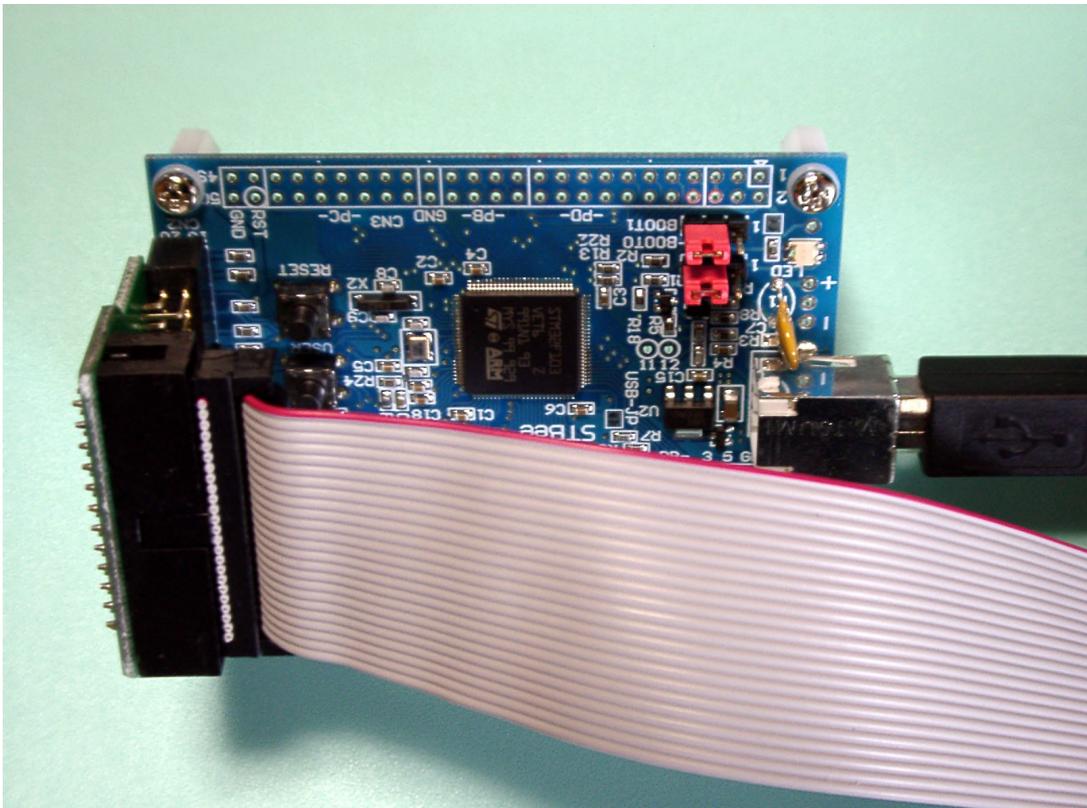


Raisonance Ride7+STX-RLINK 導入マニュアル



第 1.3 版
2010/9/11

Copyright (C) 2010 Shigeru Mitsugi

目次

1. 始めに	1
2. Ride7 のダウンロード	2
3. STM32 Primer2 用 Ride7 のアンインストール	2
4. Ride7 のインストール	3
5. RKit-ARM のインストール	5
6. STX-RLINK 用 USB ドライバのインストール	7
7. Ride7 の起動、コンパイルとデバッグ	9
8. トラブルシューティング	16
8.1 STX-RLINK の接続を確認する	16
9. ライブラリの利用	18
9.1 PLL の設定	18
9.2 新しいプロジェクトを作る	19
9.3 USART	22
9.4 タイマー	24
10. Hex ファイルの書き込み	25

1. はじめに

ストロベリー・リナックスから出ている STM32 マイコンボード STBee (72MHz, 512K+64KB)の開発環境を整えます。

STBee <http://strawberry-linux.com/catalog/items?code=32103>



ステップ実行やブレークポイントが利用できる JTAG デバッガ Raisonance RLink Standard を利用します。RLink Standard はデバッグ時、コードサイズが 32KB に制限されます。RLink Professional にはこの制限がありません。

RLink Standard

http://www.mcu-raisonance.com/~rlink-standard_microcontrollers_product~product_T017:4c05omvnccj4.html



上記の相当品と思われる STX-RLINK を Digi-Key から購入できます (¥6,776)。

STX-RLINK <http://search.digikey.com/scripts/DkSearch/dksus.dll?Cat=2621880&k=RLink>



2. Ride7 のダウンロード

フリーの統合開発環境 (IDE) Ride7 を下記からダウンロードします。

Ride7 のコンパイラは GCC です、生成できるコードサイズに制限はありません。

Ride7

http://www.mcu-raisonance.com/~ride7_microcontrollers_tool~tool_T018:4cw36y8a5c39.html

Ride7 Ride7_7.28.10.0075.exe バージョンが上がっていることがあります。

http://www.mcu-raisonance.com/mcu_downloads.html

RKit-ARM RKit-ARM_1.24.10.0050.exe バージョンが上がっていることがあります。

http://www.mcu-raisonance.com/mcu_downloads.html

Ride7 のマニュアルが下記にあります。

Ride7 for ARM http://elmicro.com/files/raisonance/gettingstartedarm_ride7.pdf

3. STM32 Primer2 用 Ride7 のアンインストール

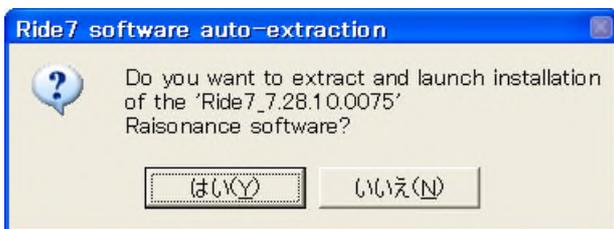
STM32 Primer2 の Ride7 をインストールしている場合は、アンインストールします。削除するのは RKit-ARM for Ride7 と Ride7 IDE です。

Ride7 IDE	サイズ	60.15MB	
RKit-ARM for Ride7	サイズ	353.00MB	
サポート情報を参照するには、 ここをクリックしてください。		使用頻度	低
このプログラムを変更したり、コンピュータから削除したりするには、[変更]または[削除]をクリックしてください。			
	<input type="button" value="変更"/>	<input type="button" value="削除"/>	

4. Ride7 のインストール

統合開発環境 Ride7 をインストールします。

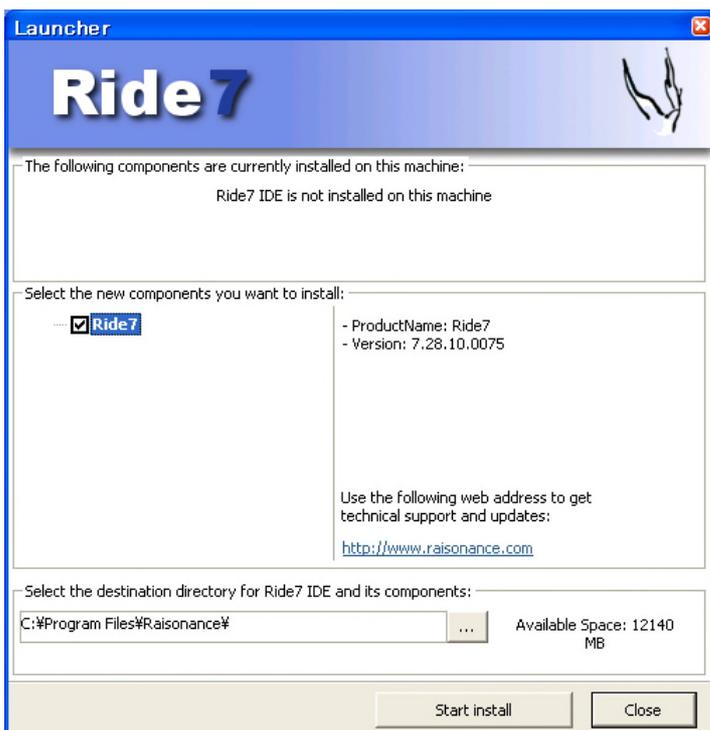
(1) Ride7_7.28.10.0075.exe をダブルクリックします。「はい」をクリックします。



