

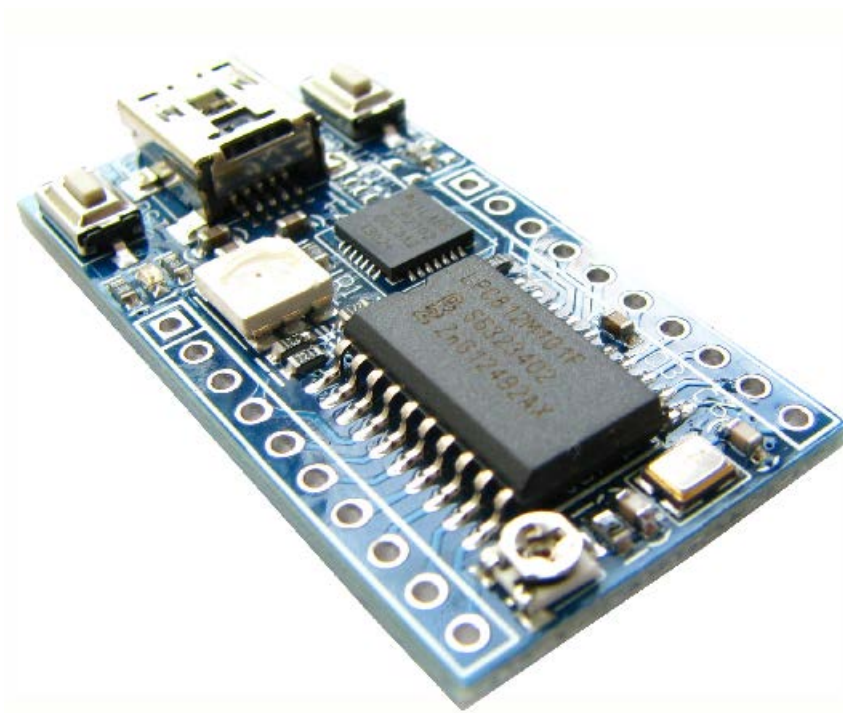
# LPC812M101 開発ボード マニュアル

株式会社日昇テクノロジー

<http://www.csun.co.jp>

[info@csun.co.jp](mailto:info@csun.co.jp)

作成・更新日 2013/09/22



copyright@2013

## ・ 修正履歴

NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2013/09/22

※ この文書の情報は、文書を改善するため、事前の通知なく変更されることがあります。  
最新版は弊社ホームページからご参照ください。「<http://www.csun.co.jp>」

※ (株)日昇テクノロジーの書面による許可のない複製は、いかなる形態においても厳重に禁じられています。



## 目次

第一章	概説	4
1.1.	チップファンクション	4
1.2.	ボードファンクション	5
第二章	Hardware layout and configuration	6
2.1.	Power supply	6
2.2.	Clock source	6
2.3.	Reset source	7
2.4.	USB-COM	7
2.5.	Potentiometer	7
2.6.	LED	7
2.7.	KEY	7
第三章	開発ツール Keil の応用	8
3.1.	Keil コンパイル環境	8
3.1.1.	Keil コンパイル環境構築	8
3.1.2.	エミュレータ設定	8
3.2.	シリアルポートでプログラムダウンロード	10
第四章	サンプルソース説明	12
4.1.	lpcopen_periph_8xx.uvmpw プロジェクト管理ファイルのプログラム説明	12
4.1.1.	¥periph_blinky	14
4.1.2.	¥periph_acmp	15
4.1.3.	¥periph_bod	15
4.1.4.	¥periph_clkout	16
4.1.5.	¥periph_crc	17
4.1.6.	¥periph_i2c	17
4.1.7.	¥periph_mrt	18
4.1.8.	¥periph_spi	18
4.1.9.	¥periph_uart	19
4.1.10.	¥periph_uart_rom	19
4.1.11.	¥periph_wkt	20
4.1.12.	¥periph_wwdt	20
4.2.	lpcopen_freertos_8xx.uvmpw プロジェクト管理ファイルのプログラム説明	21
4.2.1.	¥freertos_blinky	21

# 第一章 概説

## 1.1. チップファンクション

ARM Cortex-M0+ processor

- running at frequencies of up to 30 MHz
- Nested Vectored Interrupt Controller (NVIC)
- System tick timer.
- Serial Wire Debug (SWD) and JTAG boundary scan modes supported
- Micro Trace Buffer (MTB) supported.

Memory:

- Up to 16 kB on-chip flash (ISP and IAP via on-chip bootloader software)
- Up to 4 kB SRAM
- ROM API support: Boot loader, USART drivers, I2C drivers, Power profiles

Digital peripherals:

- High-speed GPIO interface with up to 18 General-Purpose I/O (GPIO) pins
- GPIO interrupt generation capability
- Switch matrix for flexible configuration of each I/O pin function
- State Configurable Timer (SCT)
- Multiple-channel multi-rate timer (MRT)
- Self Wake-up Timer (WKT)
- CRC engine.
- Windowed Watchdog timer (WWDT)

Analog peripherals:

- Comparator with external voltage reference

Serial interfaces:

- 3 USART interfaces
- 2 SPI controllers
- I2C-bus interface

Clock generation:

- 12 MHz internal RC Oscillator (IRC)
- Crystal Oscillator (SysOsc) with operating range of 1 MHz to 25 MHz
- Programmable watchdog oscillator with a frequency range of 9.4 kHz to 2.3 MHz
- 10 kHz low-power oscillator for the WKT
- PLL allows CPU operation up to the maximum CPU rate
- Clock output function with divider that can reflect various clocks

Power control:

- Integrated PMU (Power Management Unit)
- Reduced power modes: Sleep mode, Deep-sleep mode, Power-down mode, Deep power-down mode
  - Power-On Reset (POR)
- Brownout detect

