# Technical Note



### Silicon Labs 社 EFM32 クイックスタートガイド

2016年2月





TecStar -

Silicon Labs 社 EFM32

クイックスタートガイド

目次	
1 はじめに	4
2 開発環境のご紹介	5
2-1 ハードウェア	5
2-1-1 EFM32 Starter Kit	5
2-1-2 EFM32 Development Kit	9
2-2 ソフトウェア	10
2-2-1 Simplicity Studio	10
3 各種ドキュメント・サンプルコードの入手先	12
3-1 EFM32 のドキュメント	12
3-2 Starter Kit と Development Kit のユーザガイド	13
3-3 EFM32のAPI 情報	14
3-4 EFM32 のサンプルコード	15
4 ソフトウェア・インストール	16
4-1 Simplicity Studio のインストール	16
5 ハードウェア・セットアップ	16
5-1 Starter Kit のセットアップ	16
6 使用方法	17
6-1 サンプルコードを動かしてみる	17
6-2 デバッグ機能を使ってみる(Simplicity IDE)	21
6-3 消費電流を測定してみる(Energy Profiler)	23
6-4 ピン設定やペリフェラル設定をしてみる(Configurator)	25
6-5 電池寿命の見積もりをしてみる (energy Aware Battery)	29
6-6 ユーザ基板のプログラミング・デバッグを行ってみる	32
6-6-1 参考資料	32
6-6-2 ハードウェア接続	32
6-6-3 ハードウェア接続 (例)	33
6-6-4 デバッグ対象の切り替え (Kit Manager)	35
6-7 ユーザ基板の消費電流を測定する	36
6-7-1 VMCU をどこから取るか	36
6-7-2 Starter Kit 上の部品の消費電流を極限まで下げる	37
6-7-3 ユーザ基板の消費電流の測定手順	38
7 ソフトウェア設計	39
7-1 Cortex-M を初めて使う方に (ARM 社ドキュメント)	39

7-2 開発用のソースコードについて	40
7-3 ソースコードの追い方	42
7-4 割り込みハンドラ	42
7-5 ピン設定、ペリフェラル設定の流れ	42
7-6 ピン設定	43
7-7 ペリフェラル設定	47
7-7-1 USART (Asynchronous mode)	47
7-7-2 I2C	49
7-7-3 タイマ	51
7-7-4 タイマ (X 秒タイマの作り方)	53
7-7-5 CMU (ペリフェラル・クロックの周波数)	54
改版履歴	57
参考文献	57

#### 1 はじめに

この資料は、Silicon Laboratories(以下、Silicon Labs)社製 MCU EFM32 ファミリの開発環境について 簡易にまとめたものです。内容に誤りがないよう注意は払っておりますが、もし Silicon Labs 社が提供する ドキュメント等と差異がございましたら、メーカー提供のものを優先してご参照ください。

また、Silicon Labs 社の ナレッジベース(FAO)やコミュニティフォーラム(ユーザ同士で問題解決。 Silicon Labs のエンジニアも頻繁にコメントしています)には、本資料で取り上げていない様々な情報が記 載されております。

製品をご使用頂く過程で疑問や課題が生じることもあると思いますが、他のユーザが既に解決方法を 見つけている場合も多々ございます。非常に有益ですので、ぜひご活用下さい。

#### ◆ アクセス方法

Simplicity Studio から

✓ Resources					
Silicon Labs	Presentations and Brochures	Silicon Labs Community	Technical Support	<b>D</b> iversity	Silicon Labs Videos

Web Site から

http://community.silabs.com/t5/Forum/ct-p/Forum

◆ 使用方法						
					silabs.com	中文论坛
SILICON LABS CONTIN					Register   Sig	n In   Help
Home F	orums	Share	Training	Tools	Blog	
キーワードを入	カ (例:EFN	<b>/</b> 132)			Community	v 0
Silicon Labs Community : Microcontrolle	rs : 32-bit MCU				Category	о То 🔻
Post Message Options ▼				7	Board Knowledge Base Users	
Most Recent Posts	Knowledge	Base	Popular Posts			
					Previous 1 2 3	101 Next »
	<b>~</b> •.	L TZ 1		-		

Community か Knowledge Base を選択

#### 2 開発環境のご紹介

EFM32 の開発環境について、ハードウェアとソフトウェアに分けてご紹介します。

#### 2-1 ハードウェア

EFM32の開発環境としては、Starter KitとDevelopment Kitを用意しております。Starter Kitには周辺装置やセンサが実装されていますので、簡単に EFM32 の機能を確認できるようになっています。 Development Kit は拡張性が高く、より高度な検証を行えるようになっています。

#### 2-1-1 EFM32 Starter Kit

Starter Kit は、各ファミリに1種ずつ用意されています。同一ファミリであっても、USB 有無、LCD コントローラ有無などで幾つもの製品型番が用意されていますが、Starter Kit にはフルセットの MCU が実装されていますので、これを用いて設計を進めて頂くことが可能です。

Starter Kit には、下記が同梱されています。

Starter Kit

- •IAR Embedded Workbench 評価版
- ・ケーブル各種



ファミリ名	CPU Core	Starter Kit	実装されている型番
Wonder Gecko	Cortex M4F	EFM32WG-STK3800	EFM32WG990F256
Pearl Gecko	Cortex M4	SLSTK3401A(JGと共通)	EFM32PG1B200F256
Giant Gecko	Cortex M3	EFM32GG-STK3700	EFM32GG990F1024
Leopard Gecko	Cortex M3	EFM32LG-STK3600	EFM32LG990F256
Jade Gecko	Cortex M3	SLSTK3401A(PGと共通)	EFM32PG1B200F256
Gecko	Cortex M3	EFM32-G8XX-STK	EFM32G290F128
Tiny Gecko	Cortex M3	EFM32TG-STK3300	EFM32TG840F32
Happy Gecko	Cortex M0+	SLSTK3400A	EFM32HG322F64
Zero Gecko	Cortex M0+	EFM32ZG-STK3200	EFM32ZG222F32





#### ♦ Happy Gecko : SLSTK3400A



Zero Gecko : EFM32ZG-STK3200



#### 2-1-2 EFM32 Development Kit

Development Kit は3枚の基板から構成されています。EFM32が実装されたMCUプラグイン・ボードと、色々な部品を実装できるプロトタイピング・ボードの2つを、マザーボードに挿入して使用します。

Development Kit には、下記が同梱されています。

・EFM32 Development Kit マザーボード

- ・EFM32 MCU プラグイン・ボード
- ・EXP32 プロトタイピング・ボード
- •IAR Embedded Workbench 評価版
- Atollic TrueSTUDIO
- ・ケーブル各種

ファミリ名	CPU Core	Starter Kit	実装されている型番
Wonder Gecko	Cortex M4F	EFM32WG-DK3850	EFM32WG990F256
Giant Gecko	Cortex M3	EFM32GG-DK3750	EFM32GG990F1024
Leopard Gecko	Cortex M3	EFM32LG-DK3650	EFM32LG990F256
Gecko	Cortex M3	EFM32G-DK3550	EFM32G890F128

#### Giant Gecko : EFM32GG-DK3750



#### 2-2 ソフトウェア

EFM32の開発環境である Simplicity Studio を使用して設計を行うことになります。

#### 2-2-1 Simplicity Studio

Simplicity Studio は、EFM32 をターゲットとしたコンパイル・デバッグ・プログラミングを1 つのプラット で提供することができるソフトウェアです。統合開発環境(IDE)を中心に、非常に便利なツール群が充 実しています。同社製の 8bit MCU や無線 MCU も同一プラットフォームで開発が可能です。