(2) Install Ride7 or its components をクリックします。



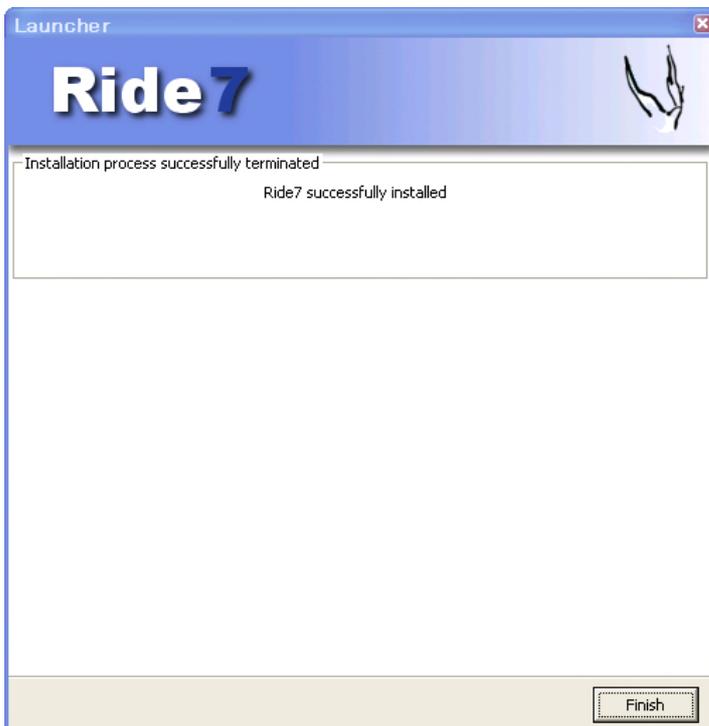
(3) Start install をクリックします。



(4) OK をクリックします。



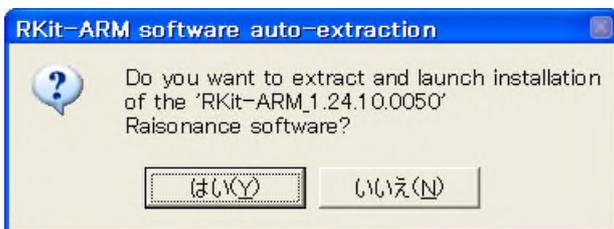
(5) Finish をクリックします。



5. RKit-ARM のインストール

arm-gcc やライブラリ、サンプルプログラムなどが入っている RKit-ARM をインストールします。

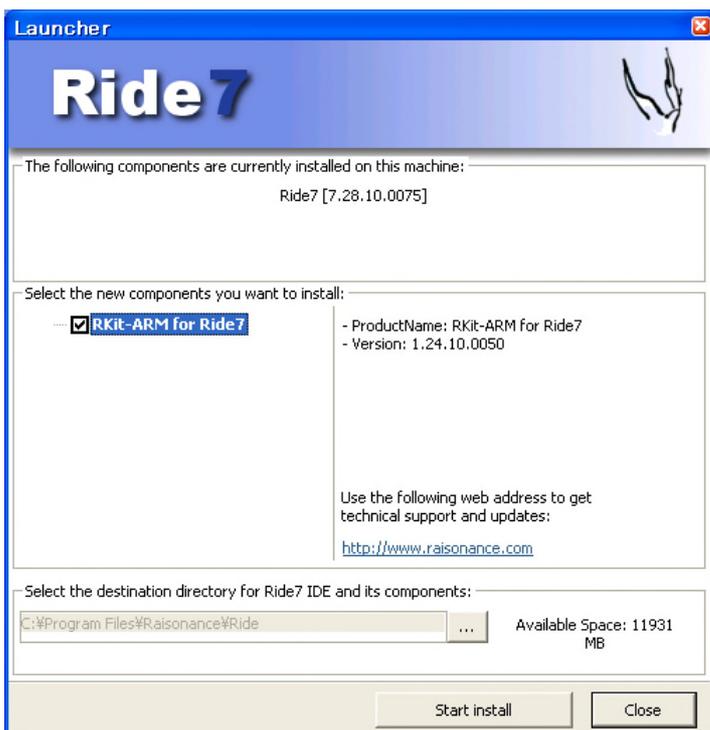
(1) RKit-ARM_1.24.10.0050.exe をダブルクリックします。「はい」をクリックします。



(2) Install Ride7 or its components をクリックします。



(3) Start install をクリックします。



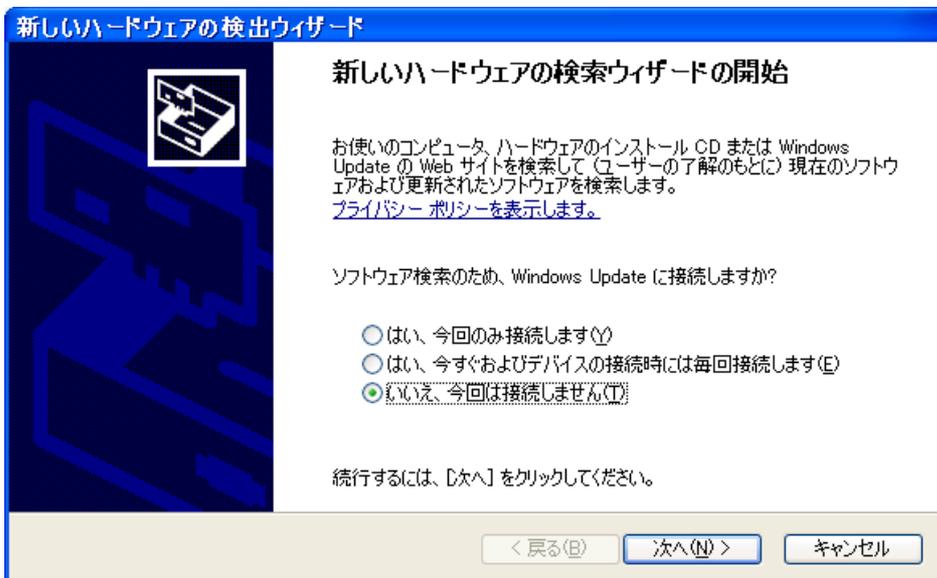
(4) Finish をクリックします。



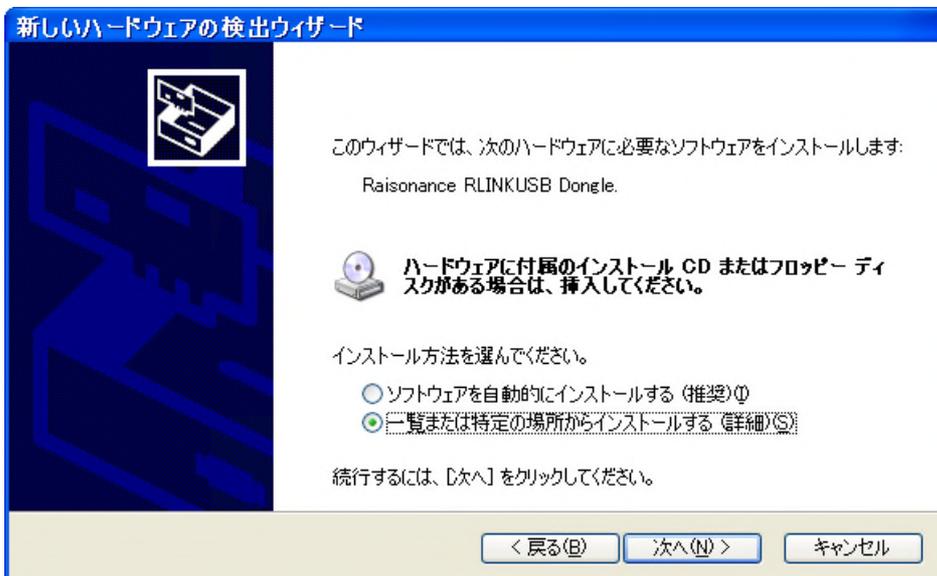
6. STX-RLINK 用 USB ドライバのインストール

(1) パソコンのUSBポートに STX-RLINK を接続します。

(2) ウィザード画面が表示されます。“いいえ、今回は接続しません”にチェックを入れ、次へをクリックします。

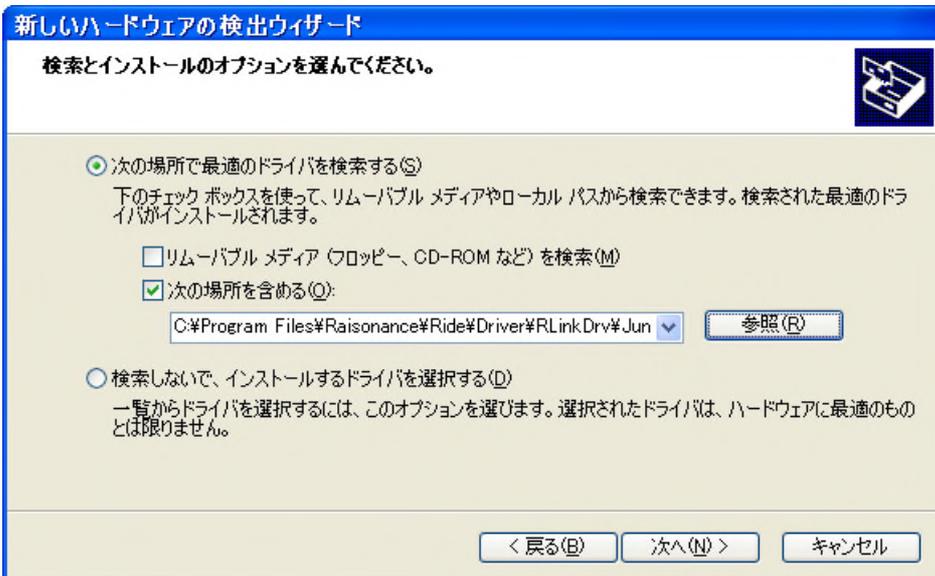


(3) “一覧または特定の場所からインストールする”にチェックを入れ、次へをクリックします。

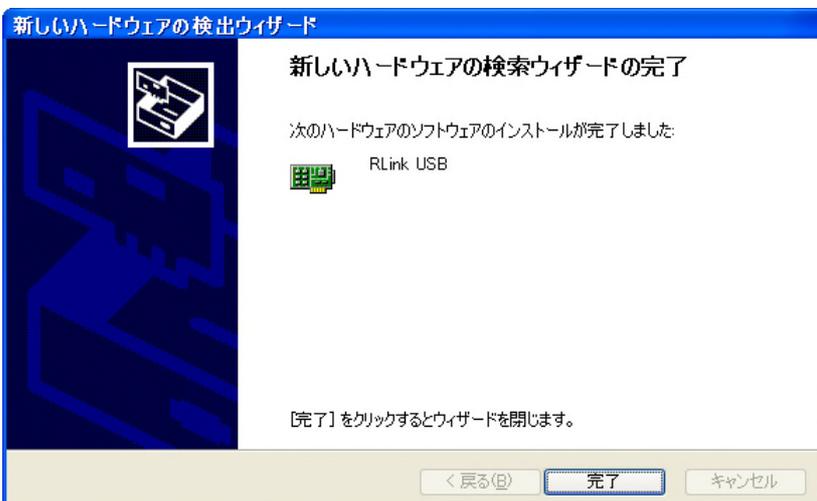


(4) 参照ボタンをクリックして、

C:\Program Files\Raisonance\Ride\Driver\RLinkDrv\Jungo_WinDriver_2000_NT_XP フォルダを選択します。次へをクリックします。



