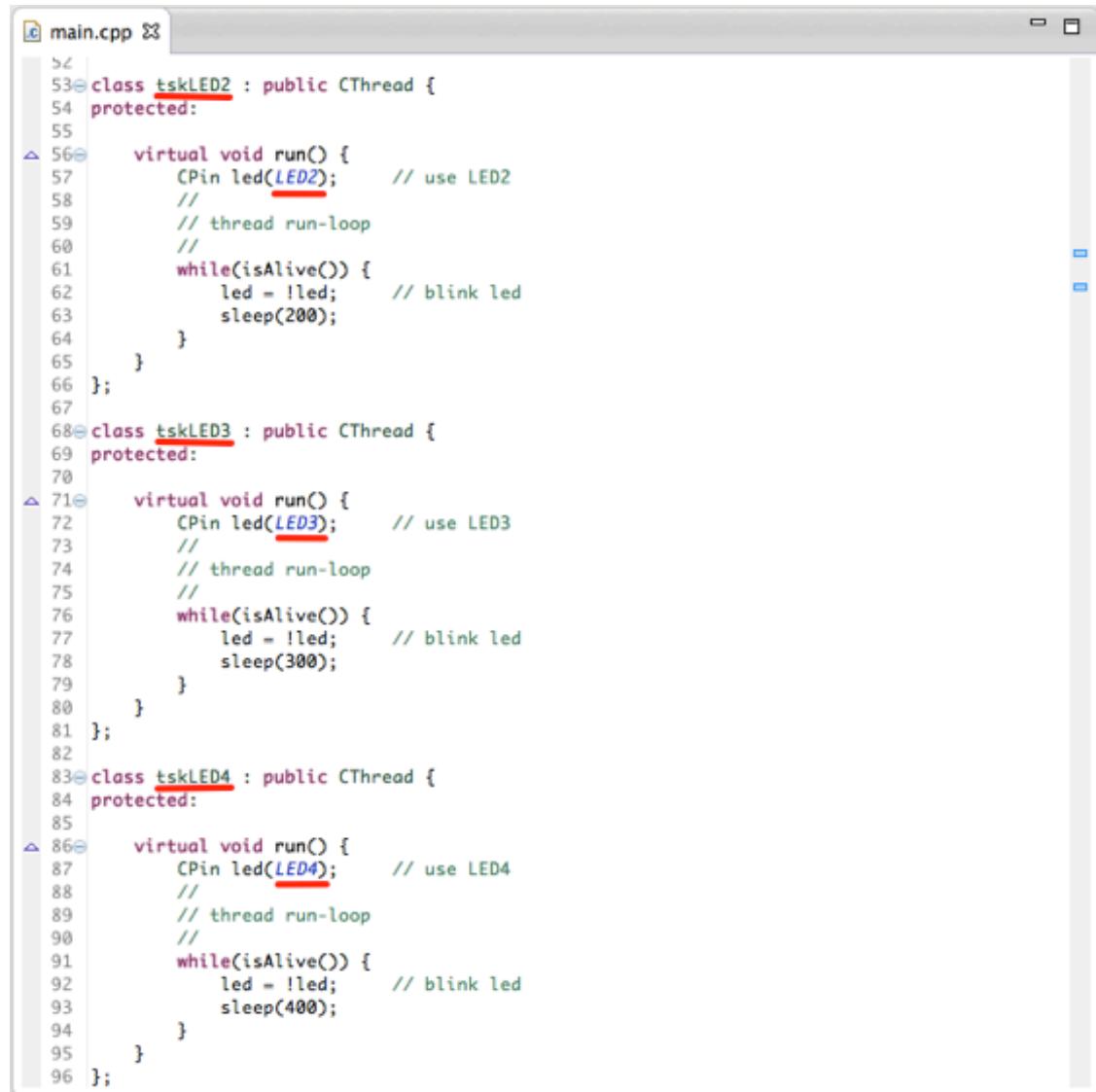
```

30 // TODO: insert other include files here
31 //
32 #include <class/thread.h>
33
34 //
35 // TODO: insert other definitions and declarations here
36 //
37
38 class tskLED1 : public CThread {
39 protected:
40
41     virtual void run() {
42         CPin led(LED1);      // use LED1
43         //
44         // thread run-loop
45         //
46         while(isAlive()) {
47             led = !led;      // blink led
48             sleep(100);
49         }
50     }
51 };

```

Implements the “virtual void run()” member function. and have to understand that “run()” member function is the life-cycle of thread. Put the isAlive() in the while-loop to check whether thread is actively or not.