Unique device serial number for identification

Single power supply



## 1.2. ボードファンクション

USB シリアル変換機能搭載、FlashMagic を介しプログラムをダウンロードサポート

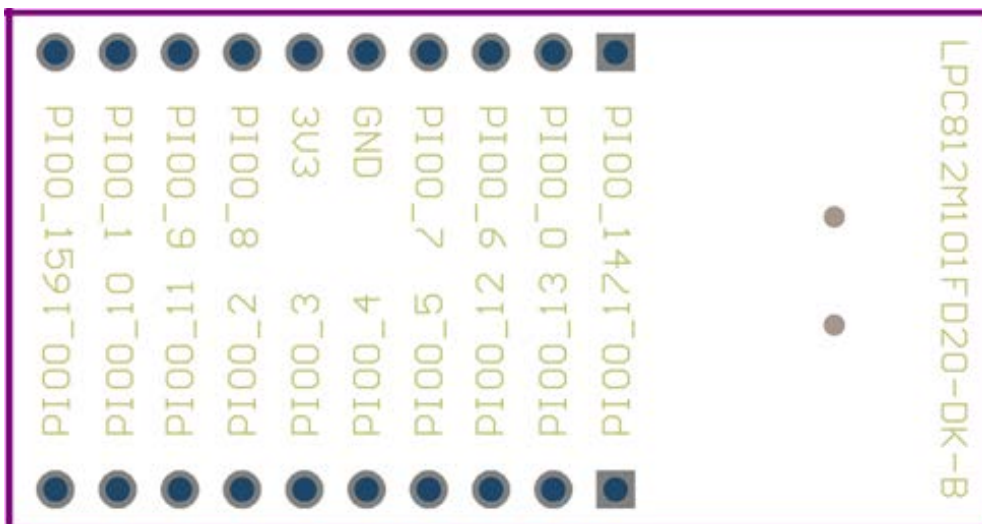
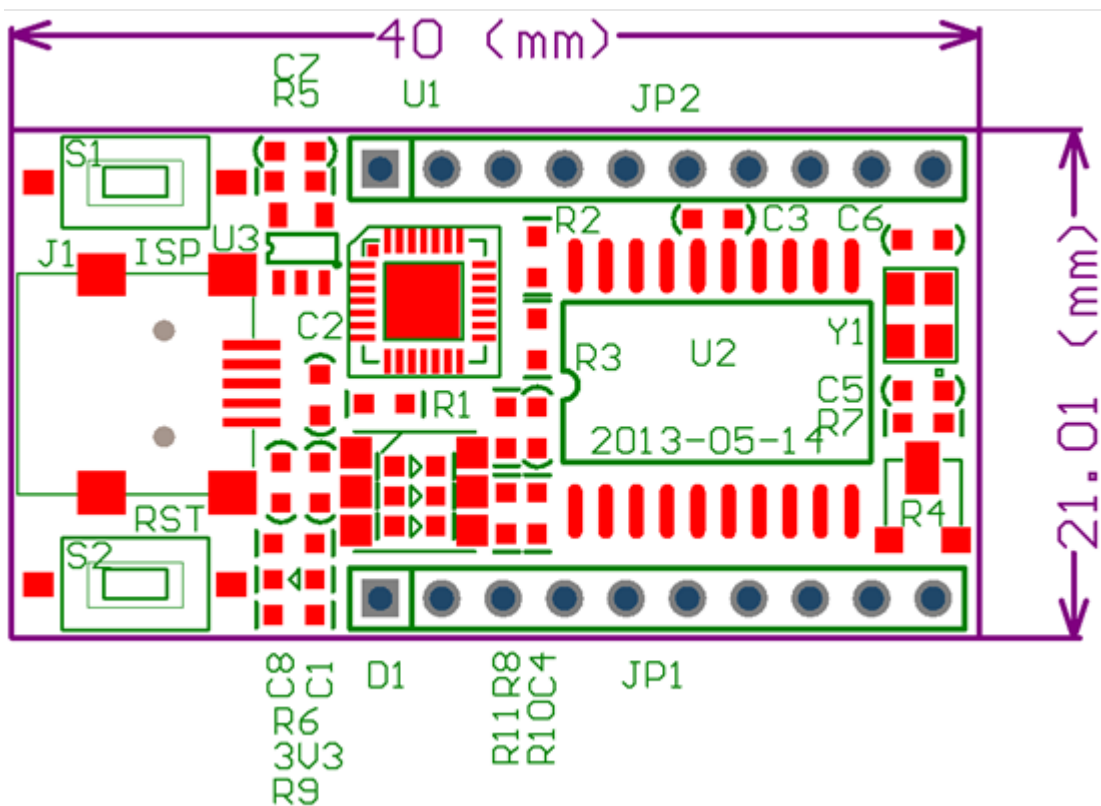
RGB LED

1 つファンクションキー

1 つ変調抵抗

全ての I/O を引き出し

## 第二章 Hardware layout and configuration



### 2.1. Power supply

開発ボードはMiniUSB インタフェースで給電する、同時に USB 仮想シリアルポートとしても使用する。

### 2.2. Clock source

開発ボードは 12MHz 外部水晶振動器をプロセッサのクロックリソースとする。

## 2.3. Reset source

開発ボードはローレベルリセット、リセットするには2つの方法がある：

- ボードの S2 ボタンを押す
- JP1-P4

## 2.4. USB-COM

開発ボードは USB シリアルポート変換チップ CP2102 を使用し、プロセッサの USART0 と接続し、シリアルポート機能を実現する。

## 2.5. Potentiometer

開発ボードに 10K の変調抵抗があり、PI00\_0 と接続し、コンパレータを行う実験に使用する。

## 2.6. LED

開発ボードに 1 つの RGB 三色 LED があり、PI00\_7、PI00\_17、PI00\_16 と接続し、ローレベルで点灯する。

## 2.7. KEY

開発ボードには 2 つのキーがあり、RST はリセットキー、ISP はプロセッサが ISP モードに切り替えキーで、通常キーにも使用できる。

## 第三章 開発ツール Keil の応用

### 3.1. Keil コンパイル環境

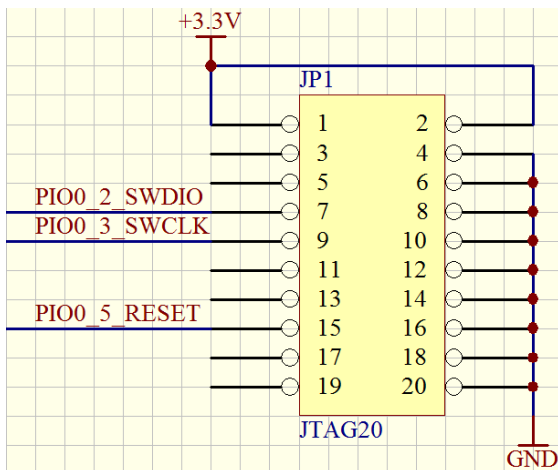
#### 3.1.1. Keil コンパイル環境構築

Keil 社の HP (<http://www.keil.com/>) から最新版（無償評価版）がダウンロードできます。完全機能を使用するには License が別途で購入する必要があります。

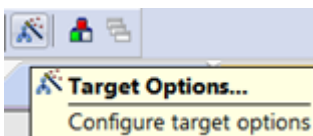
#### 3.1.2. エミュレータ設定

本開発ボードにエミュレータインタフェースがないため、エミュレータを使用するにはピン配列変換ケーブルで下記の通りに接続する必要：

エミュレータの 20 ピンソケット	開発ボード JP1、JP2 ピン	プロセッサのピン名
7	JP1-7	PI00_2_SWDIO
9	JP1-6	PI00_3_SWCLK
15	JP1-4	PI00_5_RESET
1 or 2	JP2-6	+3.3V
4、6、8、10、12、16、18、20 中の 1 ピン	JP2-5	GND

