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_T017:4co5om vnccj4.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 ですので、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	<u>353.00MB</u>
<u>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。</u>	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。	変	更削除

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。

Ride7 s	oftware auto-extra	action 🛛 🖉
2	Do you want to ext of the 'Ride7_7.28.1 Raisonance softwa	ract and launch installation 0.0075′ re?
	(4U(X)	()()え(<u>N</u>)

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

😼 Launcher		8
Ride7		M
	Install Ride7 or its components	
1 Alexa		
	View Readme	Close

(3) Start install をクリックします。

Launcher	8	
Ride7	M	
The following components are currently instal	led on this machine:	
Ride7 IDE is not i	installed on this machine	
Select the new components you want to insta	bil:	
Ride7	- ProductName: Ride7	
	Use the following web address to get	
	technical support and updates:	
	http://www.raisonance.com	
Select the destination directory for Ride7 IDE and its components:		
C:¥Program Files¥Raisonance¥	Available Space: 12140 MB	
	Start install Close	

(4) OK をクリックします。



(5) Finish をクリックします。

Launcher	×
Ride7	Pà
Installation process successfully terminated	
Ride7 successfully installed	

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。

RKit-AF	M software auto-	extraction
?	Do you want to ext of the 'RKit-ARM_1 Raisonance softwa	tract and launch installation .24.10.0050' re?
	(#UV)	いいえ(<u>N</u>)

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

📓 Launcher	
Ride7	6
	Install Ride7 or its components
	<u>Uninstall Ride7 or its</u> <u>components</u>
	S
	View Readme Close

(3) Start install をクリックします。

Launcher	×
Ride7	M
The following components are currently insta	Iled on this machine:
Ride7 [7.28.10.0075]
- Select the new components you want to inst	all:
RKit-ARM for Ride7	- ProductName: RKit-ARM for Ride7 - Version: 1.24.10.0050
	Use the following web address to get technical support and updates:
	http://www.raisonance.com
Select the destination directory for Ride7 IDE and its components:	
C:¥Program Files¥Raisonance¥Ride	Available Space: 11931 MB
<u></u>	Start install Close

(4) Finish をクリックします。

Launcher	×
Ride7	P
- Installation process successfully terminated	
RKit-ARM for Ride7 successfully installed	
	Finish

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。

新しいハードウェアの検出ウィザード		
	新しいハードウェアの検索ウィザードの開始	
	お使いのコンピュータ、ハードウェアのインストール CD または Windows Update の Web サイトを検索して (ユーザーの了解のもとに) 現在のソフトウ ェアおよび更新されたソフトウェアを検索します。 <u>プライバシー ポリシーを表示します。</u>	
	ソフトウェア検索のため、Windows Update に接続しますか?	
	 ○はい、今回のみ接続します(Y) ○はい、今すぐおよびデバイスの接続時には毎回接続します(E) ●いいえ、今回は接続しません(T) 	
	続行するには、[次へ] をクリックしてください。	
	< 戻る(B) 次へ(N) > キャンセル	

(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。

新しいハードウェアの検出ウィザード		
	このウィザードでは、次のハードウェアに必要なソフトウェアをインストールします: Raisonance RLINKUSB Dongle.	
	●一覧または特定の場所からインストールする (詳細)(S)	
	続行するには、 D次へ] をクリックしてください。	
	< 戻る(B) 次へ(M) > キャンセル	

(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。

新しいハードウェアの検出ウィザード
検索とインストールのオブションを選んでください。
 ◇次の場所で最適のドライバを検索する(S) 下のチェック ボックスを使って、リムーバブル メディアやローカル パスから検索できます。検索された最適のドラ イバがインストールされます。 □リムーバブル メディア (フロッピー、CD-ROM など)を検索(M) ◇ 次の場所を含める(Q): ○:¥Program Files¥Raisonance¥Ride¥Driver¥RLinkDrv¥Jun ▼ 参照(R) ○ 検索しないで、インストールするドライバを選択する(D) ー覧からドライバを選択する(こは、このオプションを選びます。選択されたドライバは、ハードウェアに最適のもの とは限りません。
〈戻る(B) 次へ(N) > キャンセル
フォルダの参照 ハードウェアのドライバを含むフォルダを選んでください。
Config Config Coc Coc Coc Coc Coc Coc Coc Coc
サブ フォルダを表示するには、プラス (+) サインをクリックしてください。 OK キャンセル