In the same way, to make other tskLED2, tskLED3 and tskLED4 threads to control the different LEDs and different sleep-time.

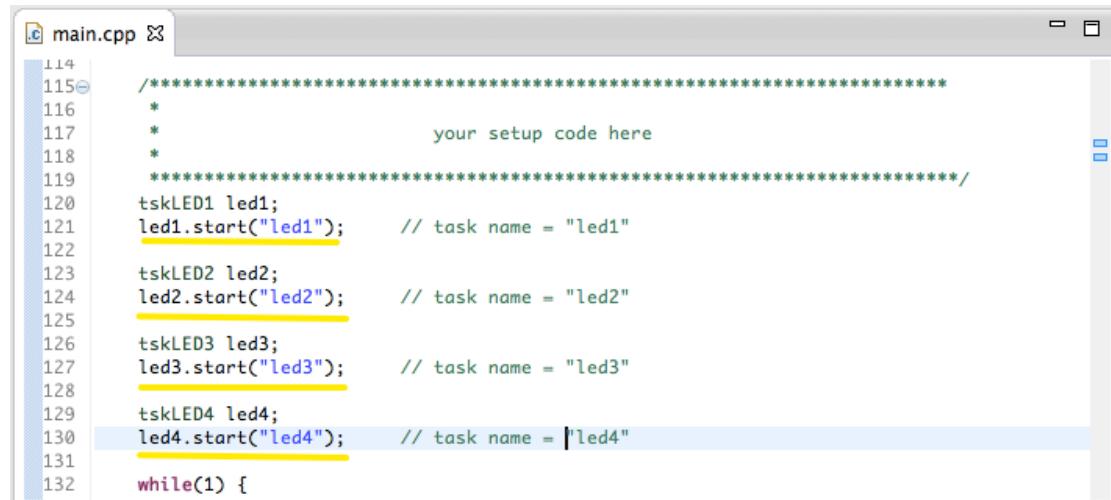


```

52
53 class tskLED2 : public CThread {
54 protected:
55
56 virtual void run() {
57     CPin led(LED2);      // use LED2
58     //
59     // thread run-loop
60     //
61     while(isAlive()) {
62         led = !led;      // blink led
63         sleep(200);
64     }
65 }
66 };
67
68 class tskLED3 : public CThread {
69 protected:
70
71 virtual void run() {
72     CPin led(LED3);      // use LED3
73     //
74     // thread run-loop
75     //
76     while(isAlive()) {
77         led = !led;      // blink led
78         sleep(300);
79     }
80 }
81 };
82
83 class tskLED4 : public CThread {
84 protected:
85
86 virtual void run() {
87     CPin led(LED4);      // use LED4
88     //
89     // thread run-loop
90     //
91     while(isAlive()) {
92         led = !led;      // blink led
93         sleep(400);
94     }
95 }
96 };

```

Start the threads



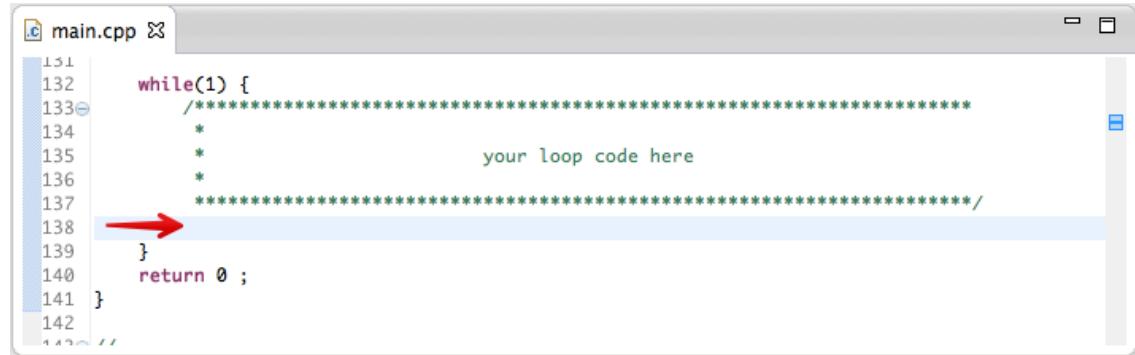
```

114 ****
115 *
116 *          your setup code here
117 *
118 ****
119
120 tskLED1 led1;
121 led1.start("led1");      // task name = "led1"
122
123 tskLED2 led2;
124 led2.start("led2");      // task name = "led2"
125
126 tskLED3 led3;
127 led3.start("led3");      // task name = "led3"
128
129 tskLED4 led4;
130 led4.start("led4");      // task name = "led4"
131
132 while(1) {