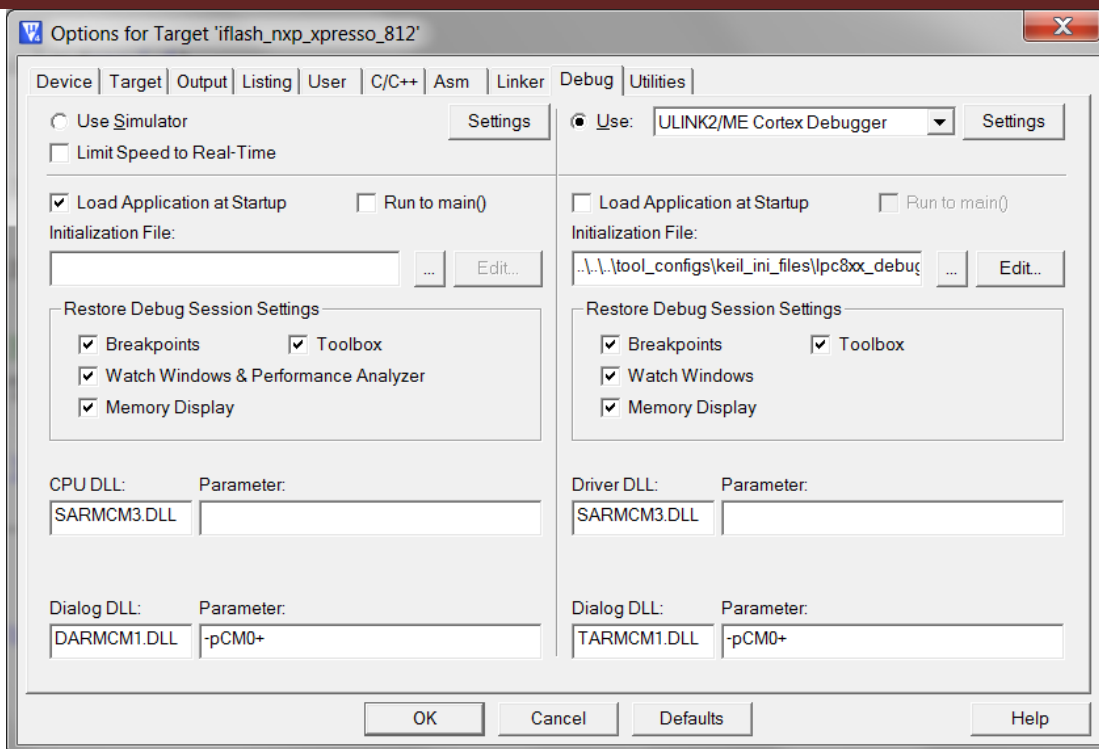


- `Options for Target`

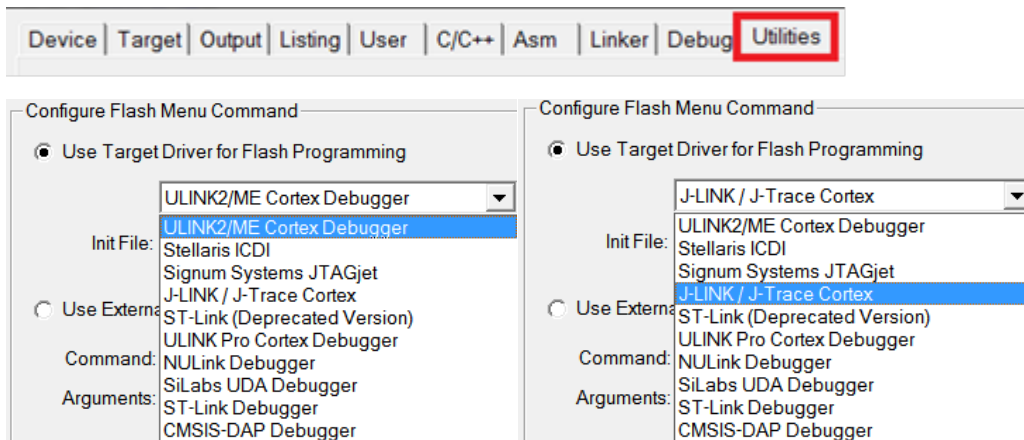


- 下記のウィンドウ：

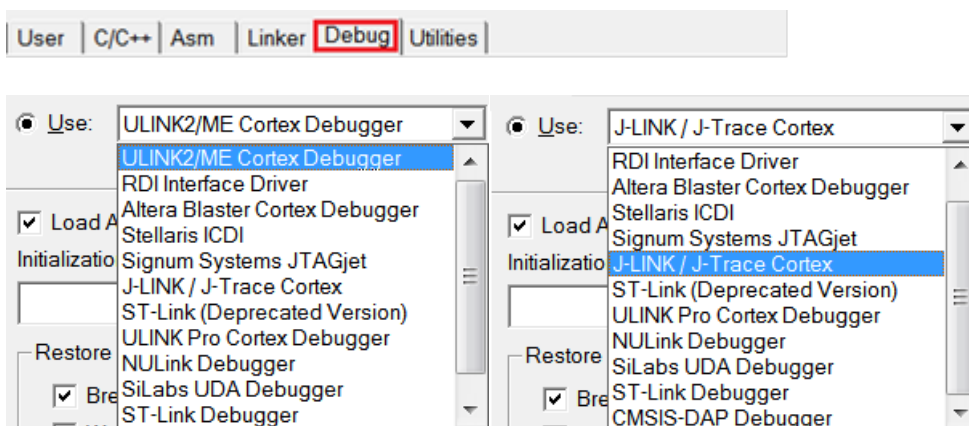




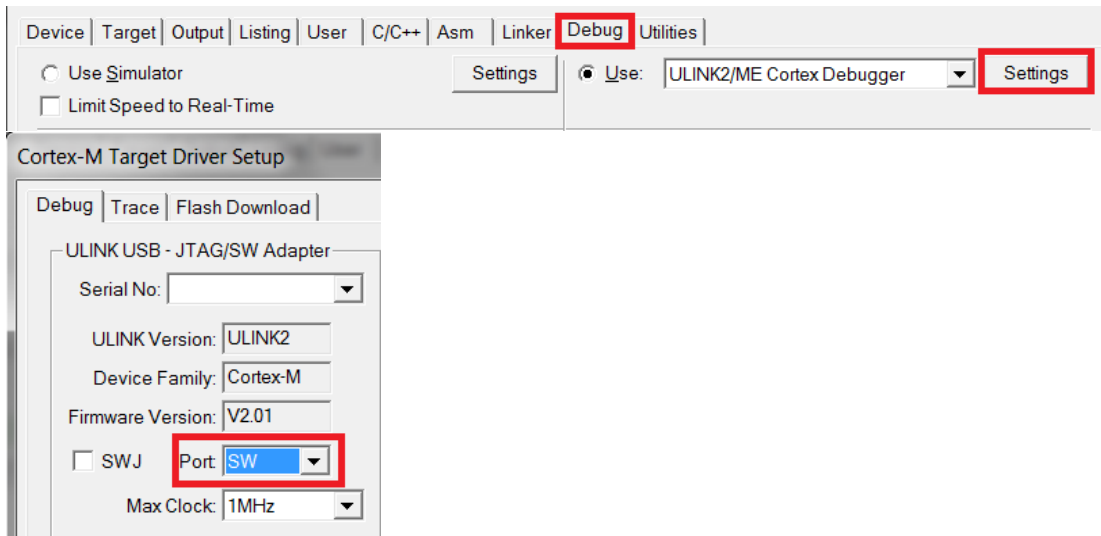
- `Utilities` 選択、`Use Target Driver for Flash Programming` でエミュレータタイプ選択、ULINK2 を使用する場合は、`ULINK2/Me Cortex Debugger` 選定；JLINK V8 を使用する場合は、`J-LINK/J-Trace Cortex` 選定。