(5) 完了をクリックします。

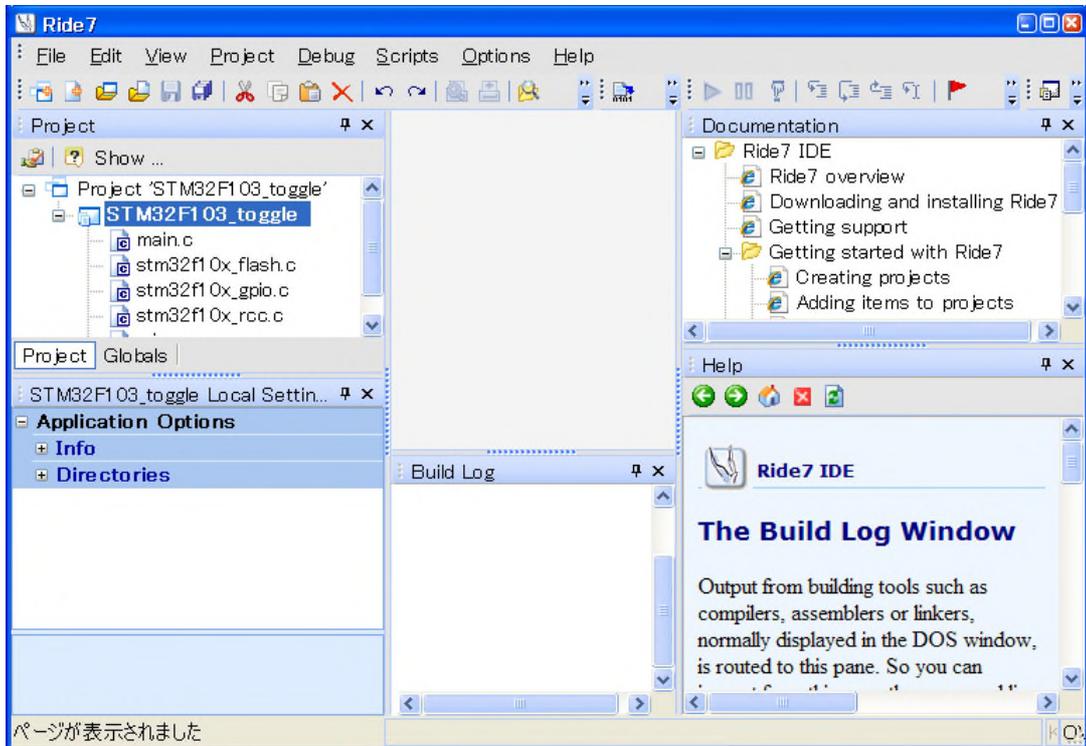


7. Ride7 の起動、コンパイルとデバッグ

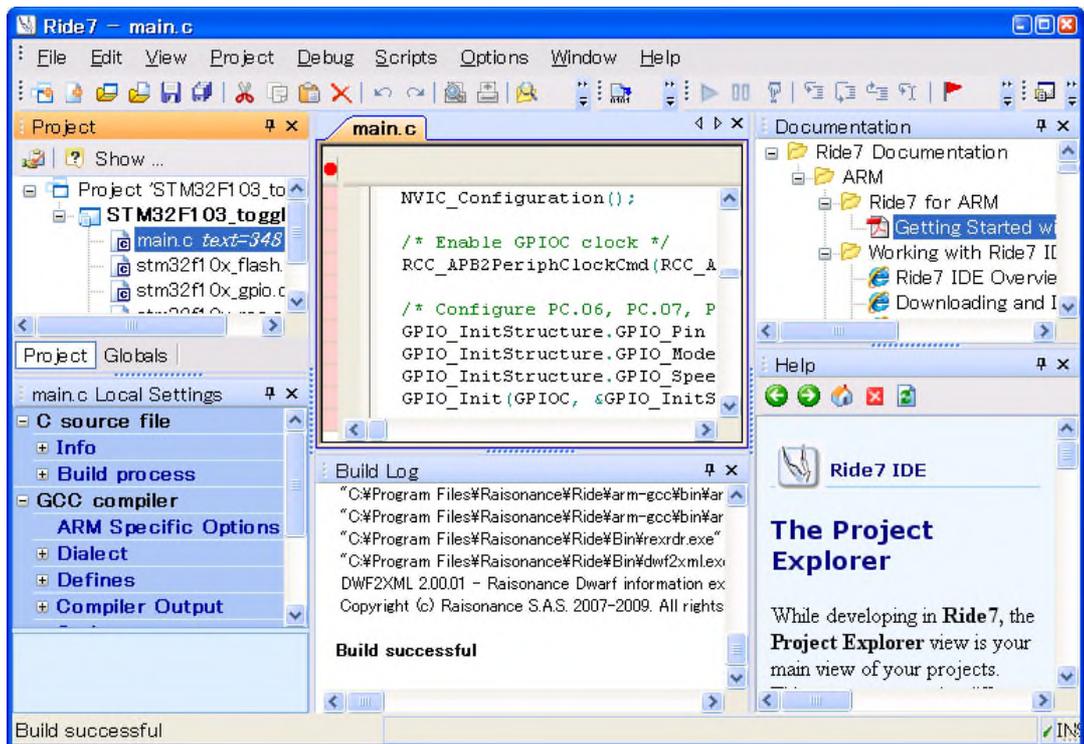
(1) Ride7 を起動します。メニュー Project > Open Project で

C:\Program Files\Raisonance\Ride\Examples\ARM\Eva\STM32F103_Toggle¥

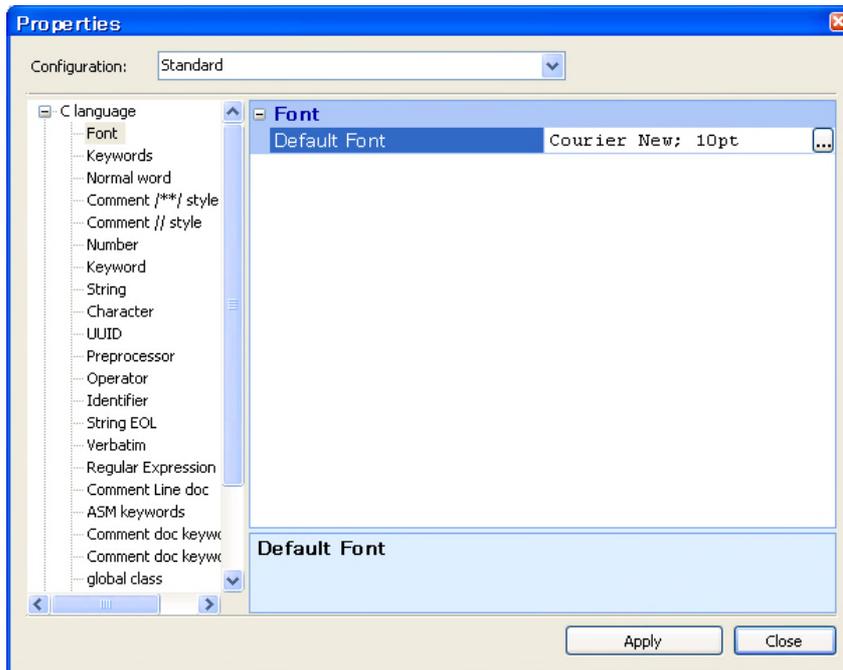
STM32F103_toggle.rprj を開きます (起動時に開いているかもしれません)。



(2) main.c をダブルクリックして開きます。メニュー Project > Build Project でコンパイルします。



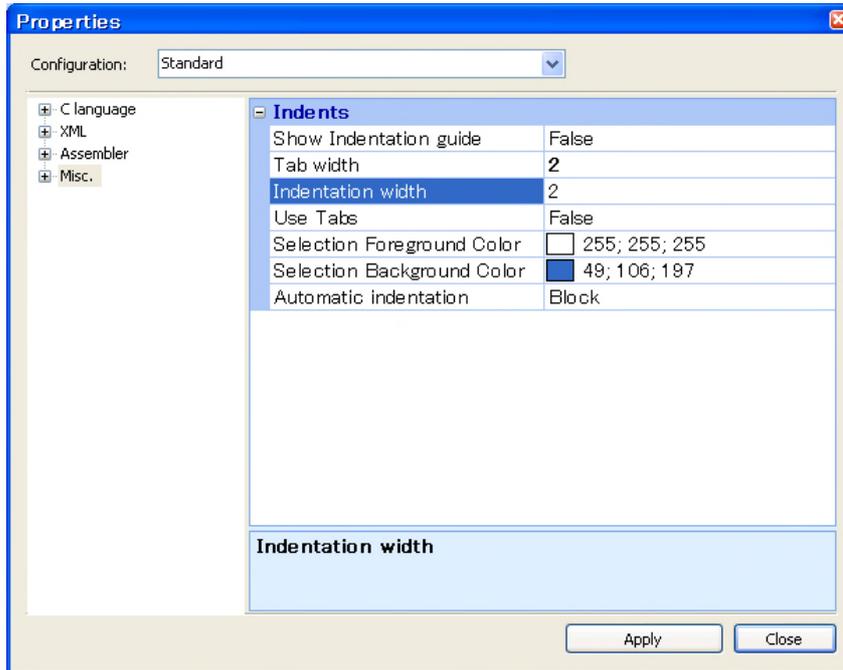
(3)メニュー Options > Editor preferences を選択します。C language の前の  をクリックして開き、Font をクリックします。Courier New; 10pt をクリックして、 をクリックします。



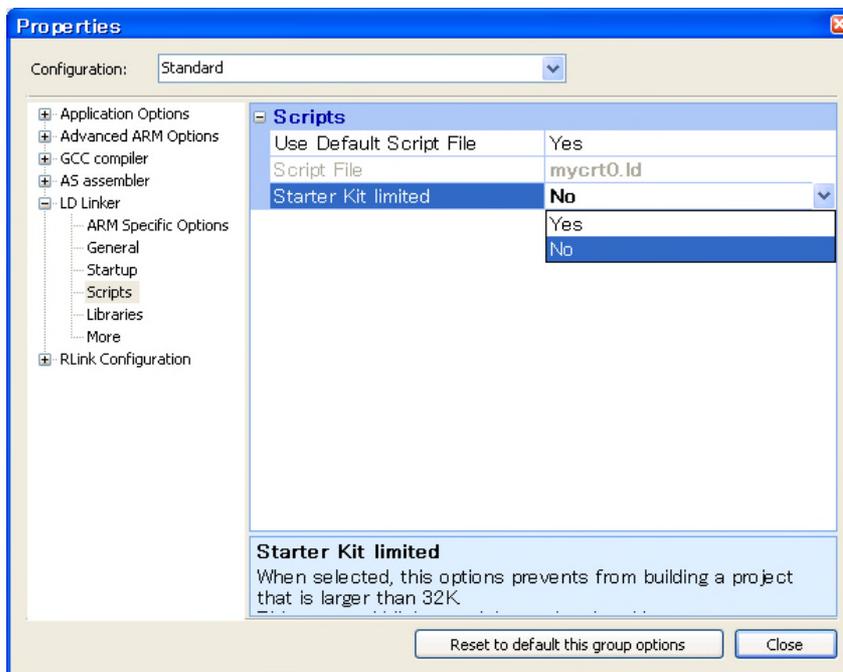
MS ゴシックを選びます。OK をクリックします。



C language の前の  をクリックして畳みます。Misc.をクリックして、Tab width と Indentation width を2に設定します。Apply をクリックします。



(4)コード生成のサイズ制限(32KB)を解除する場合は、メニュー Options > Project Properties を選択します。LD Linker の Scripts をクリックします。Starter Kit limited を No にします。Close をクリックします。