```

Remark:

The example uses the LED1~LED4, so you need to remove the all of the original LED demo code from main.cpp as below:



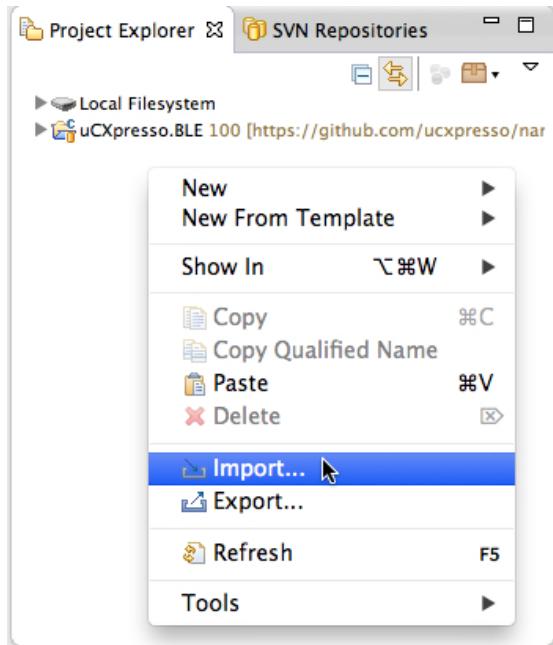
```
131
132     while(1) {
133         /*
134         *          your loop code here
135         *
136         */
137
138     }
139 }
```

To build the example and download the binary file to nano11U37. You will see that LED1, LED2, LED3 and LED4 are blinking standalone.

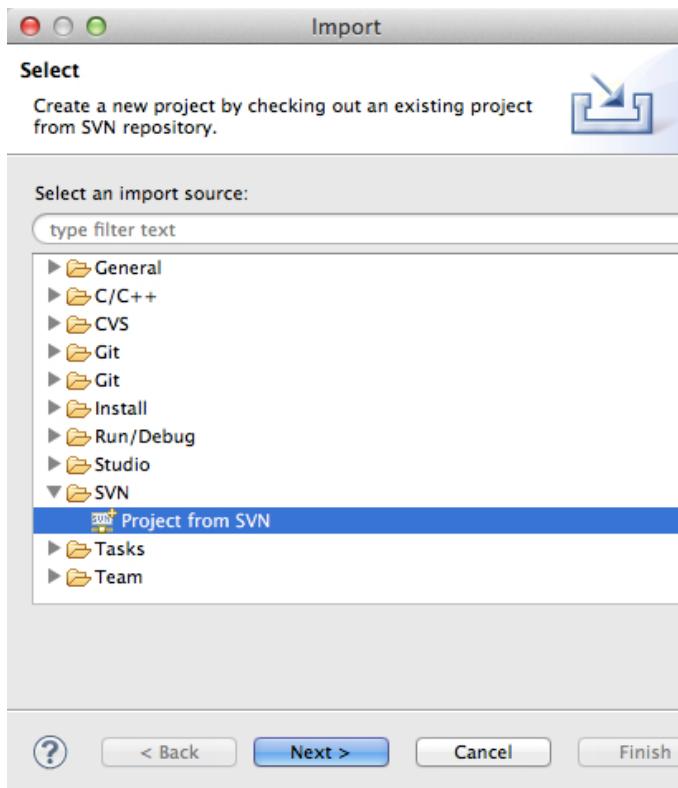
The final example “blink” project on the GitHub: /nano51822/examples/rtos :