ツール名	機能の概要
Simplicity IDE	統合開発環境(IDE)。 無償の GCC コンパイラを搭載
Energy Profiler	実機の消費電流値を測定することが可能
Configurator	ピン設定やペリフェラル設定を簡単に行うことができる
energy Aware Battery	消費電流値のシミュレータ機能。バッテリ寿命も簡単に算出
Flash Programmer	フラッシュ ROM のライト/イレース
Kit Manager	キット情報、デバッグ対象の選択

#### ◆ 消費電流が実測できます(Energy Profiler)

• • •	Energy F	Profiler - /Applications/SimplicityStudio_v3/de	eloper/sdks/efm32/v2/kits/common/drivers/segmenticd.c - Simplicity Studio	
👫 🛛 😽 EFM32 Wonder Gecko S	tarter Kit Board (4400 +		E 🛃 📶	
			PROFILER_README.txt	-
[emicd.out] for V	Vonder Gecko 380	00 Starter Kit (pr 🔸	3.9 unt16_t bitfield; unt13_t com, bit;	
Avg Current Avg Power	Total Energy Time Span	Node Save Reset Compare	ker value int i;	
Semion Counter 204.12 pA 673.90 pM	54,84 s ليو 130,14 s	Paused 1 1 1 1	<pre>length = strlen(string); index = 0;</pre>	
		4.94	<pre>/* If an update is in progress we must block, or LCD_SyncBusyDelay(@xFFFFFFF);</pre>	there m
			/* Freeze LCD to avoid partial updates */ LCD_FreezeEnable(true);	
			<pre>/* Turn all segments off */ SegmentLCO_AlphaNumberOff();</pre>	
=			<pre>/* Fill out all characters on display */ for (index = 0; index &lt; 7; index++) {</pre>	
1.2			if (index < length) {	
			data = (int) *string; } /* Reddien with source */	
			{ data = 0x20; /* SPACE */	
			} /* Defined letters currently starts at "SPACE" data = data = 0x20:	- ASCII
			<pre>/* Get fant fant this letter */ bitfield = EFM_Alphabet[data];</pre>	
			for (i = 0; i < 14; i++)	
IRQ TC.IRDNardler			<pre>1.001 + - 0 bit = EFM_Display.Text[index].bit[i]; com = EFM_Display.Text[index].com[i];</pre>	
Energy Profile (live)			- I if (bitfield & (1 << i))	
C Function	Energy	Contribution (%)	· · · · · · · · · · · · · · · · · · ·	
RTC_CounterGet	89.57 mJ	68.831%	/* Turn on segment */	
LCD_SyncBusyDelay	31.09 mJ	23.892%	LCD_SegmentSet(COM, DTE, tPde),	
CMU_OscillatorEnable	6.98 mJ	5.36%		
SegmentLCD_Write	865.92 µJ	0.665%	stringer	
LCD_SegmentSet	380.03 µJ	0.292%	a tringer,	
~~	204.40 µJ	0.157%	/* Enable update */	
BlinkTest	165.70 µJ	0.127%	LCD_FreezeEnable(false):	
delayTicks	114.68 µJ	0.088%	1	
SegmentLCD_Number	87.19 µJ	0.067%	·	
LCD_SegmentSetLow	86.21 µJ	0.066%		

◆ ピン設定やペリフェラル設定を簡単に行えます(Configurator)



◆ バッテリ寿命のシミュレーションが行えます(energy Aware Battery)



#### 3 各種ドキュメント・サンプルコードの入手先

EFM32の最新ドキュメント・サンプルコードの入手方法について紹介します。

#### 3-1 EFM32 のドキュメント

EFM32のドキュメントは、Simplicity Studioを経由してご入手頂けます。Simplicity Studioの入手先や インストール方法につきましては、「4.ソフトウェア・インストール」をご参照ください。

Simplicity Studio を起動し、Product の欄(下図の赤枠)に使用する製品型番を入力すると、青枠・緑 枠から下記情報をご入手頂けるようになります。

- ◆ Datasheet: データシート。スペック、ピン配置情報など
- ◆ Reference Manual: リファレンス・マニュアル。ペリフェラルの使用方法などの解説
- ◆ Other Documents: エラッタ(バグ情報)、Cortex-M ガイド
- ◆ Application Notes: アプリケーションノート、それに付随するサンプルコード
- ◆ Kit Documentation: Starter Kit のユーザガイド、回路・レイアウト情報

SILICON LABS	Simplic	ity Stu	dio	
Current Product	✓ Tools			
EFM32LG990F256	··· ≻-	¢.	$\equiv_{ij}^{(i)}\equiv$	۲
Favorites	Simplicity IDE	Energy Profiler	Configurator	Demos
> EFM32LG230F128	✓ Software and K	iits		
✓ EFM32LG990F256				
Core ARM Cortex-M3 Flash 256 kB			<b>Q</b>	
MHz 48 RAM 32 kB Digital I/O 87 ADC YES DAC YES	Software Documentation	Software Examples	Application Notes	Kit Documentation
Buy Sample	✓ Part Document	ation		
	Data Sheet	Reference	Other Documents	
S Refresh detected hardware				

また、ドキュメントは Silicon Labs 社の Web Site からもご入手可能です。

http://www.silabs.com/support/pages/document-library.aspx

製品型番を入力すると、関連ドキュメントがリストアップされます。

#### **Document Library**

The document library has all of Silicon Labs' technical documents conveniently located in one place. To find the documents you need, start by selecting one of the product categories below.

•
•
•

#### 3-2 Starter Kit と Development Kit のユーザガイド

以下に Starter Kit および Development Kit のユーザガイドのリンクを掲載します。Happy Gecko、 Pearl Gecko、Jade Gecko については Quick Start Guide へのリンクです。ユーザガイドは Simplicity Studio をインストールしてからご入手ください。

#### ◆ Starter Kit のユーザガイド

Wonder: http://www.silabs.com/Support%20Documents/TechnicalDocs/efm32wg-stk3800-ug.pdf Pearl: http://www.silabs.com/Support%20Documents/TechnicalDocs/QSG118.pdf Giant: http://www.silabs.com/Support%20Documents/TechnicalDocs/efm32lg-stk3600-ug.pdf Leopard: http://www.silabs.com/Support%20Documents/TechnicalDocs/QSG118.pdf Jade: http://www.silabs.com/Support%20Documents/TechnicalDocs/QSG118.pdf Gecko: http://www.silabs.com/Support%20Documents/TechnicalDocs/efm32-stk-documentation.zip Tiny: http://www.silabs.com/Support%20Documents/TechnicalDocs/efm32-stk-documentation.zip Zero: http://www.silabs.com/Support%20Documents/TechnicalDocs/EFM32HG-SLSTK3400A-QuickStartGuide.pdf

#### ◆ Development Kit のユーザガイド

Wonder: http://www.silabs.com/Support%20Documents/TechnicalDocs/efm32gg-dk3750-ug.pdf Giant: http://www.silabs.com/Support%20Documents/TechnicalDocs/efm32gg-dk3750-ug.pdf Leopard: http://www.silabs.com/Support%20Documents/TechnicalDocs/efm32lg-dk3650-ug.pdf Gecko: http://www.silabs.com/Support%20Documents/TechnicalDocs/efm32g-dk3550-ug.pdf

#### 3-3 EFM32のAPI 情報

ペリフェラルを制御するためライブラリ(API)が用意されており、それを使用することでソフト設計を円 滑に進めて頂くことが可能です。

Simplicity Studio を起動し、Product の欄に使用する製品型番を入力して、Software Documentation を クリックします。(インターネットに接続した状態で行ってください。)