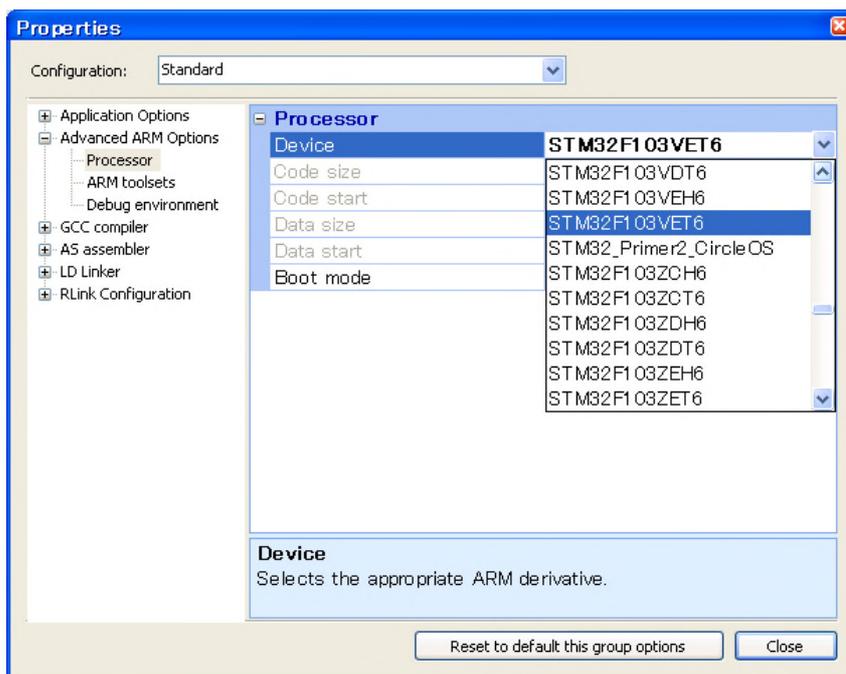
(5) マイコンボード STBee の LED はポート D の bit4 ですので、main.c の中のポート C を扱っている部分を下記のように書き換えます。

```
// GPIO D ポートを有効にします
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);

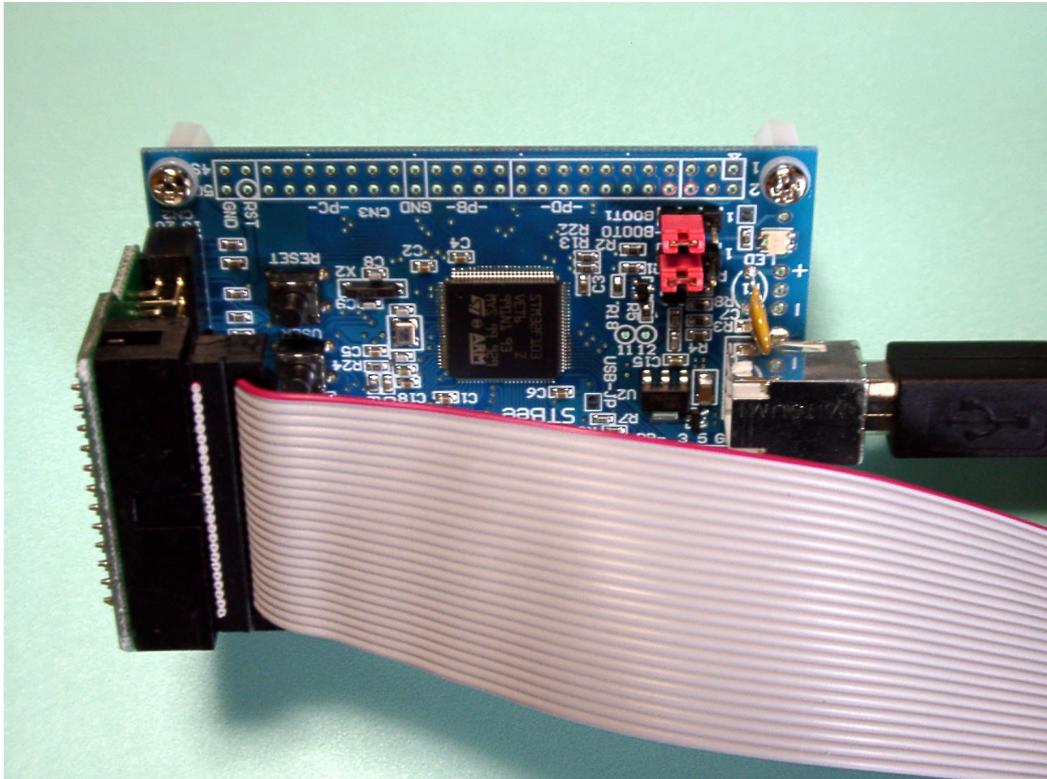
// PD.4 ポートを出力にします。PD.4=赤 LED Low で点灯
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure); // 初期化関数を読み出します。

while(1)
{
    GPIO_ResetBits(GPIOD, GPIO_Pin_4);
    Delay(200000);
    GPIO_SetBits(GPIOD, GPIO_Pin_4);
    Delay(200000);
}
```

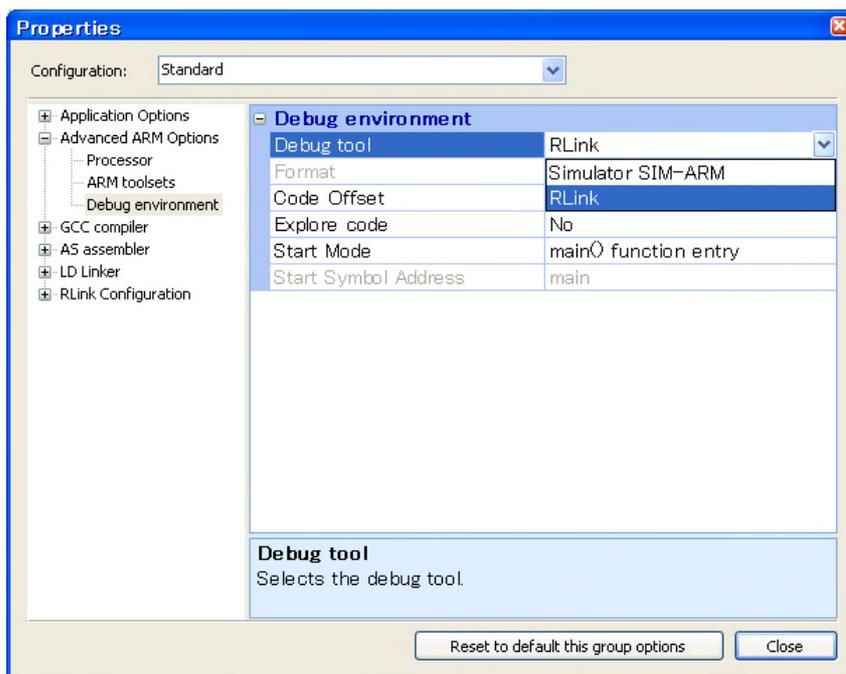
(5) マイコンボード STBee の CPU は STM32F103VET6 です。メニュー Options > Project Properties を選択します。Advanced ARM Options の Processor をクリックします。STM32F103RBT6 をクリックして、STM32F103VET6 を選びます。Close をクリックします。再度、コンパイルします。



(6) JTAG デバッガ STX-RLINK とマイコンボード STBee をフラットケーブルおよび 24pin → 20pin 変換基板で接続します。STBee を USB で PC と接続します。ブートローダー(DFU)用 USB ドライバのインストール画面が出ますが、インストールは不要です(9 項で DFU を消してしまうため)。

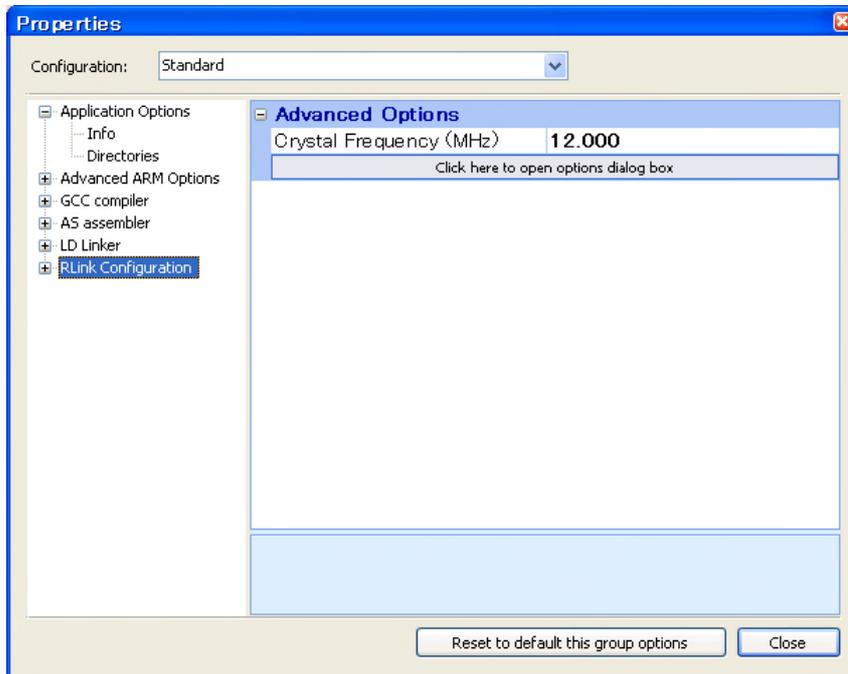


(7) メニュー Options > Project Properties を選択します。Advanced ARM Options の Debug environment をクリックします。SimulatorSIM-ARM をクリックして、RLink を選びます。