Appendix A. Safe Way to Check Out the Example Projects

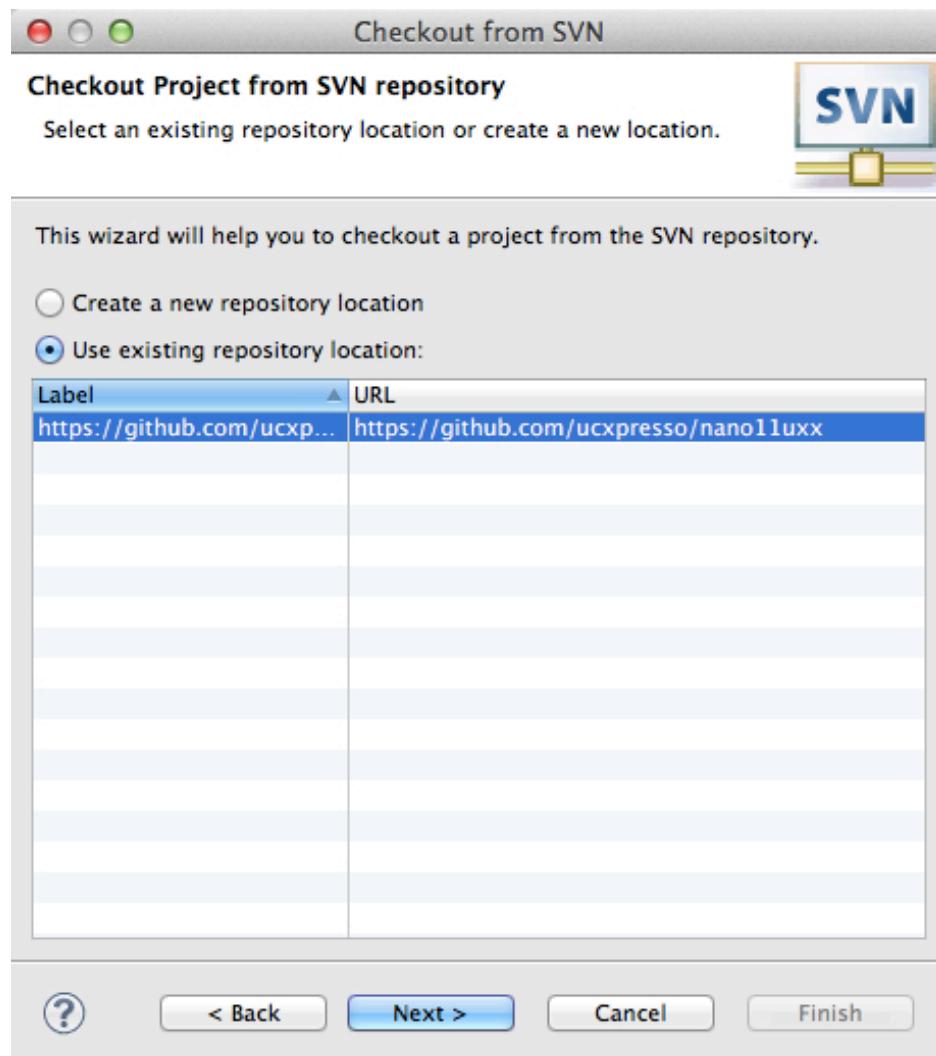
In “Project Explorer” view, click the mouse right button and show the drop-down menu, then select the “Import”.



Select “Project from SVN”.