(5)完了をクリックします。

新しいハードウェアの検出ウィ	ザード
	新しいハードウェアの検索ウィザードの完了 次のハードウェアのソフトウェアのインストールが完了しました: 理論 RLink USB
	[完了]をクリックするとウィザードを閉じます。 (〈戻る(B) 完了 キャンセル

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

 $C: \ensuremath{{\tt Files}{\tt FRaisonance}{\tt Ride}{\tt Examples}{\tt ARM}{\tt FREva} \\ \ensuremath{{\tt STM32F103}_{\tt Toggle}{\tt Files}{\tt Files}{\tt Faisonance}{\tt Files}{\tt For the transformation} \\ \ensuremath{{\tt STM32F103}_{\tt Toggle}{\tt Files}{\tt Files}{\tt Files}{\tt Faisonance}{\tt Faisonance}$

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。

Properties	× 1
Configuration: Standard	 v
Configuration: Standard - Clanguage - Font - Keywords - Normal word - Comment // style - Number - Comment // style - Number - Keyword - String - Character - UUID - Preprocessor - Operator - Identifier - String EOL - Verbatim - Regular Expression - Comment Line doc - ASM keywords - Comment doc keyw	Courier New; 10pt
- global class	
	Apply Close

MS ゴシックを選びます。OK をクリックします。

フォント			28
フォント名(E): MS Serif 予 MS UI Gothic 予 MS 明朝 の MS 明朝 の MV Boli の NS imSun の OCRB	スタイル(<u>V</u>): 標準	サイズ©): 10 11 12 14 18 20 ♥	<u>OK</u> キャンセル
文字飾り □ 取り消し線(M) □ 下線(W) 色(Q):	サンプル AaBbYyZ	z	
▲ 【 】	文字セット(<u>B):</u> 欧文	~	

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。

Properties			
Configuration:	Standard		~
 P: C language ML Assembler Misc. 		 Indents Show Indentation guide Tab width Indentation width Use Tabs Selection Foreground Color Selection Background Color Automatic indentation Indentation width	False 2 2 False 255; 255; 255 49; 106; 197 Block
			Apply Close

(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。

Properties		×
Configuration: Standard	~	
 Application Options Advanced ARM Options GCC compiler AS assembler LD Linker ARM Specific Options General Startup Scripts Libraries More RLink Configuration 	■ Scripts Use Default Script File Yes Script File mycrt0.1 Starter Kit limited No Yes No Starter Kit limited Wes When selected, this options prevents from that is larger than 32K	d h building a project
	Reset to default this group	options Close

(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。

Properties	8
Configuration: Standa	
 Application Options Advanced ARM Option Processor ARM toolsets Debug environmer GCC compiler AS assembler ID Linker RLink Configuration 	□ Processor □ Device ST M32FI 03VET 6 ○ ode size ST M32FI 03VEH6 □ Ode start ST M32FI 03VEH6 □ Data size ST M32FI 03VEH6 □ Data size ST M32FI 03VEH6 □ Data start ST M32FI 03VEH6 □ Data start ST M32FI 03VEH6 □ Data start ST M32FI 032CH6 □ ST M32FI 032CH6 ST M32FI 032DH6 □ ST M32FI 03ZDH6 ST M32FI 03ZEH6 □ ST M32FI 03ZEH6 ST M32FI 03ZET6 □ Device Selects the appropriate ARM derivative.
	Reset to default this group options Close

(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。

Properties			×
Configuration: Standard		~	
Application Options	Debug environment		
Advanced ARM Options	Debug tool	RLink	-
ARM toolsets	Format	Simulator SIM-ARM	
Debug environment	Code Offset	RLink	
GCC compiler	Explore code	No	
AS assembler	Start Mode	main() function entry	
LD Linker Durk Configuration	Start Symbol Address	main	
	Selects the debug tool.		
	Reset to	o default this group options Close	כ

(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。

Properties .	
Configuration: Standard	v
Application Options Info Directories Advanced ARM Options GCC compiler AS assembler LD Linker RLink Configuration	Advanced Options Crystal Frequency (MHz) 12.000 Click here to open options dialog box
	Reset to default this group options Close

<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。

Properties	
Configuration: Standard	~
Application Options Info Directories Advanced ARM Options GCC compiler As assembler DLinker RLink Configuration	Advanced Options Crystal Frequency (MHz) 8.000 Click here to open options dialog box
	Reset to default this group options Close

(2) Connect to RLink and read serial number \mathcal{E} *p* \mathcal{P} *p* \mathcal{P} *t* \mathcal{F} .