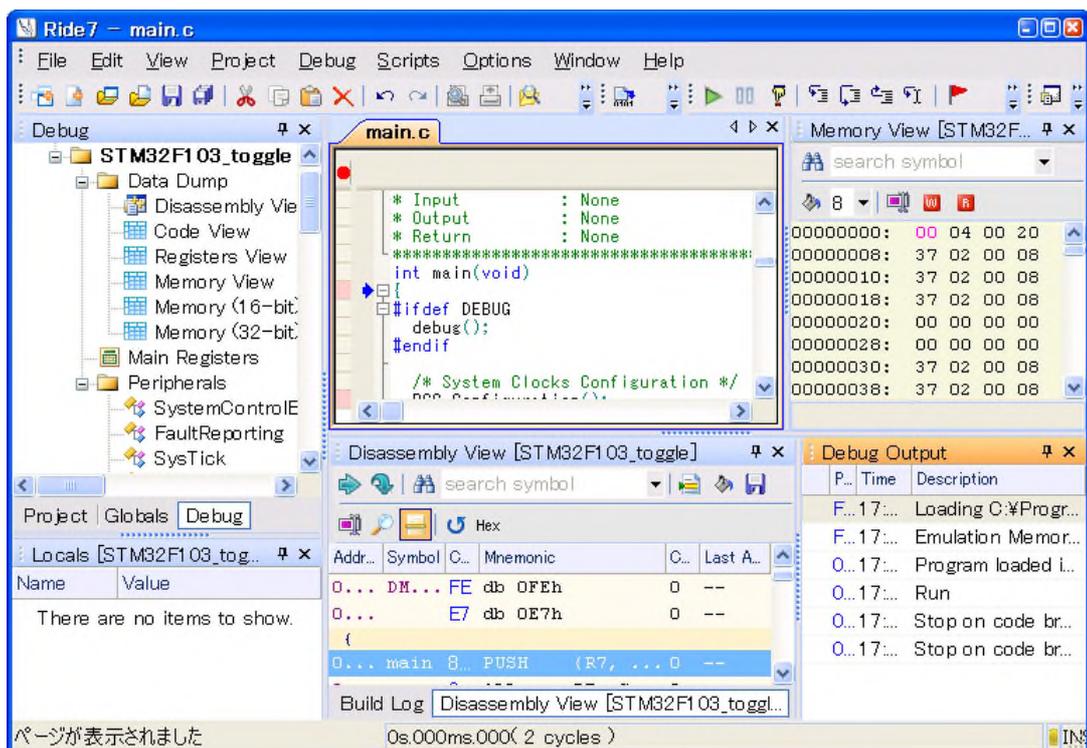
(8) RLink Configuration の Crystal Frequency を 12.000 にします。12MHz は STBee の水晶発振子の周波数です。Close をクリックします。

<参考> 8.000 のままでも支障はないようです。



<注意> 次の(9)項を行うと、STBee に予め書き込んであるブートローダー(DFU)が消えます。

(9)メニュー Debug > Start を選択します。プログラムをフラッシュに書き込み、プログラムは main()の最初で止まります。ステップ実行(F8 キー)ができます。



(10)メニュー Debug > Run でプログラムを実行します。LED が点滅します。

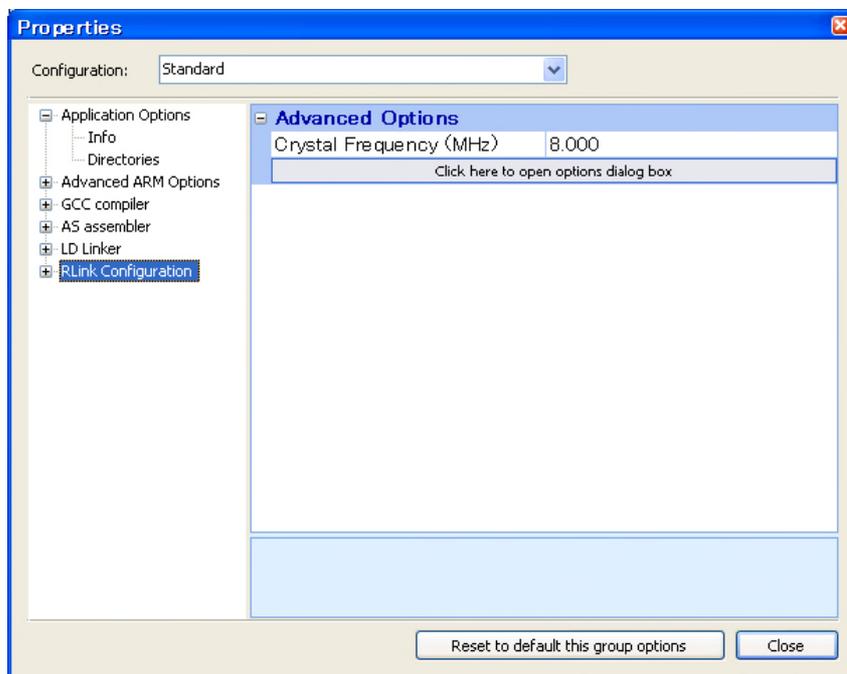
(11)メニュー Debug > Stop でプログラムを止めます。メニュー Debug > Terminate でデバッグを終了します。

(12)メニュー File > Exit で Ride7 を終了します。

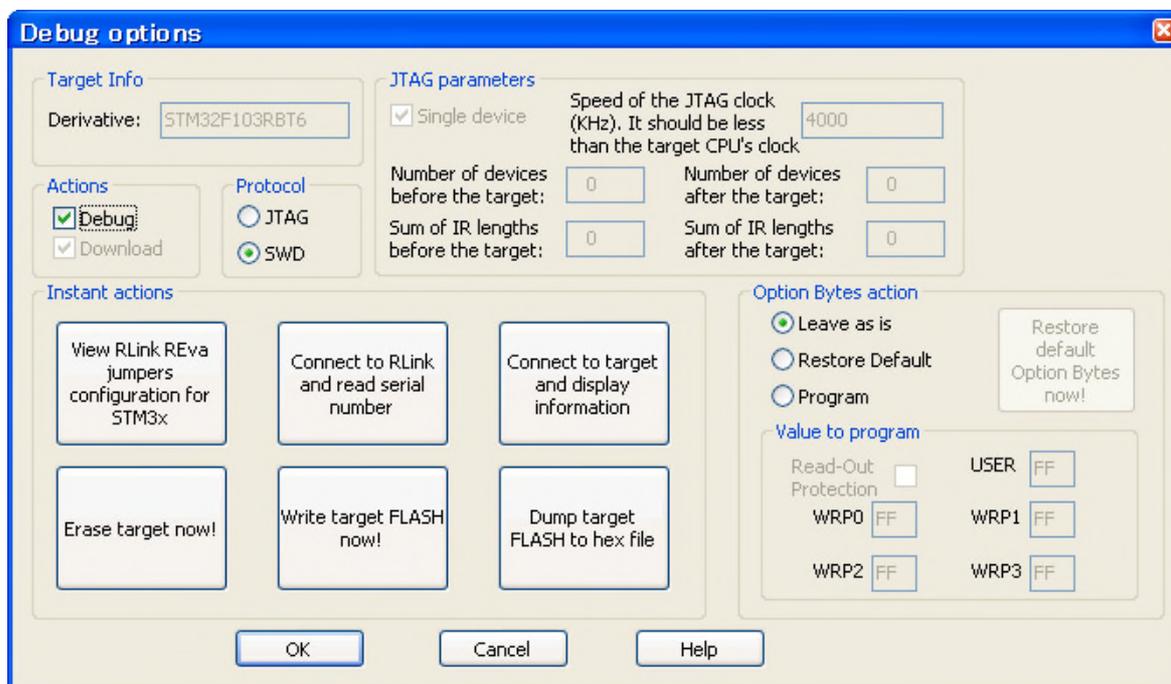
8. トラブルシューティング

8.1 STX-RLINK の接続を確認する

(1)メニュー Options > Project Properties を選択します。RLink Configuration をクリックします。
Click here to open options dialog box をクリックします。



(2) Connect to RLink and read serial number をクリックします。



(3) 下記のメッセージが出ます。



9. ライブラリの利用

9.1 PLL の設定

7 項で実行したプログラムは、PLL を利用していないため、12MHz で動作しています。

RCC_Configuration()を下記のように変更すると、72MHz で動作します。

C:¥Program Files¥Raisonance¥Ride¥lib¥ARM¥STM32F10x_lib¥examples¥USART¥Polling¥main.c を参考にしました。

```
void RCC_Configuration(void)
{
    /* RCC system reset(for debug purpose) */
    RCC_DeInit();

    /* Enable HSE */
    RCC_HSEConfig(RCC_HSE_ON);

    /* Wait till HSE is ready */
    while(RCC_GetFlagStatus(RCC_FLAG_HSERDY) == RESET)
    {}

    /* Enable Prefetch Buffer */
    FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);

    /* Flash 2 wait state */
    FLASH_SetLatency(FLASH_Latency_2);

    /* HCLK = SYSCLK */
    RCC_HCLKConfig(RCC_SYSCLK_Div1);

    /* PCLK2 = HCLK */
    RCC_PCLK2Config(RCC_HCLK_Div1);

    /* PCLK1 = HCLK/2 */
    RCC_PCLK1Config(RCC_HCLK_Div2);

    /* PLLCLK = 12MHz * 6 = 72 MHz */
    RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_6);

    /* Enable PLL */
    RCC_PLLCmd(ENABLE);
```

```

/* Wait till PLL is ready */
while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET)
{
}

/* Select PLL as system clock source */
RCC_SYSClkConfig(RCC_SYSClkSource_PLLCLK);

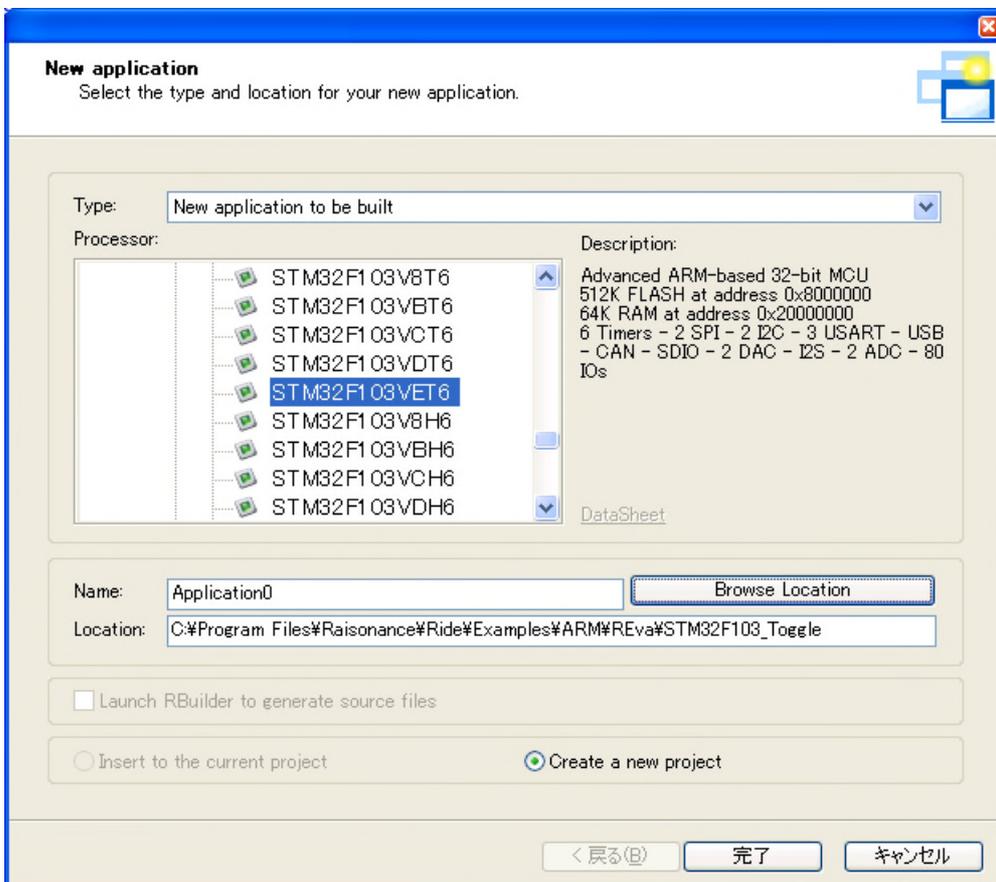
/* Wait till PLL is used as system clock source */
while(RCC_GetSYSClkSource() != 0x08)
{
}
}