Help が起動し、製品ファミリー覧が表示されますので、使用するファミリを選択します。

Device Documentation Welcome to the documentation for Silico	on Labs 32-bit MCUs and SoCs. Please select your do	evice from the list below.
EFM32 Gecko	EFM32 Tiny Gecko	EFM32 Giant Gecko
<ul> <li>Software Documentation</li> </ul>	<ul> <li>Software Documentation</li> </ul>	Software Documentation
EFM32 Leopard Gecko	EFM32 Wonder Gecko	EFM32 Zero Gecko
Software Documentation	<ul> <li>Software Documentation</li> </ul>	<ul> <li>Software Documentation</li> </ul>
EFM32 Happy Gecko	EFM32 Pearl Gecko	EFM32 Jade Gecko
Software Documentation	<ul> <li>Software Documentation</li> </ul>	<ul> <li>Software Documentation</li> </ul>
EZR32 Leopard Gecko	EZR32 Wonder Gecko	
Software Documentation	<ul> <li>Software Documentation</li> </ul>	

ウィンドウ左で閲覧したいライブラリや APIを選択すると、右側に情報が表示されます。

					efm32gg-doo	-4.2.1
Main Page	Modules	Data Structur	res Files	Documentation Home	sila Q Search	
<ul> <li>EFM32 Giant Gecko Sc</li> <li>Machiler</li> </ul>	oftware Documenta	EFM32 G	iant Gecko So	oftware Documentation	(	
► Parts		Welcome to	the software docume	ntation for the EFM32 Glant Gecko. He	ere, you will find documentation for	
► EM_Library		The Cl     The en	ASIS-CORE Device he	aders for the EFM32 Giant Gecko		
► EM_Drivers		• The er	ergyAware Drivers	ibrary		
▶ BSP		The Bo     The Bo	ard Support Packag	e for Starter Kits and Development Ki	5	
► Drivers		• The Us	5B Host and Device s	tack		
Data Structures		Please also s	ee Simplicity Studio fo	or precompiled demo applications, ap	plication notes and software example	5.
Files						
Documentation Hor	me					
silabs.com						
Main Page	Modules	Data Structu	res Files	Documentation Home	cila Or Search	
				bocamentation nome	Sild Q. Search	
♥ Chir	0	<u>.</u>		Documentation nome	Enumeration	ons   Functio
► CMU	O	GPIO			Enumeration	ons   Functio
CHIF ► CMU ► COMMON	0	GPIO EM_Library		Decantentation nome	Enumeradi	ons   Functio
<ul> <li>CMU</li> <li>► COMMON</li> <li>► DAC</li> </ul>	0	GPIO EM_Library General Pur	pose input/Output (0	iPIO) API	Enumeradi	ons   Functio
<ul> <li>CMU</li> <li>COMMON</li> <li>DAC</li> <li>DBG</li> </ul>	0	GPIO EM_Library General Pur	pose input/Output (C	SPIO) API	Sina 4, Search Enumerati	ons   Functio
CMU COMMON DAC DBG DBG DMA	0	GPIO EM_Library General Pur	pose Input/Output (C	SPIO) API	Sina K, Scalut	ons   Functio
<ul> <li>CMI</li> <li>CMU</li> <li>COMMON</li> <li>DAC</li> <li>DBG</li> <li>DMA</li> <li>EBI</li> </ul>	0	General Purp	pose input/Output (0	5P(Q) API	Sine 4, search Erunens	ons   Function
<ul> <li>CHIP</li> <li>CMU</li> <li>COMMON</li> <li>DAC</li> <li>DBG</li> <li>DMA</li> <li>EBI</li> <li>EMU</li> </ul>	¢	General Purp	pose Input/Output (C ations	5P(0) AP1	Sing of search	ons   Functio
<ul> <li>CHIP</li> <li>CMU</li> <li>COMMON</li> <li>DAC</li> <li>DBG</li> <li>DMA</li> <li>EBI</li> <li>EMU</li> <li>GPIO</li> </ul>	G	General Purp General Purp More Enumer enum GP	pose Input/Output (C ations	SPIO) API	Sind Ve startun	ons   Functio
CAMP     COMMON     COMMON     DAC     DBG     DMA     EBI     EMU     GPIO     I2C	©.	General Purp General Purp More Enumer enum GP enum GP	pose Input/Output (0 ations IO_Port_TypeDef IO_DriveMode_Type JoDriveMode_Type	SPIO) API	ANDARD	ons   Function
<ul> <li>CHIP</li> <li>COMMON</li> <li>DAC</li> <li>DBG</li> <li>DMA</li> <li>EBI</li> <li>EMU</li> <li>GPIO</li> <li>I2C</li> <li>INT</li> </ul>	©.	GPIO     EM_LBray     General Pury     General Pury     More     Enumer     enum GP     enum GP     gi     gi	pose input/Output (C ations IO_Port_TypeDef IO_DriveModeStand JoDriveModeLowe	IDEF { ard = GPIO_P_CTRL_DRIVEMODE_ST ard = GPIO_P_CTRL_DRIVEMODE_ST	ANDARD.	ons   Function
<ul> <li>CMIP</li> <li>CMIP</li> <li>COMMON</li> <li>DAC</li> <li>DBG</li> <li>DMA</li> <li>EBI</li> <li>EMU</li> <li>GPIO</li> <li>I2C</li> <li>INT</li> <li>LCD</li> </ul>	0	General Pury General Pury . More Enumer. enum GP enum GP	pose input/Output ( ations IO_Port_TypeDef IO_DriveModeStand JoDriveModeLowe JoDriveModeLowe	Definition of the second secon	ANDARD.	ons   Function
<ul> <li>CMIP</li> <li>CMIP</li> <li>COMMON</li> <li>DAC</li> <li>DBG</li> <li>DMA</li> <li>EBI</li> <li>EMU</li> <li>GPIO</li> <li>I2C</li> <li>INT</li> <li>LCD</li> <li>LESENSE</li> </ul>	0	General Pury General Pury . More Enumer enum GP enum GP sis sis sis	ations IO.Port.TypeDef Io.DriveMode.Type JoDriveModeStand JoDriveModeLow =	SPIO) API SPIO P_CTRL_DRIVEMODE_ST ard = GPIO_P_CTRL_DRIVEMODE_LOW GPIO_P_CTRL_DRIVEMODE_LOW GPIO_P_CTRL_DRIVEMODE_LOW	ANDARD.	ons   Functio
CHIP     COMP     COMMON     COMMON     DAC     DBG     DBG     DMA     EBI     EBI     EDI     COPIO     I2C     INT     LCCO     LESENSE     LETIMER	0	GPIO EN, Likrey General Pur More Enumer: enum GP enum GP enum GP si si si si si si si si si si si si si	ations IO.Port.TypeDef IO.DriveMode.Type JaDriveModeStand JaDriveModeLow = JaDriveModeLow = IO.JMode.TypeDef {	IPIO) API IDef { ard = GPIO_P_CTRL_DRIVEMODE_ST at = GPIO_P_CTRL_DRIVEMODE_LOW GPIO_P_CTRL_DRIVEMODE_LOW	ANDARD,	ons   Function

#### 3-4 EFM32のサンプルコード

TecStar =

EFM32 のサンプルコードは、Simplicity Studio を経由してご入手頂けます。Simplicity Studio の入手 先やインストール方法につきましては、「4.ソフトウェア・インストール」をご参照ください。

Simplicity Studio を起動し、Product の欄に使用する製品型番を入力して、Software Examples をクリックします。



Example Project ウィンドウにて、Kit で使用する Starter Kit が選択されていることを確認し、Next ボタンをクリックします。

New Silicon Labs MCU Project	_ <b>D</b> X
Example Project Select the kit, part, and SDK to search for examples.	
Kit:	
Leopard Gecko 3600 Starter Kit	•
	<u>Manage kits</u>
Part:	
EFM32LG990F256	Manage parts
EFM32 SDK (v3.20.5) (C:¥SiliconLabs¥SimplicityStudio¥v2¥developer¥sdks¥efm32¥v2)	•
	<u>Manage SDKs</u>
    	Cancel