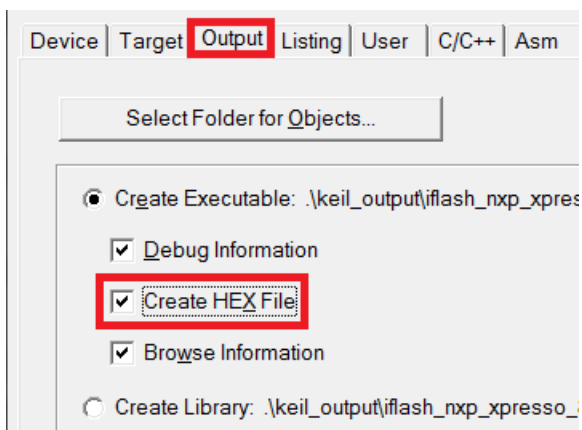
- `Debug` 選択、Use リストでエミュレータ選択：ULINK2 を使用する場合は、`ULINK2/Me Cortex Debugger` 選定；JLINK V8 を使用する場合は、`J-LINK/J-Trace Cortex` 選定。設定完了後、エミュレータでプログラムをエミュレータできる。



- `Debug` で `Settings` をクリック、次のウィンドウで `Debug` →Port SW を選定



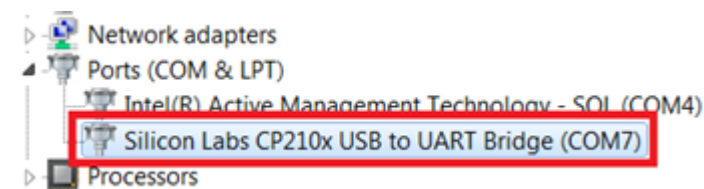
- `Output` → `Creat HEX File` 有効、FlashMagic を介し HEX ファイルをダウンロードできる。



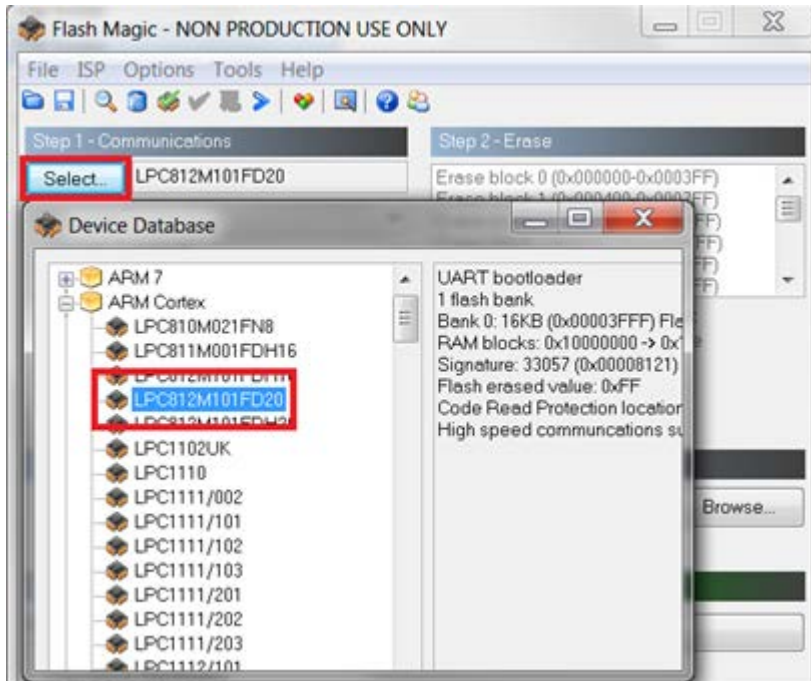
### 3.2. シリアルポートでプログラムダウンロード

- Tools ディレクトリにある FlashMagic.exe をインストール、仮想シリアルポートドライバ CP210x\_VCP\_Win\_XP\_S2K3\_Vista\_7.exe をインストール。
- MiniUSB を介し、PC と開発ボードを接続、デバイスマネージャを開き、仮想シリアルポートの割り当てのポートを確認する。PC により、仮想ポートの COM は異なる。

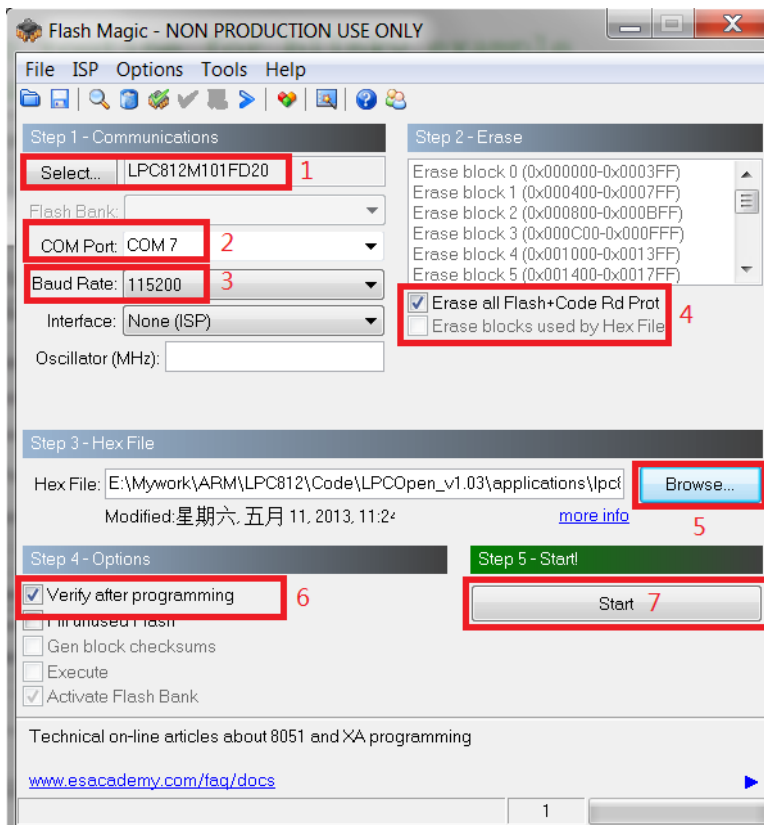
例 COM7 :



- ISP を押しながら、RST キーを押して開発ボードをリセット、プロセッサが ISP モードに入る。
- FlashMagic を実行、Select でプロセッサを選定する。



- COM Port で PC と開発ボード通信 COM を選定(本実験は COM7)、ボーレート 19200~115200、Browse で HEX のプログラムを選定、Step 4 の `Verify after program` を有効、Start をクリック、プログラムを Flash にダウンロード。完了後、開発ボードの RST でリセット、プログラム実行する。