Debug options		JTAG para	meters	
Actions	Protocol O JTAG O SWD	Single of Number of before the Sum of IR before the	device Speed of the STAR clock 4000 (KHz). It should be less 4000 than the target CPU's clock 0 f devices 0 e target: 0 lengths 0 s target: 0	
Instant actions View RLink REva jumpers configuration for STM3x	Connect and read num	to RLink d serial ber	Connect to target and display information	Restore default Option Bytes now!
Erase target now!	Write targ nov	et FLASH v!	Dump target FLASH to hex file	USER FF WRP1 FF WRP3 FF
·	ОК		Cancel Help	

(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。

Туре:	New application to be built	~
Processor		Description: Advanced ARM-based 32-bit MCU 512K FLASH at address 0x8000000 64K RAM at address 0x20000000 6 Timers - 2 SPI - 2 IZC - 3 USART - USI - CAN - SDIO - 2 DAC - IZS - 2 ADC - 8 IOs DataSheet
Name: Location:	Application0 C:¥Program Files¥Raisonance¥Ride¥Examp	Browse Location
Launch	RBuilder to generate source files	

(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。

フォルダの参照	28
Select Directory:	
	_
🖃 🚞 ARM	^
🖃 🧰 STBee	
🗉 🛅 stb-led	
🔂 STM32F103 Toggle original	
□ STW32F1U3_Loggie storied	
STM32F103_Toggle USART	
🗁 USART	~
< >	
新しいマキルがの作成(M) OK をおいけれ	

(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。

New applic. Select th Type:	ation e type and location for your new application New application to be built	n.
Processor		Description: Advanced ARM-based 32-bit MCU 512K FLASH at address 0x8000000 64K RAM at address 0x20000000 6 Timers - 2 SPI - 2 I2C - 3 USART - USB - CAN - SDIO - 2 DAC - I2S - 2 ADC - 80 IOs DataSheet
Name: Location:	USART C¥DATA¥ARM¥STBee¥USART	Browse Location
Insert	to the current project	

(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	ルを選び	OK をクリック	ルます。
-----------------------	---	------------	-----------	------	----------	------

💹 Ride7				
[∶] <u>F</u> ile <u>E</u> dit <u>V</u> iew <u>P</u> rojec	t <u>D</u> ebug <u>S</u> cripts <u>C</u>)ptions <u>H</u> elp		
i 🔁 🕒 🖨 🖨 🖓 🐰	🗟 🛍 🗙 🗠 🗠 🎆	📇 😣 📮 🗒 📮	1 D 00 🖗 🕤	i 🕻 i 🖆 🕅 🟲 🛛 📜 🛱 📜
i Project 🛛 🕂	×			Documentation 4 ×
🥔 🛛 🕐 Show				🖃 🖻 Ride7 Documentatic 🛆
😑 💼 Project 'USART'				E-12 ARM
USAPT		1		
	start/Stop Utri+D	_		🖃 📂 Working with f
Set as S	StartUp application			🦳 🏉 Ride7 IDE i
Link	Ctrl+Shift+F9			- 🏉 Downloadir 🔽
Make				< <u>></u>
Build				Help 7 ×
Clean Clean				G 🖸 🏠 🛛 🖻
Open L	isting			~
USART Local Run	•	-		Ride7 IDE
Listing Din Library Din Open W	/ith			
Library Din		Itom		The Project
	Otalilas	Add Now Applied	ation	Explorer
Processo	Otherins	Now Folder		LAPIOIO
ARM tool	Ctri+v			While developing in
Library Direc		-		Ride7, the Project
You may enter Propert	ies Alt+Enter			Explorer view is your 🚽
path of the directory(ies)				< · · · >
Ready				7, I <u>N</u> S

(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。

#ifdef STM32F10X_CL

#define HSE_Value ((uint32_t)2500000) /*!< Value of the External oscillator in Hz */
#else</pre>