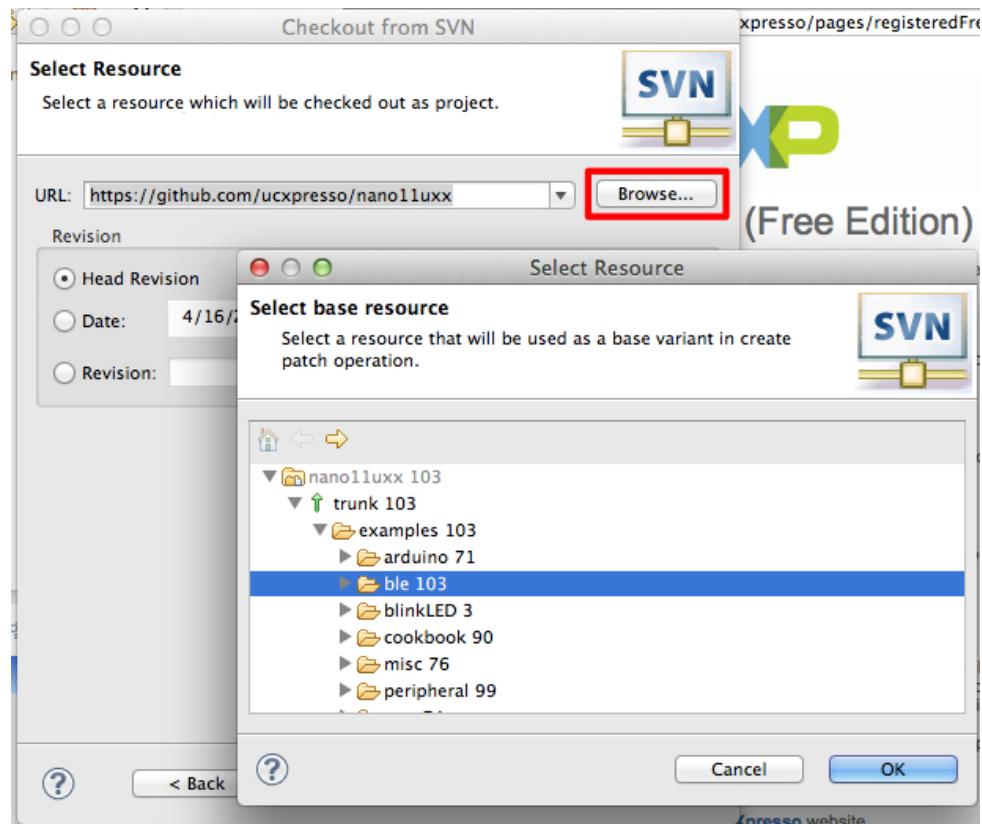


Use existing repository location:

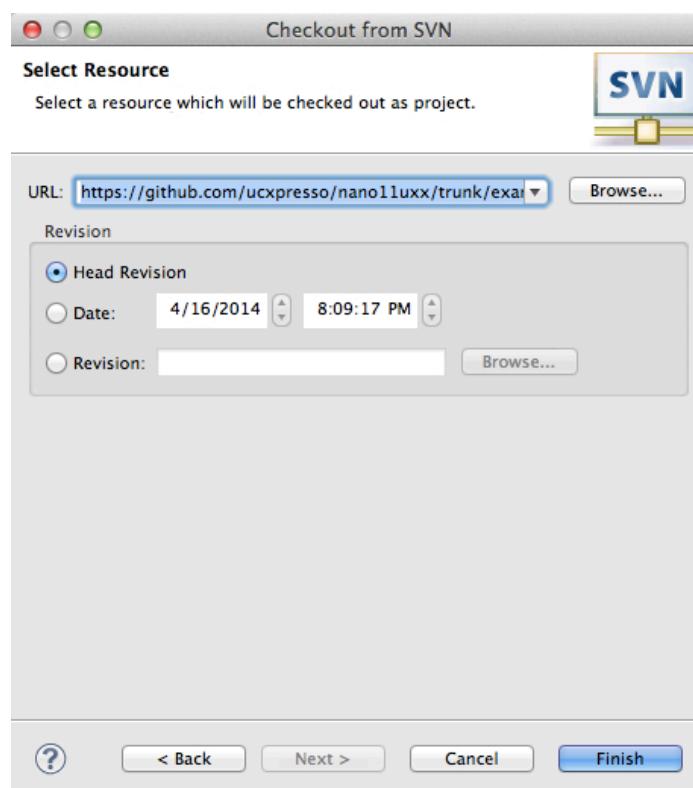


if you don't have our repository url, please select the "Create a new repository location", and input the <https://github.com/ucxpresso/nano11uxx> for new repository url.

Select SVN resource,
example, select the “ble” examples root path and click [OK].