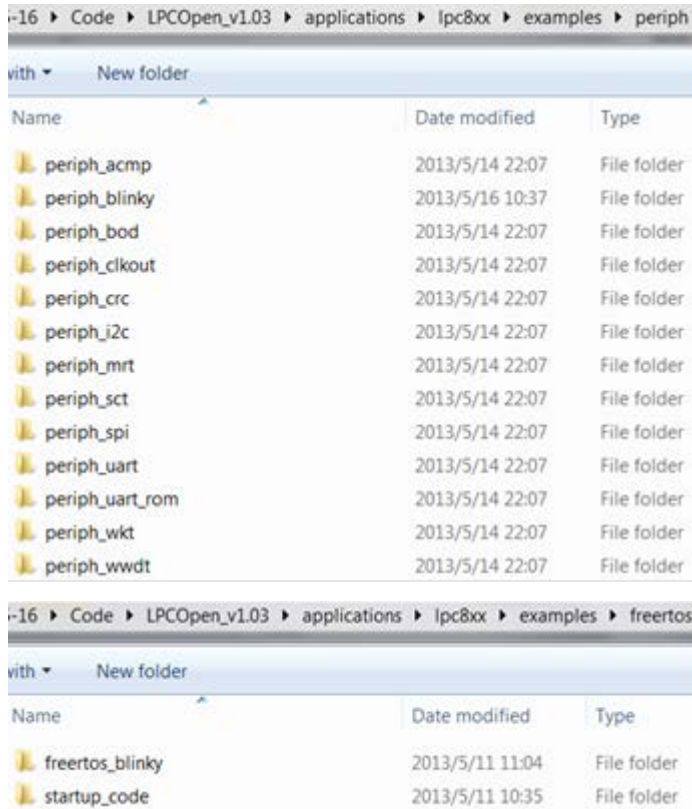


## 第四章 サンプルソース説明

¥Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥examples¥periph

¥Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥examples¥freertos

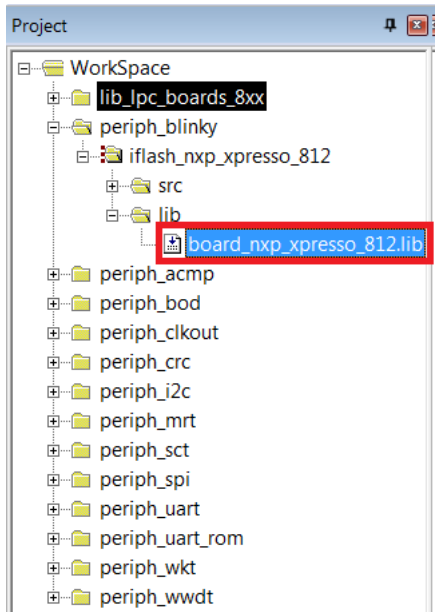
2つのディレクトリはプロセッサの外部ドライブレ例及びFreeRTOSのLEDフラッシュ例である。複数プロジェクトを管理するため、¥Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥keil\_uvision\_projectsディレクトリにlpcopen\_freertos\_8xx.uvmpw及びlpcopen\_periph\_8xx.uvmpwのプロジェクト管理ファイルを追加する。この二つのプロジェクトをオープンして全てのプロジェクトをコンパイル出来る。



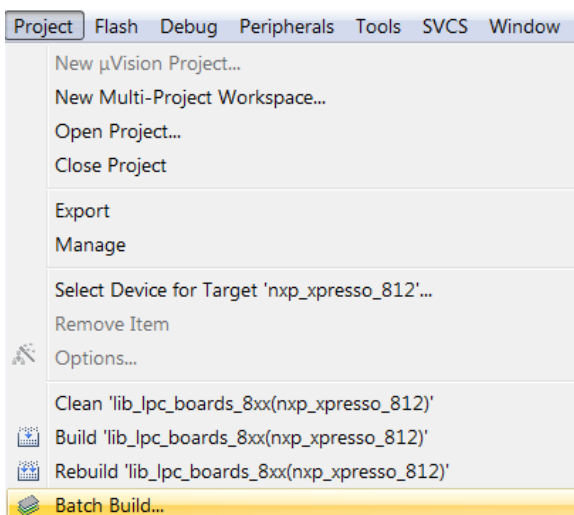
### 4.1. lpcopen\_periph\_8xx.uvmpw プロジェクト管理ファイルのプログラム説明

- プロジェクト管理ファイルは¥Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥keil\_uvision\_projectsディレクトリにあり、lpcopen\_periph\_8xxはプロジェクト管理ファイル。

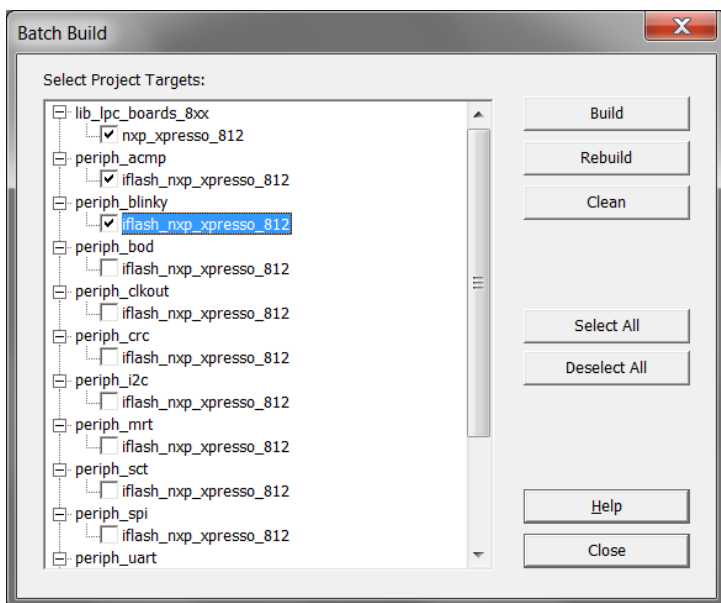
下記図の通り、Workspaceウィンドウで管理プロジェクトを確認できる、lib\_lpc\_board\_8xxプロジェクトファイルはboard\_nxp\_esspresso\_812.libファイルを生じ、libファイルは他のプロジェクトファイルが必要なデバイスドライブレのAPI関数を提供する。なので他のプロジェクトを独立コンパイルするには、board\_nxp\_esspresso\_812.libファイルを確認する必要がある。



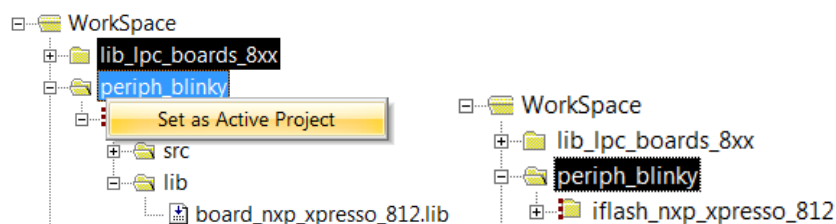
- lpcopen\_periph\_8xx.uvmpw プロジェクト管理ファイルにある全てのプロジェクトファイルを一回で全部コンパイル出来、また単独コンパイルも出来る。Project->Batch Build でコンパイルプロジェクトを管理する。