生成したいサンプルコードを選択し、Next をクリックします。

Area New Silicon Labs MCU Project	
Example Project Select the project template to open in Simplicity IDE.	
type filter text	🔿 🕼
Leopard Gecko 3600 Starter Kit     STK3600_blink     STK3600_burtc     STK3600_clock     STK3600_emlcd	4 W
STK3600_emode STK3600_energy	-
Bink example. Template for new projects. This example project use EFM32 CMSIS and the emlib peripheral library to demonstrate the use of the LED's on the starter kit. This example is intended as a skeleton for new projects. Board: Silicon Labs EFM32LG_STK3600 Starter Kit	E
C < Back     Next >     Finish	Cancel

プロジェクト名を入力し、サンプルコードの生成フォルダを確認して、Finish ボタンをクリックします。プロジェクトとサンプルコードが生成されます。

🖛 New Silicon Labs MCU Proj	ect			
Project Configuration A The project location alread	dy exists and has contents. Its contents may be c	eleted or overwritten.		
Project name: STK3600_blin	ĸ			
Use default location	SimplicityStudio¥v2_workspace¥STK3600_blink			Browse
With project files: Link to sources Unk libraries and copy Copy contents	sources			
?	< Back	lext >	Finish	Cancel

生成が完了すると、プロジェクトが自動でロードされ、コンパイルが行える状態になります。

#### 4 ソフトウェア・インストール

EFM32 の評価に必要なソフトウェアをインストールします。

#### 4-1 Simplicity Studio のインストール

Simplicity Studio を下記アドレスからダウンロードし、install-studio.exe を実行してインストールを行います。インストール path に全角文字(日本語)が入らないようにご注意ください。

http://www.silabs.com/products/mcu/Pages/simplicity-studio.aspx

#### 5 ハードウェア・セットアップ

EFM32の評価に必要なハードウェアの設定を行います。

#### 5-1 Starter Kit のセットアップ

以下の手順で設定していきます。

- 1. BAT, USB, DBG の中から、基板に給電する方法を選びます。スイッチを DBG に切り替えます。
- 2. DBG とPCをUSB ケーブルで接続します



#### 6 使用方法

Starter KitとSimplicity Studioを使用した評価手順をご紹介します。ここではSTK3600(Leopard Gecko)を使用しておりますが、他のStarter Kit でも手順は同じです。

なお、各ツールから Simplicity Studioの Top 画面に戻るには、画面右上のアイコンを使用します。



#### 6-1 サンプルコードを動かしてみる

Starter Kit 上の LED を点滅させるサンプルコードを、ダウンロードして動作を見てみます。 Starter Kit を PC に接続すると、Simplicity Studio が Starter Kit を自動認識します。うまく認識してくれ ない場合には、Refresh detected hardware アイコンを押してみてください。

S	Refresh detected hardware	~
	-	

Software Examples を選択します。

✓ Software and Kits			
		Q	
Software Documentation	Software Examples	Application Notes	Kit Documentation

接続した Starter Kit に合せて、Kit, Part, SDK が自動で選ばれますので、Next をクリックします。

- New Silicon Labs MCU Project	
Example Project Select the kit, part, and SDK to search for examples.	
Kit:	
Leopard Gecko 3600 Starter Kit	• 0
	Manage kits
Part:	
EFM32LG990F256	•
	Manage parts
SDK:	
EFM32 SDK (v3.20.5) (C:\SiliconLabs\SimplicityStudio\v2\developer\sdks\efm32\v2)	• 0
	Manage SDKs
	Cancel

Example Project で STK3600\_blink(STKxxxx\_blink)を選択し、Next をクリックします。



プロジェクト名を入力し、作業フォルダを指定します。With project files では、サンプルコードをローカ ルにコピーして使うかどうかを指定します。指定が終わったら、Finish をクリックします。

Project Configuration	n			
The project locatio	n already exists and has contents. Its	contents may be deleted or overw	ritten.	
roject name: STK36	500_blink			
Use default locat	ion			
Location: C:	¥SimplicityStudio¥v2_workspace	e¥STK3600_blink		Browse
/ith project files:				
C Link to sources				
Link libraries an	id copy sources			
Copy contents				
0.117				

With project files	内容
Link to sources	ライブラリもソースも、オリジナルのものを使う。ライブラ
	リもソースも修正しない人向け。
Link libraries and copy sources	ライブラリはオリジナルのものを参照し、ソースコードは
	ローカルにコピーして使う。
Copy contents	ライブラリもソースも、ローカルにコピーして使う。ライブ
	ラリを修正する可能性がある人向け。

サンプルコードの準備が整うと、Simplicity IDE が起動します。Simplicity IDE の使い方については 「6-2 デバッグ機能を使ってみる」で詳しく紹介します。



この Simplicity IDE を使用して、サンプルコードをビルドし、Starter Kit にダウンロードします。まずはト ンカチのアイコン(Build)をクリックします。コンパイラが走り、サンプルコードがビルドされます。



ビルドが完了したら、次に虫のアイコン(Debug)をクリックし、Starter Kit にダウンロードします。

or Navigate Search Project Run 🛞 • 🔨 • 🏇

ダウンロードが完了すると、デバッグ用の画面に切り替わります。



ビルド用の画面と、デバッグ用の画面の切り替えは、ウィンドウ右上のアイコンで行います。



サンプルコードを実行します。下図の実行のアイコン(Resume)をクリックしてください。



Starter Kit 上の LED が、ゆっくりと点滅しているのが確認できます。STK3600の場合には、LCD 下の LED0 と LED1 が点滅します。



◆ LED の点滅スピードを変更してみましょう。

Development perspective アイコンをクリックして、ビルド用の画面に切り替えます。



TecStar =

画面中央に、コード(blink.c)が表示されていると思います。blink.cの最後あたりに、Delay(1000)という記述が見つかります。この数値が点滅時間を決定しています。

尾 bl	ink.c 🕱	
	<pre>/* Infinite blink loop */ while (1) {     BSP_LedToggle(0);     BSP_LedToggle(1);</pre>	*
	Delay(1000); } }	-
	< III	•

値を200に変更してみます。

TecStar =

.c	*blink.c 🔀	1	
	<pre>/* Infinite blink loop */ while (1) {     BSP_LedToggle(0);     BSP_LedToggle(1);     Delay(200); }</pre>		*
-	} }	+	-

あとは、ビルドして、ダウンロードして、実行します。

先ほどと同じ手順で、トンカチのアイコン(Build)⇒虫のアイコン(Debug)⇒実行のアイコン(Resume)の順にクリックします。

Starter KitのLEDの点滅が、先ほどよりも早くなったことを確認できるかと思います。

◆ サンプルコードのツリー構造

ビルド用の画面 (Development Perspective)を開くと、Project Explorer というウィンドウがあり、サンプ ルコードで使用されているライブラリが表示されています。



- ◆ BSP: Starter Kit 上の、MCU 外のペリフェラル (LCD、LED など)の定義
- CMSIS: 標準 Cortex-M と、ユーザが選択した製品
   (Tiny, Leopard, Giant...)との差分の定義
- ◆ emlib: MCU 内のペリフェラルの定義
- ♦ src: サンプルコード

MCU 内外のペリフェラルのライブラリが用意されていますので、簡単に評価・設計を行える環境が整っています。

#### 6-2 デバッグ機能を使ってみる (Simplicity IDE)

**TecStar** 

ソフトウェア・デバッグの際に使用する、ブレークポイント、ステップ実行などの機能は、Simplicity IDE が提供します。「6-1 サンプルコードを動かしてみる」でも紹介しましたが、Simplicity IDE には、ビルド用 の画面と、デバッグ用の画面が用意されています。ソフトウェア・デバッグは、デバッグ用の画面で行い ます。