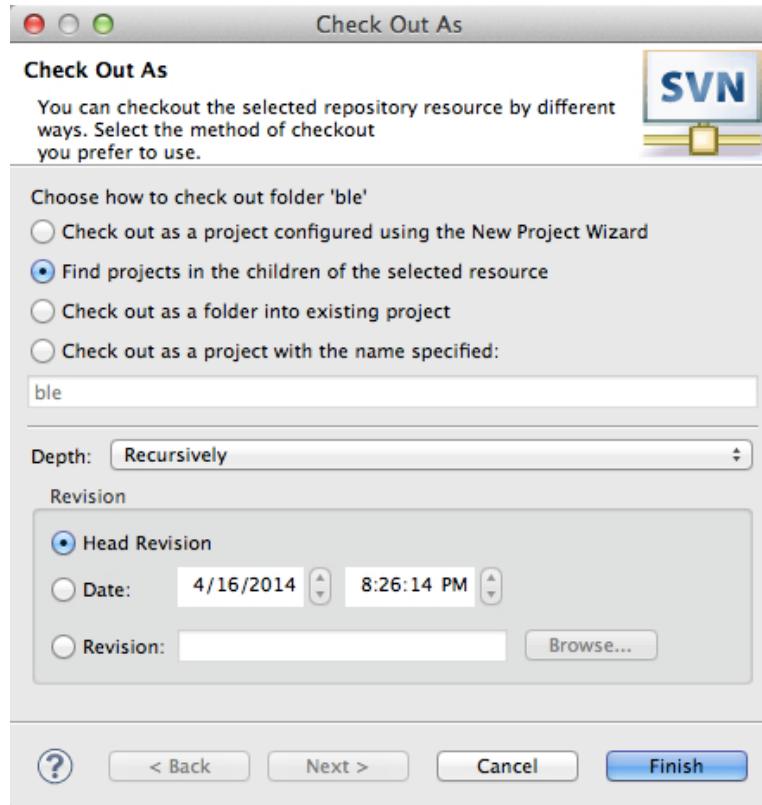


Click [Finish] to check out SVN project from the URL.

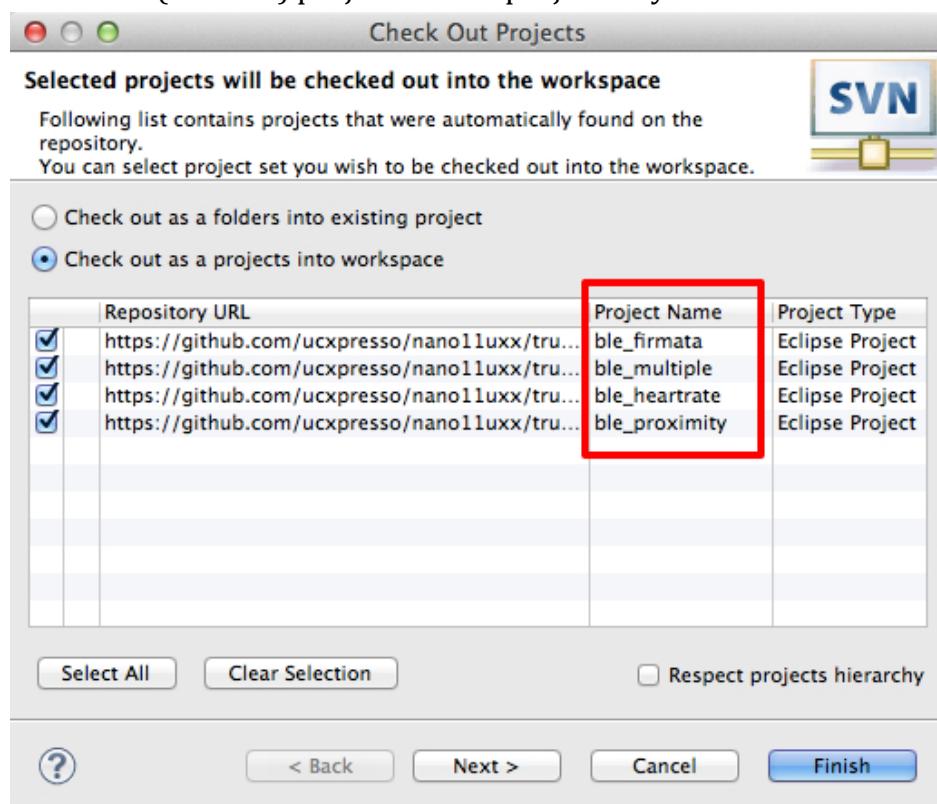


Check Out As,

select "Find projects in the children of the select resource".



Select one (or more) projects which project do you want.