```

<参考> この改造を行ったファイルが、ブログ「すいか村の電子工房」にアップロードしたSTM32F103_Toggle.zip です。

9.2 新しいプロジェクトを作る

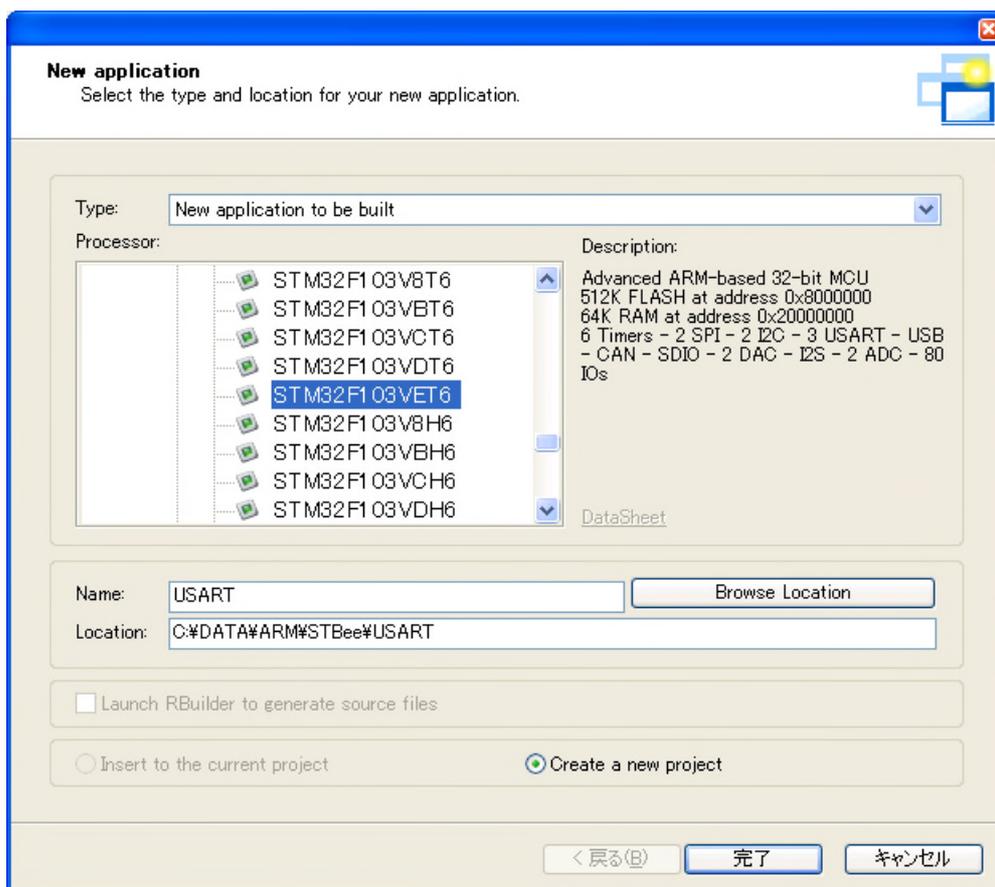
(1)メニュー Project > New Project を選びます。Browse Location をクリックします。



(2)新しいフォルダの作成をクリックして、適当なフォルダ(ここでは C:\¥DATA¥ARM¥STBee)の下に、USART フォルダを作ります。OK をクリックします。



(3)Processor を STM32F103VET6、Name 欄に USART と入れ、Create a new project をチェックして、完了をクリックします。

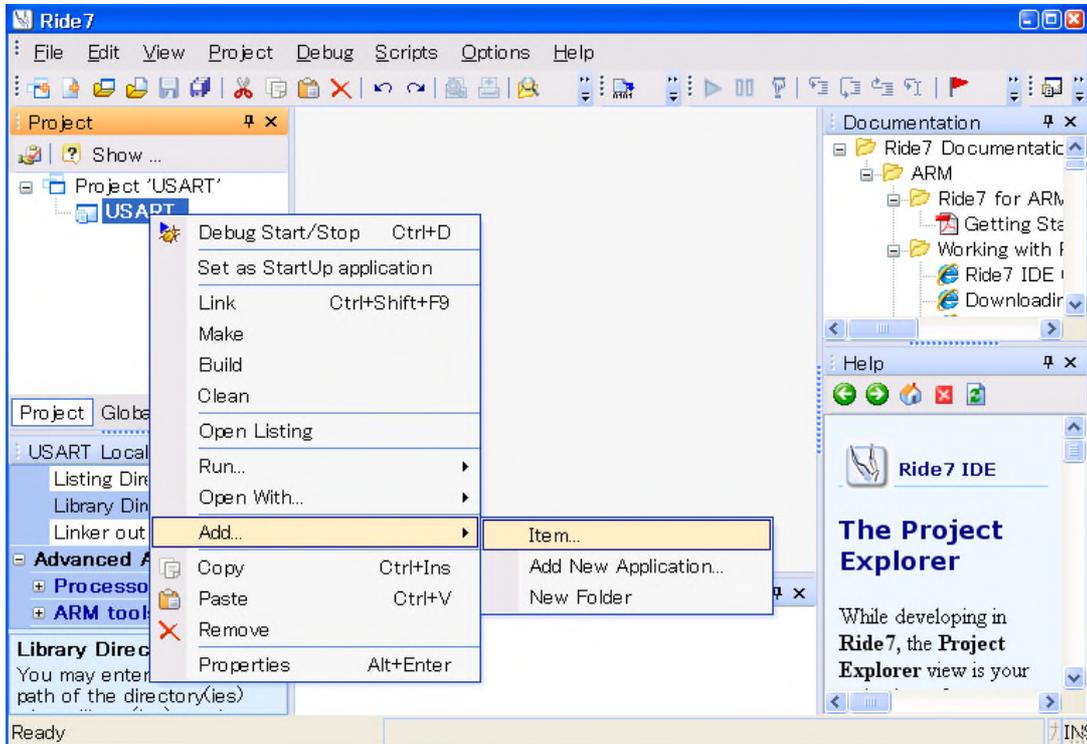


(4)7 項のプロジェクト STM32F103_toggle から、下記の.c ファイルと、全ての.h ファイルを USART フォルダにコピーします。

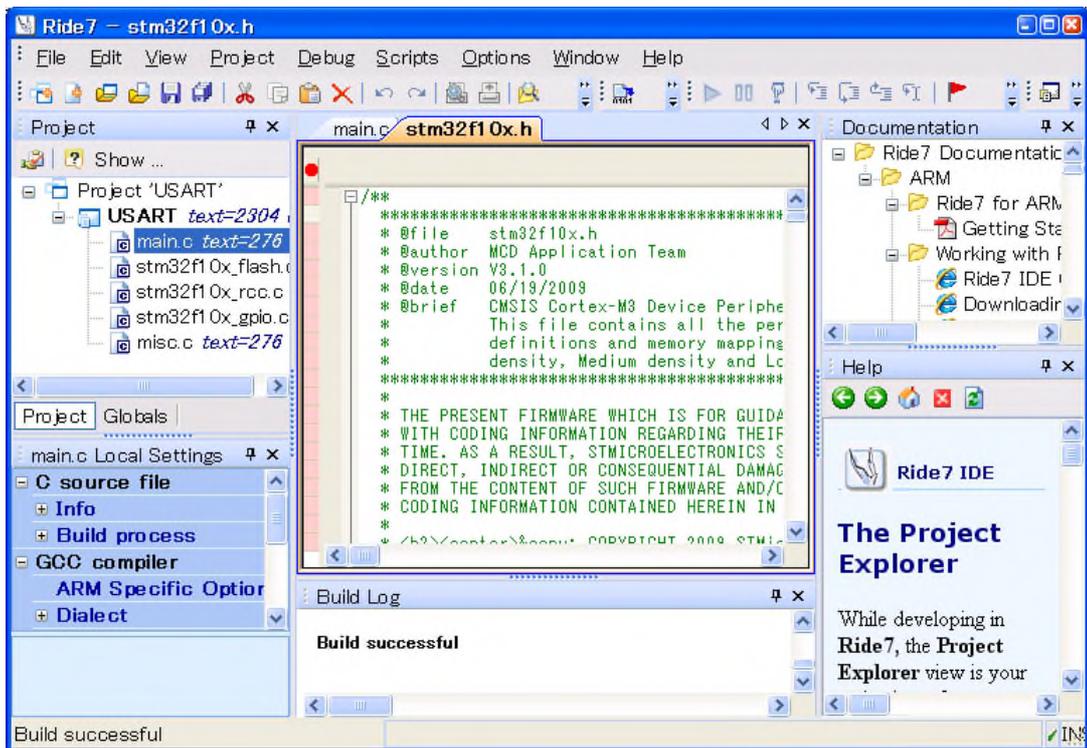
main.c stm32f10x_flash.c stm32f10x_gpio.c stm32f10x_rcc.c misc.c

core_cm3.h misc.h stm32f10x.h stm32f10x_conf.h stm32f10x_flash.h
 stm32f10x_gpio.h stm32f10x_rcc.h system_stm32f10x.h

(5) USART で右クリックして、Add > Item を選びます。上記の.c ファイルを選び OK をクリックします。



(6)メニュー Project > Build Project でコンパイルします。



9.3 USART

(1) C:\Program Files\Raisonance\Ride\lib\ARM\STM32F10x_Lib\examples\USART\Polling\main.c を参考に 9.2 項の main.c を改造します。