#define HSE_Value ((uint32_t)1200000) /*!< 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 の中の 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(GPIOD, GPIO_Pin_4, (BitAction)(1 - GPIO_ReadOutputDataBit(GPIOD, 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。

Properties	5
Configuration: Standard	~
 Application Options Info Directories Advanced ARM Options GCC compiler AS assembler LD Linker RLink Configuration 	Advanced Options Crystal Frequency (MHz) Click here to open options dialog box
	Reset to default this group options Close

<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。

Debug options	
Target Info Derivative: STM32F103RBT6 Actions Protocol Debug JTAG Download SWD	JTAG parameters Speed of the JTAG clock (KHz). It should be less than the target CPU's clock Number of devices before the target: 0 Number of devices of the target; Sum of IR lengths before the target: 0 Sum of IR lengths of the target;
Instant actions View RLink REva jumpers configuration for STM3x	to RLink ad serial nber Connect to target and display information Option Bytes action
Erase target now!	get FLASH w! Bump target FLASH to hex file WRP2 FF WRP3 FF WRP3 FF
ОК	Cancel Help

OK をクリックします。

Cortex Driver message	
Flash Erase successfully perfo	rmed.
(
OK	

(3) Write target FLASH now!をクリックします。

Debug options					×
Target Info Derivative: STM32F10 Actions P ✓ Debugi () ✓ Download ()	O3RBT6 Protocol D JTAG S SWD	TAG parameters Single device Number of devices before the target: Sum of IR lengths before the target:	Speed of the . (KHz). It shou than the targe 0	JTAG clock Jd be less et CPU's clock Number of devices after the target: Sum of IR lengths after the target: 0	
Instant actions View RLink REva jumpers configuration for STM3x	Connect to R and read se number	RLink erial Connec and info	t to target display rmation	Option Bytes action • Leave as is • Restore Default • Program Value to program	Restore default Option Bytes now!
Erase target now!	Write target F now!	FLASH Dum FLASH	p target to hex file	WRP0 FF	USER FF WRP1 FF WRP3 FF
(ок	Cancel	H	Help	

LEDSign.hex を選択します。

Flash Contents	3					2 🛛
ファイルの場所型:	📋 マイ ドキュメント		*	G 🦻	► 🔝 🏷	
していたファイル	 ■ LEDSign.hex ● マイ ピクチャ ● Visual Studio 3 ● マイ ミュージック 	2008				
じ デスクトップ						
71 F#1X2F						
₹1 ⊒ンピュータ						
२२ २७-७	ファイル名(<u>N</u>):	LEDSign.hex			~	<u>開((0)</u>
	ファイルの種類(工):	Intel HEX Files(*.hex)			*	キャンセル

「はい」をクリックします。

ARM Drv message	8
Program has been s Start execution?	successfully loaded.
はい(込)	(いいえ(<u>N</u>)

OK をクリックします。

Cortex Driver message
Flash programming successfully performed.
OK

(4) Cancel をクリックします。

Debug options				
Target Info Derivative: STM32F103VET6 Actions Protocol ✓ Debug ◯ JTAG ✓ Download ⓒ SWD	JTAG param Single de Number of d before the t Sum of IR let before the t	eters Speed of the J (KHz). It should than the targe levices arget: 0 N arget: 0 S arget: 0 0 0 0 0 0 0 0 0 0 0 0 0	TAG clock d be less 4000 t CPU's clock lumber of devices 0 fter the target: 0 um of IR lengths 0 fter the target: 0	
Instant actions View RLink REva jumpers configuration for STM3x	t to RLink ad serial mber	Connect to target and display information	Option Bytes action Leave as is Restore Default Program Value to program	Restore default Option Bytes now!
Erase target now!	get FLASH ow!	Dump target FLASH to hex file	Read-Out Protection WRP0 FF WRP2 FF	WRP1 FF WRP3 FF
ОК] []	ancel H	elp	

(5) Close をクリックします。

Properties	🛛
Configuration: All Configurations (Standard)	
Application Options Info Directories I-Advanced ARM Options GCC compiler AS assembler I-LD Linker I-RLink Configuration	Advanced Options Crystal Frequency (MHz) 12.000 Click here to open options dialog box
	Reset to default this group options Close

すいか村の電子工房

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