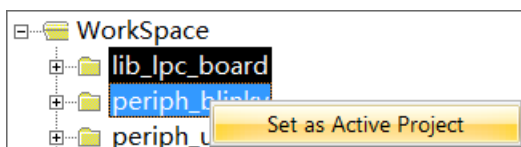
- Deselect All、デフォルトクリア、編集プロジェクトを選定する。Build または Rebuild をクリック、選択したファイルを同時にコンパイル出来る。



- 通常は1つファイルを選択、単独コンパイルする。例 periph\_blinky、右クリックし、Set as Active Project、プロジェクトファイルを青から黒に変化し、現在アクティブなプロジェクトとなる。



#### 4.1.1. ¥periph\_blinky



このプロジェクトは簡単なLEDドライブプログラム、システムタイマーを使用し、割り込みプログラムで開発ボードの青いLEDをフラッシュさせる。

- まずは Board\_Init() 関数でシステムタイマー、三つのLED制御ピンを初期化する。次に SysTick\_Config でシステムタイマーの割り込み周波数を設定する。システムタイマープログラムは Board\_LED\_Toggle(0) を呼び出して青いLEDを値の反転動作し続ける。

```
void SysTick_Handler(void)
{
    Board_LED_Toggle(0);
}

int main(void)
{
    Board_Init();
    Board_LED_Set(0, false);
    /* Enable SysTick Timer */
```

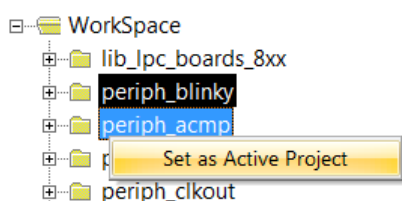
```

SysTick_Config(SystemCoreClock / TICKRATE_HZ);
while (1) {
    __WFI();
}
return 0;
}

```

- 生成した HEX ファイルは¥Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥examples¥periph¥periph\_blinky¥keil\_output¥iflash\_nxp\_xpresso\_812 ディレクトリに保存する。

### 4.1.2. ¥periph\_acmp

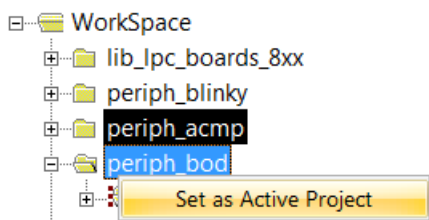


本プロジェクトのプログラムはアナログコンパレータを実演する。プログラムは PI00\_0 をコンパレータのプラス (+) 電圧入力に設定し、チップ内部の 0.8V バンドギャップ基準電圧は、コンパレータのマイナス (-) 電圧入力と設定し、PIN0.15 はコンパレータの出力と設定する。変調抵抗の抵抗値を変化させて PI00\_0 の電圧を調整する。電圧 < 0.8V な場合開発ボードの青い LED 点灯、PIN0.15 ピンは 0 を出力；電圧 > 0.8V な場合、LED 消灯、PIN0.15 ピンは 1 を出力。

- HEX ファイルは  
¥Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥examples¥periph¥periph\_acmp¥keil\_output¥iflash\_nxp\_xpresso\_812 ディレクトリ下に保存する。

注：プログラムを再ダウンロードする前に、変調抵抗を調整し、LED を消灯させる必要がある。

### 4.1.3. ¥periph\_bod

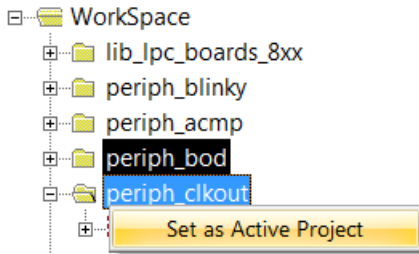


低電圧検出の例である。開発ボードの電圧が低すぎな場合、BOD 割り込み生成、割り込みサービスプログラムは青い LED 値を反転動作する。

- プログラム実行後、開発ボードの USB ケーブルを抜いたら、青い LED が 1 回点灯する、開発ボードの電源に大容量コンデンサを接続すると青い LED の点滅を確認できる。
- HEX ファイルは  
¥Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥examples¥periph¥periph\_bod¥keil\_output¥iflash\_nxp\_xpresso\_812 ディレクトリ下に保存する。

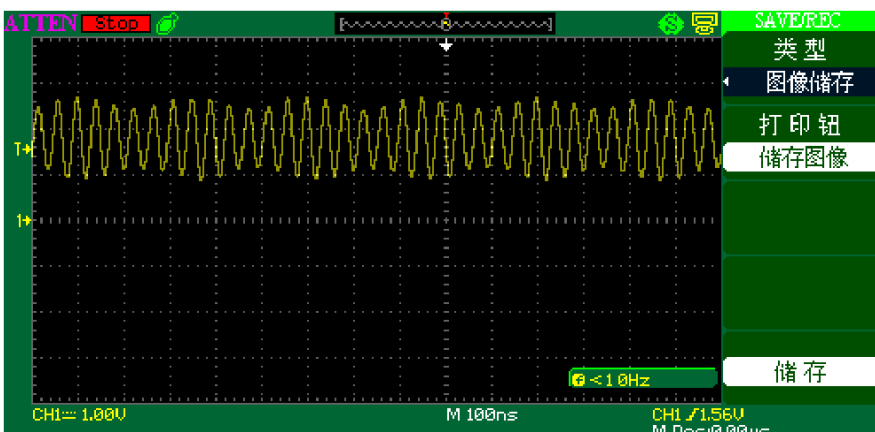
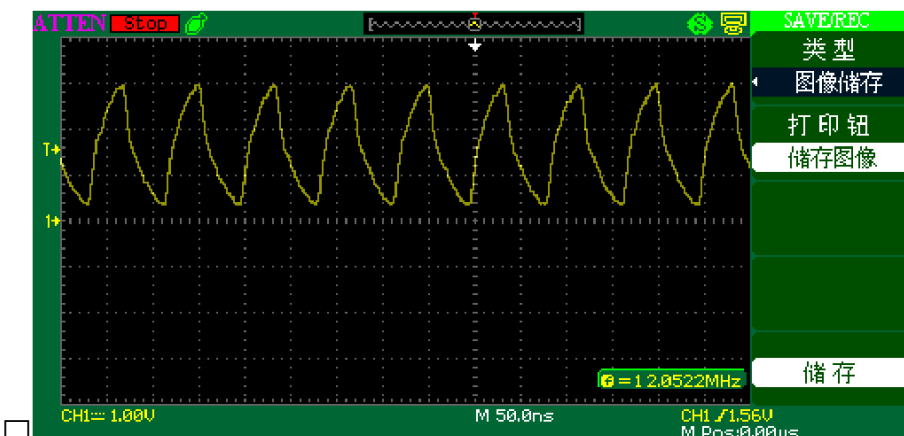