<参考> この改造を行ったプロジェクトファイルが、ブログ「すいか村の電子工房」にアップロードした USART.zip です。

(2) C:\Program Files\Raisonance\Ride\lib\ARM\STM32Lib-v312\Libraries\STM32F10x_StdPeriph_Driver\inc と、同\src から下記ファイルを USART フォルダにコピーします。9.2 項(5)と同じようにして、stm32f10x_usart.c をプロジェクトに加えます。

```
stm32f10x_usart.h  stm32f10x_usart.c
```

(3) USART フォルダの stm32f10x.h を開き、下記の赤の部分を変更します。12MHz は STBee の水晶発振子の周波数です。ここが 8000000 のままでは、232C の通信速度が 115.2Kbps の 1.5 倍となり、正常に通信できません。

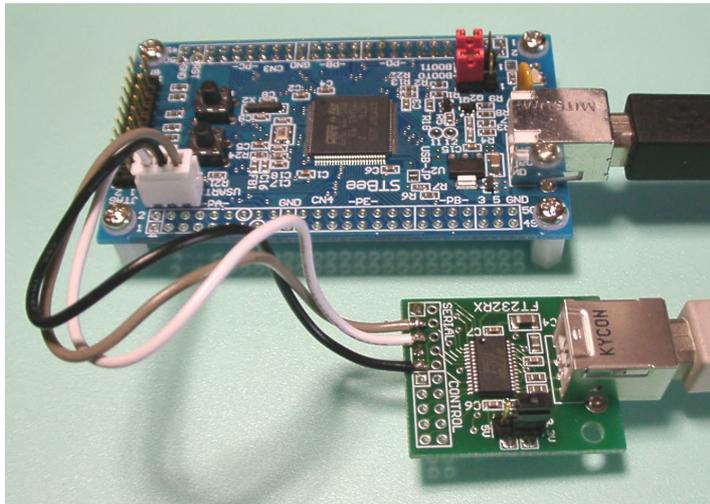
```
#ifndef STM32F10X_CL
#define HSE_Value  ((uint32_t)25000000) /*< Value of the External oscillator in Hz */
#else
#define HSE_Value  ((uint32_t)12000000) /*< Value of the External oscillator in Hz */
#endif /* STM32F10X_CL */
```

(4) USART フォルダの stm32f10x_conf.h の下記のコメント化を外します。

```
/* #include "stm32f10x_usart.h" */
=> #include "stm32f10x_usart.h"
```

(5)メニュー Project > Build Project でコンパイルします。メニュー Debug > Start を選択します。プログラムをフラッシュに書込み、プログラムは main()の最初で止まります。メニュー Debug > Run でプログラムを実行します。

(6) STBee の USART1 を PC と接続します。下記では、ストロベリー・リナックスの FT232RL USB シリアル変換モジュールキット「FT232RX」で接続しています。



<参考>

FT232RL USB シリアル変換モジュールキット メーカー品番:FT232RX
<http://strawberry-linux.com/catalog/items?code=50025>

USB-TTL シリアルコンバータ(3.3V) メーカー品番:TTL-232R-3V3
<http://strawberry-linux.com/catalog/items?code=50030>

(7) PC のハイパーターミナルを起動し、文字を打ち込んで Enter キーを押すと、打ち込んだ文字を返してきます。その際に LED が一瞬光ります。

9. 4 タイマー

(1) C:\Program Files\Raisonance\Ride\lib\ARM\STM32Lib-v312\Project

STM32F10x_StdPeriph_Examples\TIM\TimeBase\main.c を参考に 9.3 項の main.c を改造します。

Prescaler を 4 から 3600 に変更して、TIM2 counter clock = 72MHz/3600 = 20KHz にしています。0.5 秒間隔でタイマー割り込みが発生するように、CCR1_Val = 10000 としました (20KHz/10000=2Hz)。

<参考> この改造を行ったプロジェクトファイルが、ブログ「すいか村の電子工房」にアップロードした USART_Timer.zip です。

(2) TimeBase フォルダにある下記ファイルを USART フォルダにコピーします。

9.2 項(5)と同じようにして、stm32f10x_it.c をプロジェクトに加えます。

```
stm32f10x_it.h    stm32f10x_it.c
```

stm32f10x_it.c 中の void TIM2_IRQHandler(void) を改造します。使用しない TIM_IT_CC2~4 の処理部分を削除します。LED が点滅するように赤の部分を変更します。

```
GPIO_WriteBit(GPIO_D, GPIO_Pin_4, (BitAction)(1 - GPIO_ReadOutputDataBit(GPIO_D,
GPIO_Pin_4)));
```

(3) C:\Program Files\Raisonance\Ride\lib\ARM\STM32Lib-v312\Libraries\

STM32F10x_StdPeriph_Driver\inc と、同\src から下記ファイルを USART フォルダにコピーします。

9.2 項(5)と同じようにして、stm32f10x_tim.c をプロジェクトに加えます。

```
stm32f10x_tim.h    stm32f10x_tim.c
```

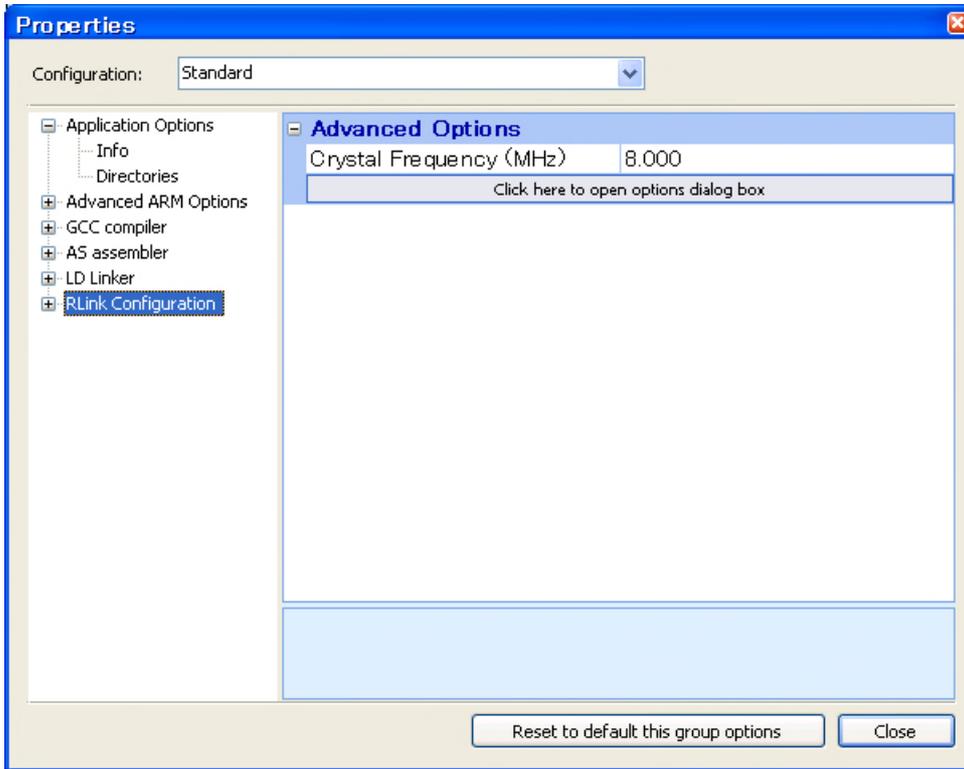
(4) USART フォルダの stm32f10x_conf.h の下記のコメント化を外します。

```
/* #include "stm32f10x_tim.h" */
=> #include "stm32f10x_tim.h"
```

(5)メニュー Project > Build Project でコンパイルします。メニュー Debug > Start を選択します。プログラムをフラッシュに書込み、プログラムは main()の最初で止まります。メニュー Debug > Run でプログラムを実行します。LED が点滅します。

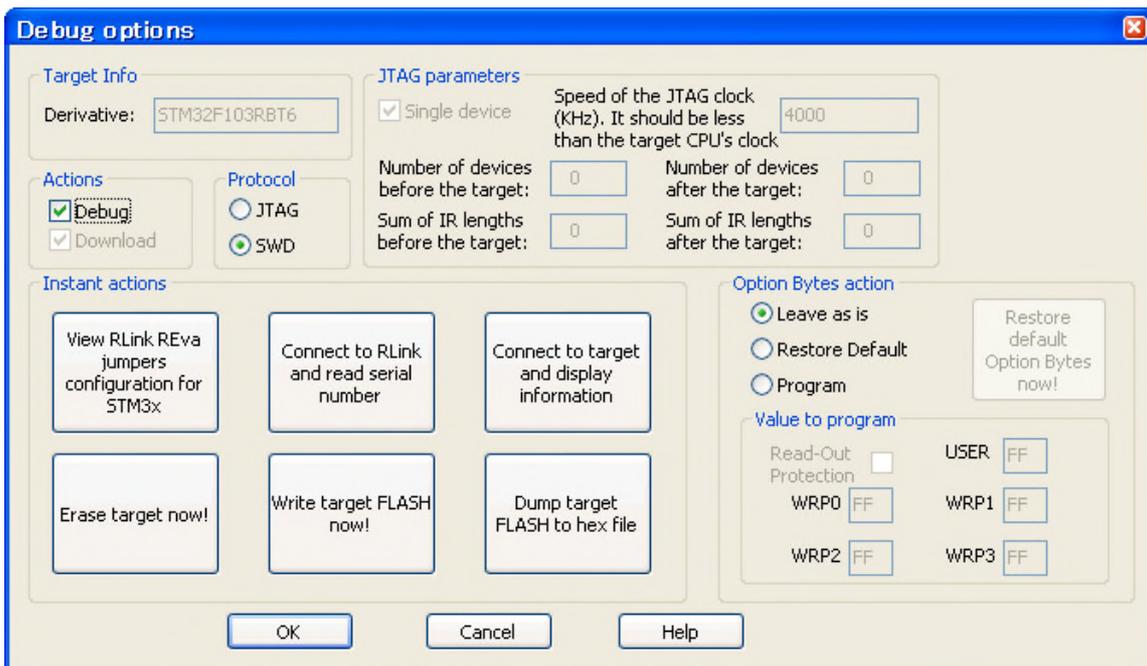
10. Hex ファイルの書き込み

(1)メニュー Options > Project Properties を選択します。RLink Configuration をクリックします。
Click here to open options dialog box をクリックします。