ブレークポイントを設定するには、停止させたい行の左横をダブルクリックします。設定されると、水色の小さな〇印が表示されます。再度ダブルクリックすれば解除されます。





CPU Core	ハードウェア・ブレークポイントの数
Cortex M0+	4
Cortex M3	6
Cortex M4F	6

### TecStar -

◆ ステップ実行

各種ステップ実行に対応しています。



実機で実際に動作を見て頂くのが、判りやすいです。



◆ レジスタ値の閲覧・変更

レジスタ・変数の閲覧や変更は、下のウィンドウ(Register ウィンドウなど)で行うことができます。前回の停止から、値が変化した場合には黄色で表示されます。

🕪= Variables 🤷 Breal	kpoints 🚻 Registers 🕿	🖓 👯 Expressions 🖓 🖓	
			~
Name	Value	Description	^
⊳ 👬 General		General Purpose Registers	Ε
⊳ 🛗 MSC		MSC	Ē
▷ 👬 EMU		EMU	
⊳ 👬 RMU		RMU	
⊳ 👬 CMU		CMU	
▷ 👬 AES		AES	
▷ 👬 LESENSE		LESENSE	
⊳ 👬 EBI		EBI	
· *** 0010		0.0710	1

🕪= Variables 🔍 💁 Breakpoint	s 👭 Registers 🛛 🚱 E>
Name	Value
🔺 👬 General	
1111 R0	0xC8
888 R1	0x4
888 R2	0x30
888 R3	0xC8
888 R4	0x0
888 R5	0x0

6-3 消費電流を測定してみる (Energy Profiler)

Starter Kit には電流センサが搭載されており、消費電流測定ツール(Energy Profiler)と組み合わせる ことで nA レベルでの電流測定が可能です。Starter Kit には LCD など外部部品も実装されていますが、 MCU 単体の消費電流が測定できるように配慮されています。

ここではサンプルコードを使用して、消費電流測定ツール(Energy Profiler)の使用方法をご紹介します。

Energy Profiler を起動します。

✓ Tools							
Simplicity IDE	Energy Profiler	E Configurator	Oemos	Flash Programmer	Kit Manager	E SWO Terminal	energyAware Battery

Run メニューから、Run Demo を選択します。



STK3600 touch(STKxxxx touch)を選択します。Mode は Run under Energy Profiler に設定し、Enable Code Correlation にチェックが付いていることを確認してください。そして Start をクリックします。

Demos for EFM32 Leopard Ge	ecko Starter Kit
Name	Description
STK3600 rtx_tickless	Keil RTX RTOS - tick-less example
STK3600 touch	Capactive touch example.
STK3600 ucos2_port	uC/OS-II RTOS on EFM32 using example.
STK3600 ucos3_port	uC/OS-III RTOS on EFM32 using example.
STK3600 usbdcomposite	USB Composite Device example, MSD + CDC + Vendc
Capactive touch example.	(E)
This example demonstrates	the capacitive touch capability of the EFM32 e
Mode: Run under Energy Prof	iler 🗸
Enable Code Correlation	)
?	Start Cancel

このサンプルコードは、通常時は Idle(EM2)で待機しており、Starter Kit 上のタッチ・スライダーに触れ ると通常動作(EM0)に移行する、というモード遷移を含んだものです。EM0⇔EM2の遷移で消費電流 が大きく変化しますので、消費電流の観点では非常に判りやすいサンプルコードになっています。 Starter Kit 上のタッチ・スライダーに触ってみます。







画面下には、関数単位での消費電流値が表示されます。消費電流の最適化を行いたい場合に、どの関数に手を加えるのが効率的か?の手がかりとなります。

	🕎 Energy Profile (live) 🙁 🔪 🎼 Energy Profile (range) 🕎 Range 12725 s+3750 ms						
С	Function	Energy	Contribution (%) 🔻				
Ŵ	LCD_SyncBusyDelay	49.62 mJ	66.557%				
W	CMU_OscillatorEnable	11.71 mJ	15.713%	וו			
W	<unknown_functions></unknown_functions>	3.83 mJ	5.135%	ו			
W	LESENSE_IRQHandler	1.07 mJ	1.436%				

なお、消費電流波形(Profiling)の上でクリックをすると、その瞬間に実行されているコード部分にジャンプすることができます。消費電流が大きく変化した箇所を特定したい場合などに役立つ機能です。



注意: Cortex-M0+(Zero Gecko)は、関数単位でのプロファイリング機能、ソースコードとの連動機能 をサポートしていません。デバッグ時にだけ M3 を使用する、という方法もございます。

#### 6-4 ピン設定やペリフェラル設定をしてみる (Configurator)

Simplicity Studio では、ピン設定やペリフェラル設定を直感的に行うことができるツールを用意しています。ここでは設定ツール(Configurator)の使用方法をご紹介します。

#### Configurator をクリックします。



Configurator を使用するために、プロジェクトの作成が求められます。Kit で None を選択し、Part で 使用する MCU を選択し、Project name にプロジェクト名を入力してプロジェクトを作成します。

🕶 New Silicon Labs MCU Project	<u> </u>	🖙 New Silicon Labs MCU Project 📃	1 ×
Simplicity Configurator Project Select the kit, part, and SDK for the configuration.		Project Configuration Select the project name and location, and choose the device to configure.	2
Kit		Project name: myProject	_
None	<b>•</b> (1)	☑ Use default location	
	<u>Manage kits</u>	Location: C# #SimplicityStudio#v3_workspace#myProject Browse,	
Part			
EFM32LG990F256	• (1)	Device: EFM32LG990F64	
	Manage parts		
EF M32 SDK (V320.12) (C#SiliconLabs#SimplicityStudio#V3#developer#sdks#en	Manage SDKs		
	Manage obriss		
Kext > Finish	Cancel	Cancel     Sack Next > Finish Cancel	

Configurator が起動します。画面中央に2つのタブがあり、Default Mode Port I/O でピン設定を、 Default Mode Peripherals でペリフェラル設定を行うことができます。



#### • Default Mode Port I/O

myProjecthwconf 😫	□ □ 🗄 Outline 🛛	
DefaultMode Port I/O: PORTIO	⊟-€, >>> DefaultMode ⊕-477 Peripherals	
	PortA     P	
DefaultMode Port I/O     F DefaultMode Peripherals		- \
		-

• Default Mode Peripheral



◆ ADC0を設定してみましょう

ADC0を有効にし、ADC0の動作モードを設定し、ピン設定を行う、という流れになります。

ADC0 を有効にするには、Default Mode Peripherals に切り替えて、ADC0 にチェックを入れます。 Outline ウィンドウに ADC0 が追加されます。



ADC0 の設定を変更するには、Outline に表示された ADC0 を選択します。Properties ウィンドウに ADC0 の設定画面が表示されますので、動作設定を行います。Single sample mode にして、リファレンス 電圧は 2.5V に、ADC の入力には Channel0(L11 ピン)を使用する設定にしてみます。

operties of ADC0		Properties of ADC0	
ADC 0 Single sample mode Sc	an mode	ADC 0 Single sample mode Sca	an mode
Property	Value	Property	Value
- ADC Settings		🖃 Single sample mode settir	ngs
Oversampling	2×	Single sample mode	Enabled
Low pass filter mode	None	PRS input	None
Warmup mode	Shut down after each c.	Acquisition time (ADC cloc	sk c: 1
ADC Frequency (Hz)	7000000 (0×6ACFC0)	Reference	Internal 2.5V
Conversion tailgating	Disabled	Resolution	12 bit
Input Settings		Input	Channel 0 (PD0)
Channel 0 (PD0)	Enabled	Left-adjust result	Disabled
Channel 1 (PD1)	Disabled	Continuous conversion	Disabled
Channel 2 (PD2)	Disabled		
Channel 3 (PD3)	Disabled		
Channel 4 (PD4)	Disabled		
Channel 5 (PD5)	Disabled		
Channel 6 (PD6)	Disabled		
Channel 7 (PD7)	Disabled		