#### 4.1.4. ¥periph\_clkout



SYSCTL ドライバーを介し、プロセッサの異なるクロックリソースを通常ピンに引き出す実例である。

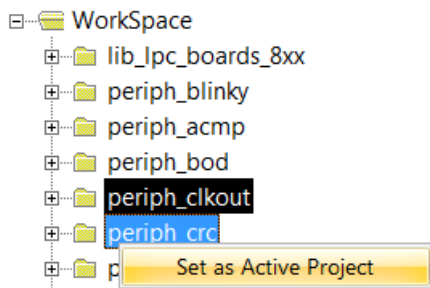
- プログラムは Chip\_Clock\_SetCLKOUTSource 関数で出力クロックを設定する。クロックリソースは下記の通り：SYSCTL\_CLKOUTSRC\_IRC、SYSCTL\_CLKOUTSRC\_SYSOSC、SYSCTL\_CLKOUTSRC\_WDTOSC、SYSCTL\_CLKOUTSRC\_MAINSYSCLK。
- プログラム実行後、開発ボードの青い LED が点滅する、オシロスコープで PI00\_1 ピンを測定、クロック出力を観察する。数秒後 LED は消灯し、PI00\_1 も出力波形を停止する



- HEX ファイルは  
 ¥Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥examples¥periph¥periph\_clkout¥keil\_output¥iflash\_nxp\_xpresso\_812 ディレクトリ下に保存する。



## 4.1.5. ¥periph\_crc

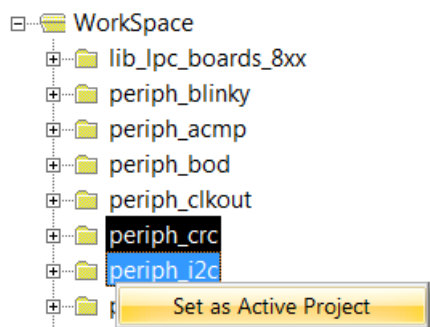


チップ内部の CRC エンジンを使用し、8、16、32 ビットの CRC パリティチェックを行う実例である。

プログラムループは順序に 8、16、32 ビットデータを CRC パリティチェックを行い、結果を別データ組に保存し、奇数回数は正確動作、偶数回数は誤動作。最後に CRC パリティチェックの結果と予測結果をもう一度 32 ビットのパリティチェックを行い、2 回の CRC パリティチェック結果を比較し、正確であれば青い LED 点灯、間違であれば LED 消灯。奇数回数は正確、偶数回数は間違い動作なのでプログラム実行後、青い LED は点滅し始める。

- HEX ファイルは  
`¥Code¥LPCOpen_v1.03¥applications¥lpc8xx¥examples¥periph¥periph_crc¥keil_output¥iflash_nxp_xpresso_812` ディレクトリ下に保存する。

## 4.1.6. ¥periph\_i2c

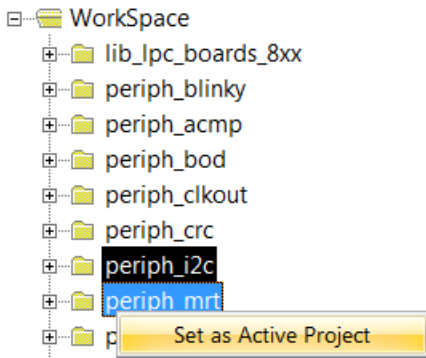


I2C メインコントローラーとスレーブ機能を実現する例。2 つの開発ボードが必要とする。1 つの開発ボードはメインコントローラープログラムをダウンロードし、1 つの開発ボードはスレーブドライバをダウンロード。マクロ `I2C_MASTER_MODE=1` メインコントローラープログラム実行、`I2C_MASTER_MODE=0` スレーブプログラム実行する。

- テストする前に、2 つの開発ボードの P0.10 (SDA) 同士、P0.11 (SCL) 同士を接続、2 つの開発ボードの GND も接続する。1 つボードの SDA 及び SCL ピンに 4.7K のプルアップ抵抗と接続する必要。
- メインコントローラーのプログラムは先ずクエリモードで 7-bit サブアドレスでスレーブデバイスプログラムの開発ボードに 1 バイトのデータを書き込み、読み取り動作を実行し、データを比較する。
- メインコントローラープログラムは割り込みモードで同じくデータを書き込み、読み取り、比較する。
- 最後にメインコントローラーは割り込みモードで 10-bit サブアドレス方式で書き/読み/比較する。
- 全ての動作が正しく実行すると、開発ボードの青い LED 点灯。
- HEX ファイルは  
`¥Code¥LPCOpen_v1.03¥applications¥lpc8xx¥examples¥periph¥periph_i2c¥keil_output¥iflash_nxp`

\_xpresso\_812 ディレクトリ下に保存する。

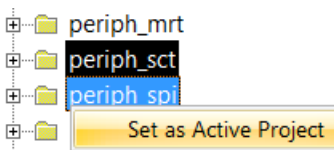
#### 4.1.7. ¥periph\_mrt



マルチレートタイマーの例である。マルチレートタイマー（MRT）は4つのチャンネルの繰り返し割り込みタイマーを提供し、各チャンネルは独立の時間間隔を設定出来る。

- プログラムはChip\_MRT\_GetIdleChannel で adle チャンネルを取得し、ランダム時間間隔を得られる。4つのチャンネルはビジーな場合、青いLED点灯。
- システムタイマーは緑LEDを5回点滅/秒と設定する。プログラム実行後、青いLED点灯、緑LEDは固定周波数で点滅し、赤LEDはランダムで点滅すると確認できる。
- HEX ファイルは  
`¥Code¥LPC0pen_v1.03¥applications¥lpc8xx¥examples¥periph¥periph_mrt¥keil_output¥iflash_nxp_xpresso_812` ディレクトリ下に保存する。

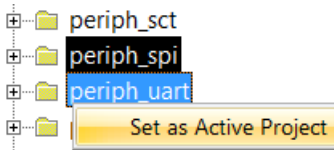
#### 4.1.8. ¥periph\_spi



SPI クエリと割り込みの2つの動作モードがある。

- プログラムはループバック通信モードを有効し、1つの開発ボードで完成できる。ユーザーは spi.c の87~92行のマクロ定義を編集により、プログラムの動作モードを変更する。
- プログラムは8バイトのデータを書き込み、ループバック通信モードでバッファのデータと送信データを比較し、正確なら、青いLED点灯。
- HEX ファイルは  
`¥Code¥LPC0pen_v1.03¥applications¥lpc8xx¥examples¥periph¥periph_spi¥keil_output¥iflash_nxp_xpresso_812` ディレクトリ下に保存する。