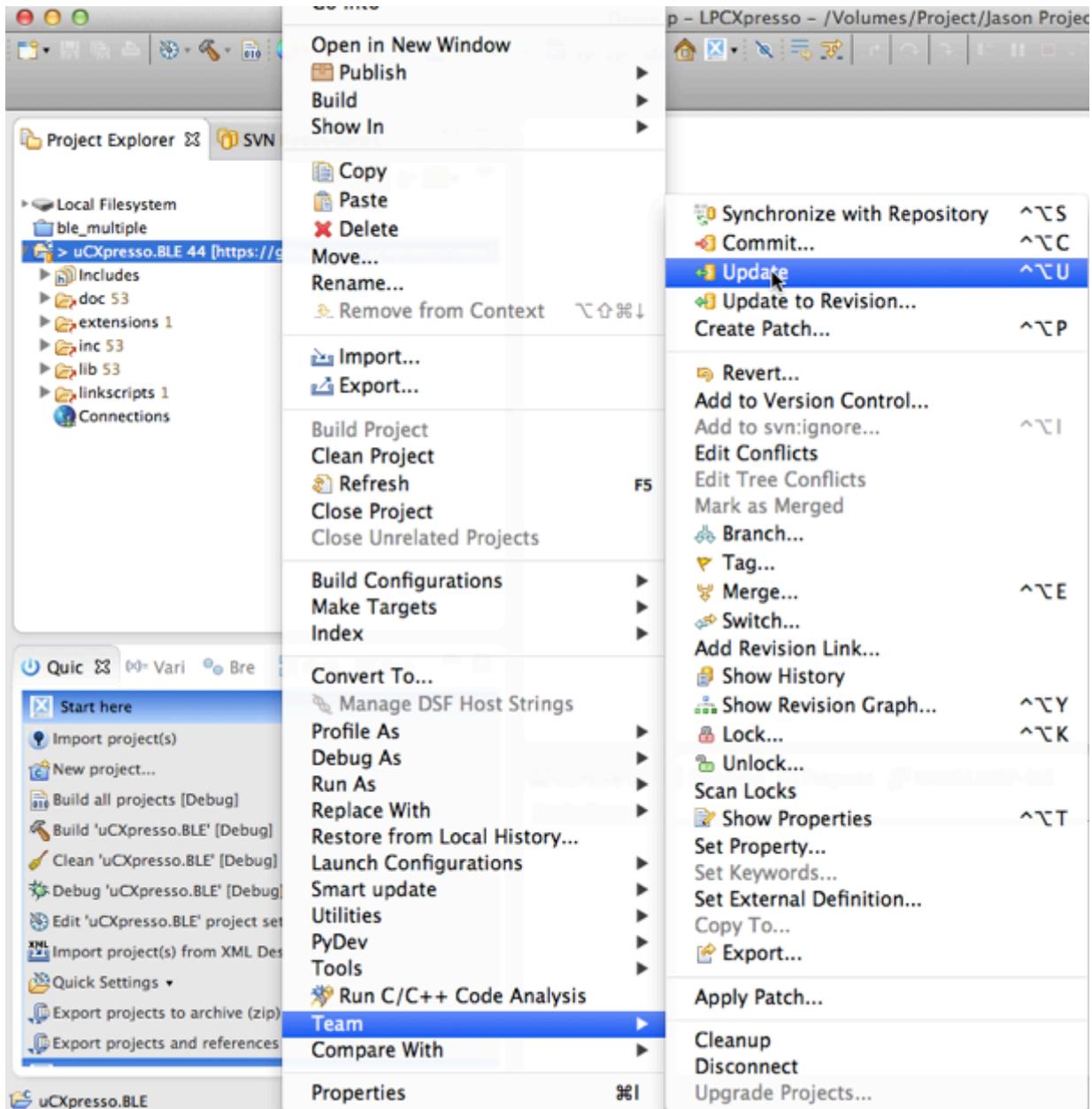


Click [Finish] to start to check out the selected projects into your workspace.

Appendix B. Update uCpresso.NRF framework

When your uCpresso.NRF is check-out from the [trunk](#) of SVN repository, you may use this appendix to update your uCpresso.NRF framework.

Click right button of mouse on the “uCpresso.NRF”, in drop down menu , select Team > Update