続いてピン設定を行います。Default Mode Port I/O ウィンドウに切り替え、Outline に表示された Port I/O を選択します。Peripheral Mapping ウィンドウにペリフェラル一覧が表示されますので、そこから

ſ	🔲 Properties 🗰 Periph	neral Mapping 🟻 🕅		- 6		🔲 Properties 🏢 Perip	heral Mapping 🕅		- E
ľ	DefaultMode : PORT	0				DefaultMode : PORT	10		
	0 -								<b>_</b>
	ADC0	СН0	PD0			ADC0	СН0	PD0	_
		СН1	PD1				СН1		
		CH2	PD2				CH2		
		СН3	PD3				СН3		
		СН4	PD4						
		CH5	PD5						
		CH6	PD6		'				
		CH7	PD7				СН7		
	D BU	STAT		<b>T</b>		D BU	STAT		•

ADC0 を探して、ピン設定を ADC0 の入力設定に合わせます。(つまり CH0 だけを有効にする)

設定が完了すると、Default Mode Port I/O ウィンドウに、ADC0 入力(L11ピン)の設定が反映されました。使用中のピンは紫色になります。ピンをクリックすると、そのピンの設定(入出力、プルアップの有無など)も設定が行えます。

	2 DEC_2	🔲 Properties 🛿 🗯 Peripheral Mapping	🛃 🛛 🖓 🖬
	0 (611)	Properties of PD0	
ピッナカリックナスト 発知	6 P07	Port Pin	
ヒノをクリック9 ると、計械		Property	Value
	9 00	Settings	
設定が行える。	PD7	Pin mode	Disabled
		Pullup Pullup	Disabled
	0) (J11)	Custom pin name	
	8 PD4	Reserve	Not reserved
	0 (K11)		
にアサインされた			
	2,0 PD0 ADC0,CH0		

◆ ピン配置の矛盾を解決(Resolve Pin Conflicts)

ペリフェラルごとに使用可能なピンはある程度決まっています。使用するペリフェラルが増えてくると、 ピン配置が競合してしまい、うまくピン配置が行えない場合が出てきます。それを解消してくれるのが Resolve Pin Conflicts 機能です。競合が生じないピン配置の組み合わせを探してくれます。

機能紹介のために、ピン配置が競合させてみます。LCDとUSART0をイネーブルにしてみてください。 ピン4本分の設定が競合してしまい、ペリフェラル設定とピン設定が、競合状況を示す赤色になります。

C *myProject/wconf 22	D BE Outline 13	-
DefaultMode Port I/O: PORTIO	DefaultMode     Peripherals	
	Properties Pericharal Macoine 13	0
	DefaultNode : PORTIO	
H1         H2         H2         H4         H5         H1         H1<		
967 968 9669 10000 9691 9693 10000 9699 9699 Language		
DefaultMode Port I/O 📢 DefaultMode Peripherals		
÷0	USART2 P CLK	

このような場合、Resolve Pin Conflict 機能を使います。Outline の Port IO を選択して(或いは Default Mode I/O ウィンドウの上で)右クリックし、Resolve Pin Conflicts を選択します。



解決案を提示してくれますので、OKをクリックすれば問題が生じないピン配置に割り当てなおしてくれます。競合を示す赤色のピンがなくなり、問題が解消されました。



◆ コードに反映させる(Generate Source)

設定が完了したら、Default Mode I/O ウィンドウか、Default Mode Peripheral ウィンドウの上で右クリックし、Generate Source を選択します。ペリフェラル設定、ピン設定を含んだ C コードが生成されます。



6-5 電池寿命の見積もりをしてみる (energy Aware Battery)

バッテリー・アプリケーションにおいて、電池寿命の見積もりは非常に重要です。しかし MCU には幾 つもの消費電力モードがあり、多数のペリフェラルがあり、電池寿命の見積もりを複雑にしています。 Simplicity Studio には、電池寿命を簡単に見積もることが可能なツールを用意しています。ここでは、電 池寿命の見積もりツール(energy Aware Battery)の使用方法を紹介します。

電池寿命を見積もるにあたって、MCU の動作を特定する必要があります。ここでは以下の動作を例 に取って、電池寿命を見積もっていきます。通常動作(EM0)で10msの作業をし、Idle(EM2)に移行して 1s 待機し、という動作を繰り返し行う、という設定です。



energy Aware Battery をクリックします。

✓ Tools							
Simplicity IDE	Energy Profiler	E Configurator	Oemos	Flash Programmer	E Kit Manager	SWO Terminal	energy Aware Battery

File メニューから New を選択します。

ᡖ eABattery					
File Simu	ulation Help				
🗋 New	Ctrl+N				
🚘 Open	Ctrl+0				

シミュレーションを行いたいファミリ名を選択し、OK をクリックします。

🐻 New Projec	? ×	
Project Name	Untitled	
MCU Family	Leopard Gecko	-
	Cancel	Ok

◆ EM0の期間(ADC 有効、10ms)を登録しましょう
 Add State をクリックします。



Energy ModeをEM0に設定し、期間を10msとします。ペリフェラルはADCを有効にしますので、ADC にチェックを付けます。設定が終わったら、Close をクリックします。

	Z eABattery	
	EM0 - 2.92mA	
	Description State 1 10.00ms	━ 期間を指定
	Energy Mode EM0  1.0με 100.0με 10.0ms 1.0ε 1 m 1 h 1 d 1 y 15 y	
消費電力モード	High Frequency Clocks	
を指定		
	Peripherals	H m + 7
		使用する
	3.000µA 3.000µA 33.000µA	<b>ペリフェラルに</b>
	DAC0         □         DMA         □         EBI         □         12C0         □         12C1           38.000µA         113.880µA         21.840µA         87.500µA         87.500µA         87.500µA	チェック
	LCD         Ε           8.550μA         Ε           150.000nA         Ε           150.000nA         Ε	
	PCNT0         PCNT1         PCNT2         PRS         RTC           1UU.UUUnA         1UU.UUUnA         39.340µA         1UU.UUUnA	
	Close	

なお、ADC の横のボタンをクリックすると、動作モードをより詳細に指定することが可能です。

Sampling Profile	10kS, 12bit, internal reference, Warmup 0b01 💌
I    🔲 I2C	10kS, 12bit, internal reference, Warmup 0b10
A 875000A 875000A	10kS, 12bit, internal reference, Warmup 0b01
	10kS, 12bit, internal reference, Warmup 0b00
	1MS, 12bit, external reference

また、MCU だけではなく、外部部品での消費電流も含めてシミュレーションを行いたい場合には、 External Current (ペリフェラルを下にスクロールしていくとあります)に数値を入力してください。

– External Cu	irrent		_					
📝 Enable e	external co	nponent curi	rent					
	1.00mA							
						[		
_ ·								
1.0nA	10.0nA	100.0nA	1.0µA	10.0µA	100.0µA	1,000.0µA	10.0mA	100.0mA

◆ EM2 の期間(RTC 有効、1s)を登録しましょう

EM0 の時と同じ作業を、EM2 についても行います。 Add State をクリックします。

Energy Mode を EM2 に設定し、期間を 1s とします。ペリフェラルは RTC を有効にしますので、RTC に チェックを付けます。設定が終わったら、Close をクリックします。 登録が完了すると、次図のように表示されます。