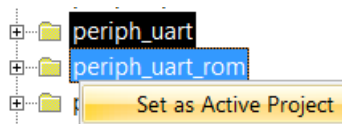
### 4.1.9. ¥periph\_uart



UART0 と UART1 間のデータ伝送プロセスであり、UART0 はクエリモードで、UART1 は割り込みモードを使用する。

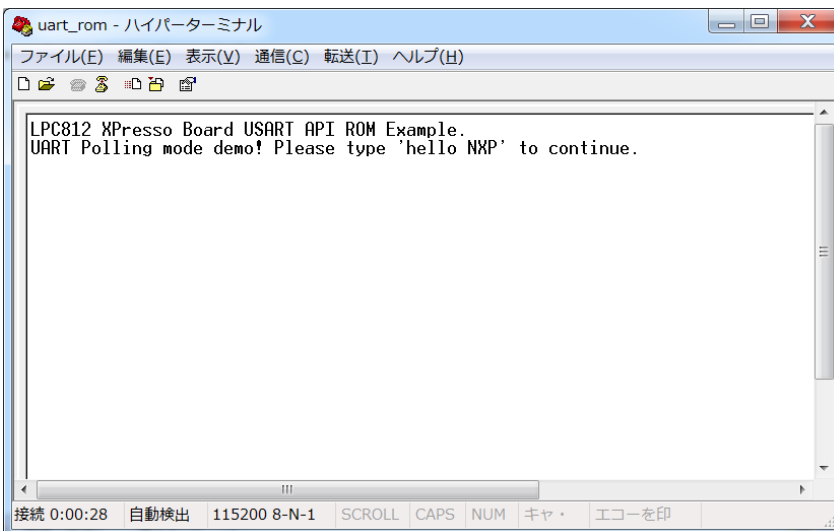
- テストする前に TXD0 (P\_0.4) と RXD1 (P\_0.14) を接続し、RXD0 (P\_0.0) と TXD1 (P\_0.13) を接続。
- プログラムは UART0 から UART1 へデータ送信する、次に UART1 は受信データを UART0 に返信する。受信データを比較、同じなら、青い LED 点灯。
- HEX ファイルは  
`¥Code¥LPCOpen_v1.03¥applications¥lpc8xx¥examples¥periph¥periph_uart¥keil_output¥iflash_nxp_xpresso_812` ディレクトリ下に保存する。

### 4.1.10. ¥periph\_uart\_rom

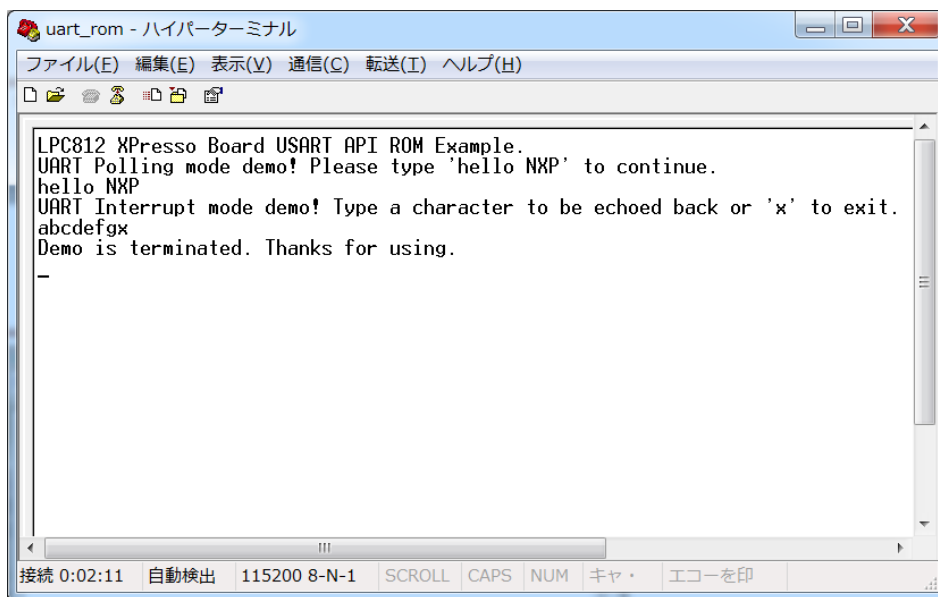


内部 ROM スペースにある UART API 関数の使い方を実演する。

- 開発ボードにプログラムをダウンロード後、ボーレートを 115200、ハードウェアフロー制御なしと設定、RST キーでプロセッサリセット、ハイパーターミナル上の表示情報は下記の通り：



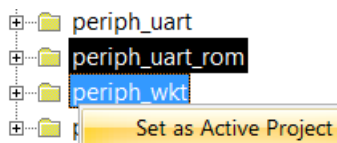
- シリアルポートはクエリモードで入力文字を待ち、ハイパーターミナルで hello NXP を入力し、正確な文字を入力すると、シリアルポートは割り込みモードに入り、送信した文字を受信する。x を入力すると、テストを終了する。



HEXファイルは

¥Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥examples¥periph¥periph\_uart\_rom¥keil\_output¥iflash\_nxp\_xpresso\_812 ディレクトリ下に保存する。

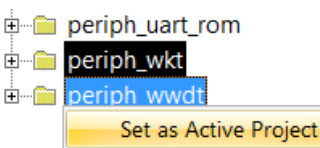
#### 4.1.11. ¥periph\_wkt



ウェイクアップ・タイマにより、プロセッサを低消費電力モードからウィックアップさせる。

- プログラム実行後、開発ボードの青いLED点滅、PI00\_6ピンローにし、プロセッサは低消費電力モードに入る。PI00\_6をハイにして、数秒後、ウェイクアップタイマーは動作モードに入り、青いLED点滅、PI00\_6ピンがローにするたびに、プロセッサは低消費電力モードを切り替える、4つの低消費電力モードはSLEEP、DEEP\_SLEEP、POWER\_DOWN、DEEP\_POWER\_DOWN。
- HEXファイルは  
Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥examples¥periph¥periph\_wkt¥keil\_output¥iflash\_nxp\_xpresso\_812 ディレクトリ下に保存する。

#### 4.1.12. ¥periph\_wwdt



ウォッチドッグ割り込みトリガー機能を実演する。

- LPC812プロセッサはウォッチドッグ機能があり、ユーザーは警報の時間をフィールド時間内に設定し、

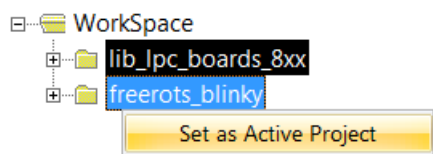
プログラムは警報時間内で、フィードしない場合、ウォッチドッグ警報がトリガーし、割り込みが発生する。

- プログラム実行後、割り込み発生する場合、青いLEDは値を反転動作する。
- HEX ファイルは

¥Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥examples¥periph¥periph\_wwdt¥keil\_output¥iflash\_nxp\_xpresso\_812 ディレクトリ下に保存する。

## 4.2. lpcopen\_freertos\_8xx.uvmpw プロジェクト管理ファイルのプログラム説明

### 4.2.1. ¥freertos\_blinky



FreeRTOS OS でLED点滅を実現する。

- 3つのタスクを作成する：1つは青いLED点滅、1つは緑LED点滅、1つは赤LED点滅を制御する。
- HEX ファイルは

¥Code¥LPCOpen\_v1.03¥applications¥lpc8xx¥examples¥freertos¥freertos\_blinky¥keil\_output¥iflash\_nxp\_xpresso\_812 ディレクトリ下に保存する。

以上。