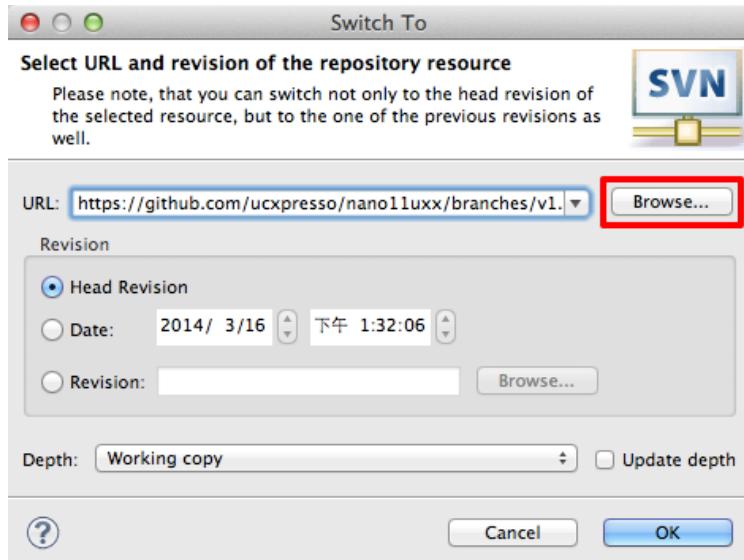


Appendix C. Switch uCpresso.NRF version Branch

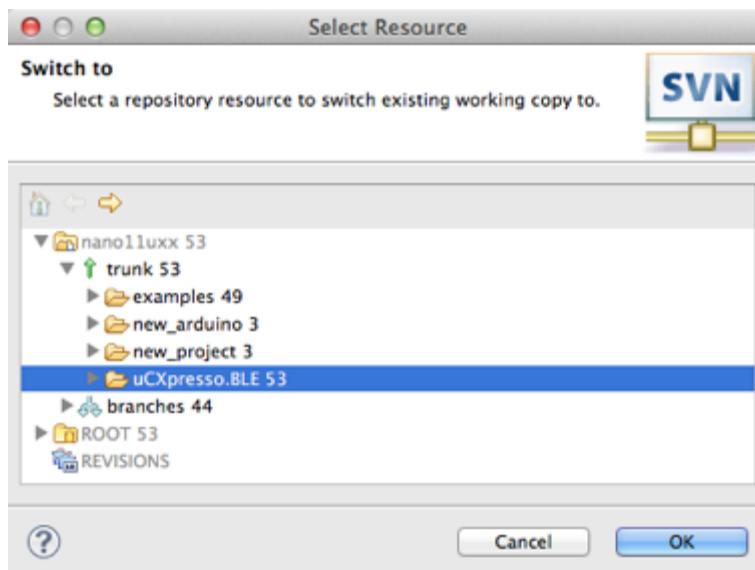
C.1 Switch SVN Branch

To switch the uCpresso.NRF version branch, click the right button of mouse on the “uCpresso.NRF” in Project Explorer View.

Select drop down menu “Team” > “Switch...” :



Click [Browse...] select resource (ex. select to trunk/uCpresso.NRF)



Click [OK] on “Select Resource” > Click [OK] on “Switch To”

Final, see the Appendix A to update the uCpresso.NRF framework.

Appendix D. Class Manual

D.1 Online Class Manual

<https://cdn.rawgit.com/ucpresso/nano51822/master/documents/doxygen/html/index.html>

The screenshot shows a web browser displaying the uCpresso.NRF doxygen-generated class manual. The title bar reads "uCpresso.NRF: bleBase". The main content area is titled "bleBase Class Reference" under the "Bluetooth" category. It includes a brief description, the header file "#include <class/ble/ble_base.h>", an inheritance diagram showing bleBase as a base class for bleAdvertising, bleDevice, bleDeviceManager, bleGAP, and bleService, which in turn have their own subclasses, and a list of public member functions: "virtual bool isThread ()" and "virtual bool isValid ()". The bottom right corner of the page footer indicates it was generated on Sun Oct 19 2014 16:41:05 for uCpresso.NRF by doxygen 1.8.8.

Appendix E. The uCpresso.NRF License

Hardware		nano51822	nRF51822
License type	Free	module	chip
Can use in commercial application	No	Yes	Yes
How number of the Task	1	Unlimited	Unlimited
How number of the BLE service	1	Unlimited	Unlimited
SoftDevice Version	Must be 7.1	7.1 or later	7.1 or later
App RAM size	6KB	Unlimited ¹	Unlimited ¹
App Code Size	Unlimited ¹	Unlimited ¹	Unlimited ¹
Peripherals framework (I/O)	Unlimited	Unlimited	Unlimited
Price	Free	Contact us	Contact us

Note:

1. Limit by chip condition.
2. [The nano51822 datasheet](#).
3. Another license requirement, please contact us : service@embeda.com.tw