TecStar —

EMO State 1	EM2 State 2	Add state			
*		0		Cl	
10.0mA -		Curr	ent consumption	n profile	
1,000.0µA					
10.0µA -					
1.0µA					
100.0nA					
10.0nA -					
0.0s	200.0ms	400.0ms	600.0ms	800.0ms	1.0s

#### ◆ シミュレーションを行いましょう

使用する電池を決めます。今回は、CR2016を並列に2個繋いだ場合を考えます。Cell type で CR2016を選択し、Number of parallels(並列の電池数)を2、Cells in series(直列の電池数)を1にしま す。

使用する雷池を選択	Battery setup		a
	Cell type	CR2016 -	[ —       +  ]
電池数(並列)>	Number of parallels	2	JJ
7	Cells in series	1	
電池数(直列) 🦯			

Simulation メニューから、Start Simulation を選択します。

File	Simulation	Help
	🕨 Start Sin	nulation

シミュレーションが実行され、この使用条件の場合には電池寿命が 260 日程度であることが算出できました。



#### 6-6 ユーザ基板のプログラミング・デバッグを行ってみる

Starter Kitを使用することで、ユーザ基板上の EFM32 に対して、プログラミング或いはデバッグを行う ことが可能です。その手順について紹介します。



#### 6-6-1 参考資料

•AN0042: EFM32 Debug and Trace

• AN0062: Programming Internal Flash Over the Serial Wire Debug Interface

・各ファミリの Starter Kit ユーザガイド

http://community.silabs.com/t5/32-bit-MCU-Knowledge-Base/Using-an-EFM32-Starter-Kit-as-an-exter nal-debugger/ta-p/124194

#### 6-6-2 ハードウェア接続

Starter Kit の Debug Connector を介して、ユーザ基板に接続します。 Debug Connector は、 Starter Kit の右上にある 20 ピンのコネクタです。



Debug Connector のピン配置情報は、Starter Kit ユーザガイドに記載されています。下図は、 EFM32LG-STK3600 ユーザガイドからの抜粋です。このうち、SWCLK、SWDIO、SWO、#RESET、 VTARGET、GND の計 6 ピンを使用します。

		I	
VTARGET 1		2	NC
#TRST 3		4	GND
TDI 5		6	GND
TMS/SWDIO 7		8	GND
TCK/SWCLK 9	900	10	GND
RTCK 11	╸╸५	12	GND
TDO/SWO 13	00	14	GND
#RESET 15	00	16	GND
PD 17		18	Cable Detect
PD 19		20	GND
		1	

SWCLK	EFM32(ユーザ基板上)の該当ピンに接続してください。直結で結構です。
SWDIO	EFM32(ユーザ基板上)の該当ピンに接続してください。直結で結構です。
SWO	EFM32(ユーザ基板上)の該当ピンに接続してください。直結で結構です。
#RESET	EFM32(ユーザ基板上)の該当ピンに接続してください。直結で結構です。
VTARGET	EFM32(ユーザ基板上)への供給電源に接続してください。Starter Kit とユー
	ザ基板の信号レベルを合せるために使用します。接続し忘れると、Starter
	Kit からユーザ基板を認識できませんので、ご注意ください。
GND	Starter Kit の GND と、ユーザ基板の GND を接続してください。

#### 6-6-3 ハードウェア接続 (例)

例として、EFM32LG990の場合を取り上げて説明します。

まず、データシートの Alternate Functionality Pinout で、SWCLK、SWDIO、SWO がどの Pin Name (ピン名称)に割り当てられているかを確認します。下図のとおり、SWCLK は PF0、SWDIO は PF1 に割り当てられています。SWO は PF2/PD1/PD2 のいずれかに割り当てが可能となっていますが、デフォルトは Location0 の PF2 になっています。

Alternate			LOC	ATION			
Functionality	0	1	2	3	4		Description
DBG_SWCLK	PF0	PF0	PF0	PF0			Debug-interface Serial Wire clock input. Note that this function is enabled to pin out of reset, and has a built-in pull down.
DBG_SWDIO	PF1	PF1	PF1	PF1			Debug-interface Serial Wire data input / output. Note that this function is enabled to pin out of reset, and has a built-in pull up.
DBG_SWO	PF2		PD1	PD2			Debug-interface Serial Wire viewer Output. Note that this function is not enabled after reset, and must be enabled by software to be used.

次に、データシートの Pinout で、どの Pin Name (ピン名称)が、どの Pin Number (ピン番号)に割り振られているかを確認します。

PF0(SWCLK)はE8ピン、PF1(SWDIO)はD8ピン、PF2(SWO#0)はC8ピン、#RESETはK6ピンに割り当てられていることが判ります。

B	GA112 Pin# and Name		Description								
Pin #	Pin Name	Analog	ЕВІ	Timers	Communication	Other					
E8	PF0			TIM0_CC0 #5 LETIM0_OUT0 #2	US1_CLK #2 LEU0_TX #3 I2C0_SDA #5	DBG_SWCLK #0/1/2/3					
D8	PF1			TIM0_CC1 #5 LETIM0_OUT1 #2	US1_CS #2 LEU0_RX #3 I2C0_SCL #5	DBG_SWDIO #0/1/2/3 GPIO_EM4WU3					
C8	PF2	LCD_SEG0	EBI_ARDY #0/1/2	TIM0_CC2 #5	LEU0_TX #4	ACMP1_O #0 DBG_SWO #0 GPIO_EM4WU4					
К6	RESETn	Reset input, active low. To apply an external res that reset is released.	Reset input, active low. To apply an external reset source to this pin, it is required to only drive this pin low during reset, and let the internal pull-up ensure that reset is released.								

上記を見ると、デバッグピンには、TIM\_CC0#5 や US1\_CLK#2 など、他の機能を割り当てることが可能 になっていることが判ります。デフォルトでは、デバッグ用のピンに設定されていますので、ユーザプログラ ムから設定変更を行うことになります。しかし、特別な理由がない限り、他の機能は割り当てずに、デバッ グ専用ピンとしてご使用になることを強くお勧めします。いったん設定変更を行うと、デバッグ回路にアクセ スできなくなりますので、プログラムの書き替えも行えなくなる可能性があります。

#### 6-6-4 デバッグ対象の切り替え (Kit Manager)

Kit Manager の Debug Mode 選択メニューにて、Out を選択してください。MCU Information が Unknown に変わります。

✓ Tools							
>-	C,	E <sup>nn</sup> E	۲	<b>*</b>			
Simplicity IDE	Energy Profiler	Configurator	Demos	Flash Programmer	Kit Manager	SWO Terminal	energyAware Battery
Kit Na Ba Se Fii	t <b>Information</b> ame aard Name rial Number rmware Version				EFM32 Le	opard Gecko Starter BRD2201A 1 0v	Kit Board Rev. A02 43201296 9p16b546
	Jpdate Kit Istallation Package					Browse   Install Pa	ackage
	ebug Mode: Out 💌						
Na Ar Fi Lo	ame chitecture ash Size cked					~	Unknown Unknown Unknown Unknown

次に、Simplicity Studio の Top 画面に戻り、画面左下の Detected Hardware に表示されたスターター キットの横のボタンを押し、Select Target Part を選択します。

<b>G</b> Refresh detected hardware	✓ Resources
Detected Hardware	
✓	Rart Not Netsound
Unknown Device (use menu to select or detect target part)	Select Target Part Configure Adapter
	👆 Add as Favorite

Target Selection ウィンドウが表示されますので、Detect Target ボタンを押します。ユーザ基板上の MCU が認識されました。あとは、Starter Kit 上の EFM32 と同様に使用できます。