<注意> 次の(2)項を行うと、STBee に予め書き込んであるブートローダー(DFU)が消えます。

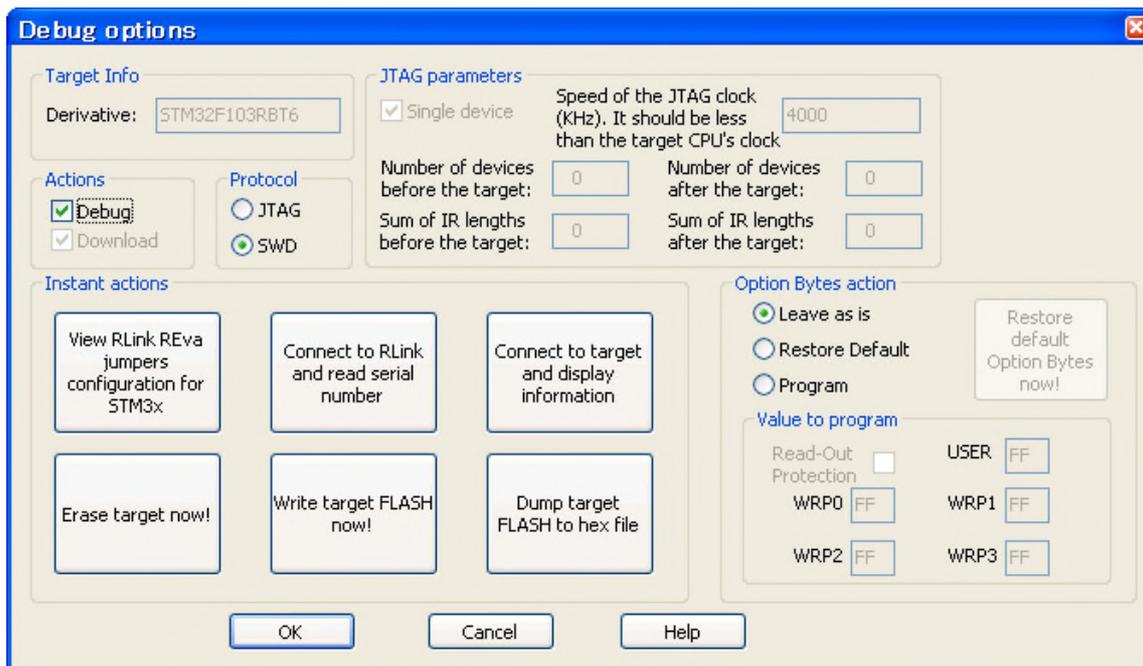
(2) Erase target now!をクリックします。



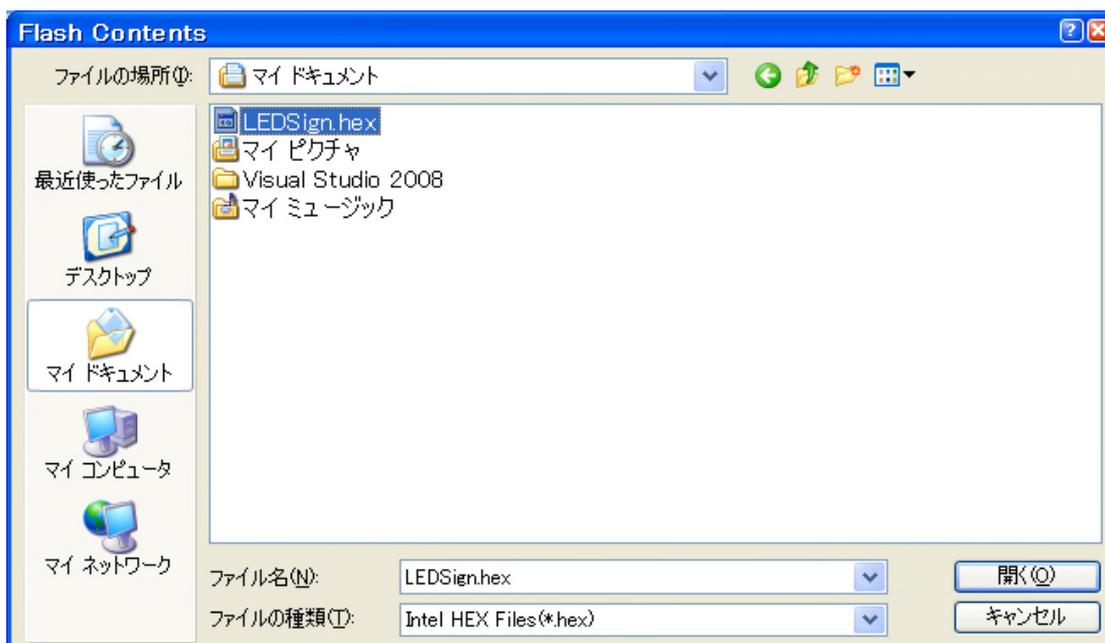
OK をクリックします。



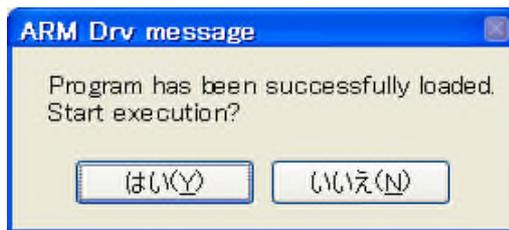
(3) Write target FLASH now!をクリックします。



LEDSign.hex を選択します。



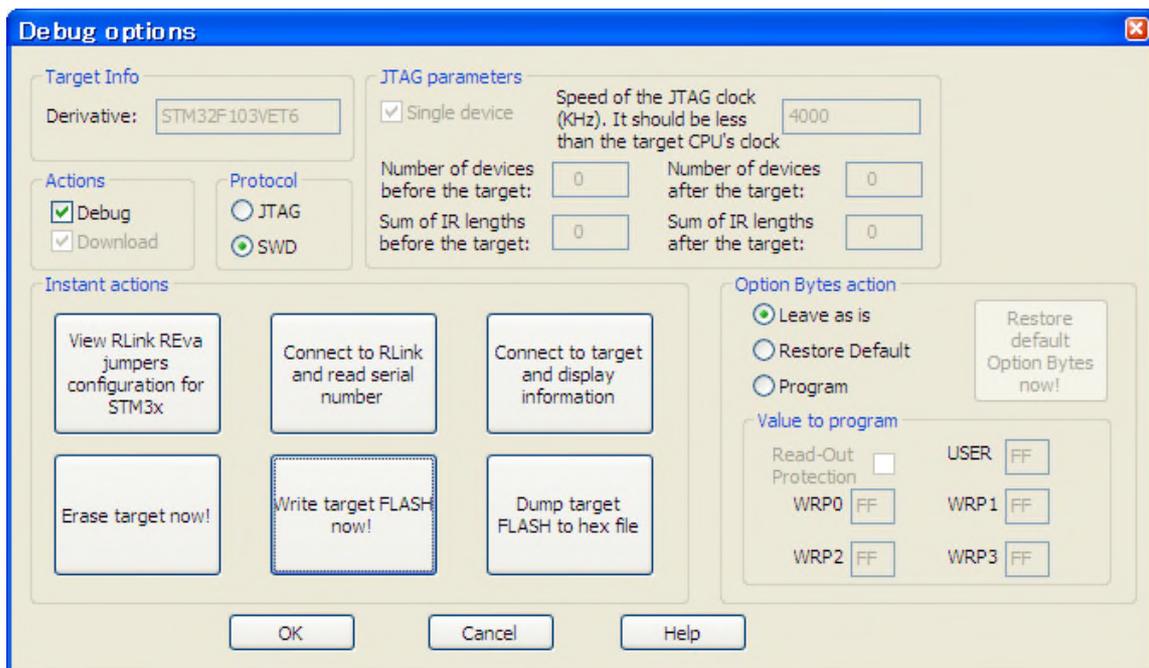
「はい」をクリックします。



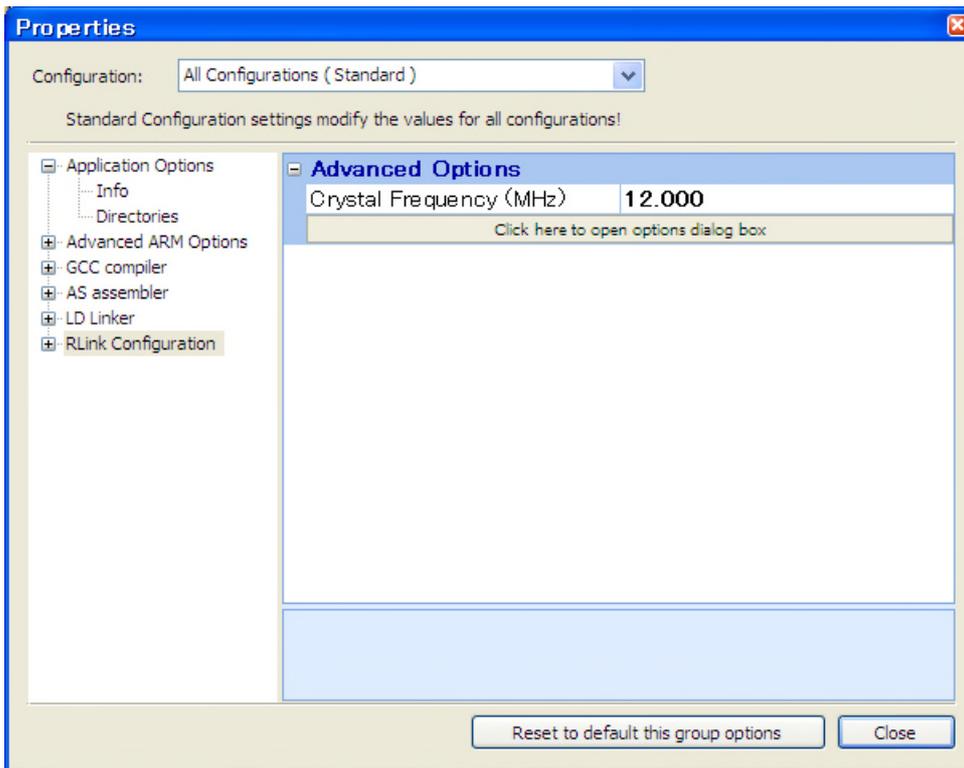
OK をクリックします。



(4) Cancel をクリックします。



(5) Close をクリックします。



すいか村の電子工房

<http://suikamura.blog91.fc2.com/>