Target Selection for EFM32 Leopard Gecko Starter Kit X      Adapter Details     Serial Number 440026487     Vendor Segger      Target Settings	Target Selection for EFM32 Leopard Gecko Starter Kit      Adapter Details     Serial Number 440026487     Vendor Segger     Target Settings
Part No Part Target Interface SWD Detect Target Current detected part: Unknown Forget Target Protect from device discovery	Part EFM32TG840F32  Target Interface SWD  Detect Target Current detected part: EFM32TG840F32 Forget Target Protect from device discovery
OK Cancel	OK Cancel

6-7 ユーザ基板の消費電流を測定する

ユーザ基板の消費電流を測定するためには、次の2つの処理を行う必要があります。

- ◆ ユーザ基板には、Starter Kit の VMCU ピンから給電を行う
- ◆ Starter Kit 上の部品(EFM32, その他周辺部品)の消費電流を極限まで下げる

Starter Kit には、電流モニタ用の回路が実装されています。下図は STK3600(Leopard Gecko)のユー ザガイドからの抜粋ですが、Debug 用 USB から 5V 給電を受け、LDO で 3.3V を作り、それを VMCU と して Starter Kit 上の EFM32 やセンサ、ペリフェラルに給電しています。Energy Profiler は VMCU の消 費電流をモニタするようになっています。



#### 6-7-1 VMCU をどこから取るか

VMCU は Starter Kit の Expansion Header から出ています。Expansion Header は、Starter Kit の右側 にある 20 ピンのコネクタです。



Expansion Header のピン配置情報は、Starter Kit ユーザガイドに記載されています。下図は、 EFM32LG-STK3600 ユーザガイドからの抜粋です。2 ピンに VMCU が配置されています。

GND	1	•	0	2	VMCU
PC0	3	•	•	4	PD0
PC3	5	•	•	6	PD1
PC4	7	•		8	PD2
PC5	9	•	•	10	PD3
PB11	11	•		12	PD4
PB12	13		•	14	PD5
PC6	15	•	•	16	PD6
PD7	17	•	•	18	5V
GND	19	•	•	20	3V3

6-7-2 Starter Kit 上の部品の消費電流を極限まで下げる

Starter Kit 上の EFM32 は、EM4(消費電流がもっとも低い電力モード)に移行することで消費電流を 極限まで下げることができます。また EFM32 が EM4 にいる間は、センサやペリフェラルはほとんど電流 を消費しません。ですから EFM32を EM4 にすれば良い事になりますが、電力モードを切り替えるデモコ ードがありますので、それを利用します。デモコードのダウンロード手順は以下の通りです。

- Starter KitをPCに接続し、Detect Connected Device でStarter Kitを認識させます。ユーザ基板は 接続しない状態で行ってください。もしStarter Kitを認識しない場合には、Kit Manager のDebug Mode が MCU になっているか確認してください。
- 2. Energy Profiler を起動します。
- Run メニューから Profile Demo を選択します。実行可能なデモの一覧が表示されますので、 STK3600 emlcd を選択し、Start を押します。Energy Profiler に電流波形が表示されればダウンロ ード完了です。

🖛 Demos for EFM32 Leopard Gecko Starter Kit 🛛 🗙					
Name STK3600 burtc STK3600 clock	Description Backup power domain RTC example. Wall Clock example using the segment LCD				
STK3600 emicd STK3600 emode STK3600 freertos blink STK3600 freertos blink	Energy Modes with segment LCD example. Select a single energy mode, and stay there. Board Support Package API for voltage and current. FreeRTOS - Blink example EmeRTOS - Ticklose example				
Energy Modes with segment LCD example.					
EM2 is used for dela         "Outo conduit" and proceeding         Mode:       Run under Energy Prof         Image:       Enable Code Correlation ()	ys in between refreshing the LCD displ				
?	Start Cancel				

このデモは、ボタンを押すと電力モードを順々に切り替えていきます。STK3600の場合は、PB0を 押すとEM0⇒EM1⇒EM2⇒EM3の順に切り替えていき、PB1を押すとEM0⇒EM1⇒EM2⇒EM4の 順に切り替えていきます。今回は EM4 にしたいですので PB1を押すことになります。LCD に動作説 明が表示されますので、そちらも参考にしてください。

試しにPB1を押してEM4にしましょう。消費電流が極限まで下げられたことが確認できます。(下図で、測定結果がマイナスになっているのは読み取り誤差です)



#### 6-7-3 ユーザ基板の消費電流の測定手順

手順は以下の通りです。

- 1. 「6-7-1 VMCU をどこから取るか」を参考に、VMCU をユーザ基板に接続します。
- 2. 「6-6 ユーザ基板のプログラミング・デバッグを行ってみる」を参考に、デバッグ経路をユーザ基板 に接続し、Kit Manager の Debug Mode を Out に変更します。
- 3. 「6-7-2 Starter Kit 上の部品の消費電流を極限まで下げる」を参考に、PB1 を押して Starter Kit 上の EFM32 を EM4 に移行させます。
- 4. (必要であれば、ユーザ基板のプログラムの書き換えなどを行います。)
- 5. Energy Profiler を起動し、Run メニューから Profile Running Program を選択します。ユーザ基板の 消費電流がモニタできます。



7 ソフトウェア設計

ソフトウェア設計に役立つ情報をご紹介します。

7-1 Cortex-M を初めて使う方に (ARM 社ドキュメント)

ARM 社が、Cortex-M そのものに関する日本語ドキュメントを提供しております。ARM 社の Web Site からご入手頂くことができます。

◆ Cortex-M4 テクニカル・リファレンスマニュアル

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439cj/DDI0439CJ\_cortex\_m4\_r0p1\_trm\_ja-JP.pdf

◆ Cortex-M3 テクニカル・リファレンスマニュアル

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337ij/DDI0337IJ cortexm3 r2p1 trm japanese.pdf

◆ Cortex-M0+ テクニカル・リファレンスマニュアル

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0484cj/DDI0484CJ\_cortex\_m0p\_r0p0\_trm\_JP.pdf

◆ Cortex-M0+ デバイス・ジェネリックユーザ・ガイド

http://infocenter.arm.com/help/topic/com.arm.doc.dui0662bj/DUI0662BJ cortex m0p r0p1 dgug JP.pdf

また、役立ちそうな日本語アプリケーションノートもご紹介します。

◆ Cortex-M3 組み込みソフトウェア開発(アプリケーションノート 179)

http://infocenter.arm.com/help/topic/com.arm.doc.dai0179bj/DAI0179BJ.pdf

◆ PIC マイコンから Cortex-M3 への置き換え(アプリケーションノート 234)

http://infocenter.arm.com/help/topic/com.arm.doc.dai0234a/DAI0234A migrating from pic to m3.pdf

◆ 8051 マイコンから Cortex-M への置き換え(アプリケーションノート 237)

http://infocenter.arm.com/help/topic/com.arm.doc.dai0237a/DAI0237A\_migrating\_from\_8051\_to\_Cortex\_M.pdf

#### 7-2 開発用のソースコードについて 📜 an ソースコードー式は、下記フォルダにインストールされています。 👢 CMSIS ¥¥SiliconLabs¥SimplicityStudio¥v2¥developer¥sdks¥efm32¥v2 👢 Device 👢 emdrv 📜 emlib ◆ アプリケーションノート(¥an フォルダ) 📜 kits EFM32を使用した応用例(各種ペリフェラル、割り込み、Bootloader 等)を 🗼 reptile 提供しています。 📙 StudioModules 📜 university 用意されたソースコードを、簡単に Simplicity Studio から呼び出す事が 👗 usb

できますので、その手順を紹介します。

Simplicity Studio を起動し、Product の欄に使用する製品型番を入力して、Application Notes をクリッ クします。

➤ Software a	ind Kits		
		Q	
Software Documentat	Software Examples	Application Notes	Kit Documentat

使用したいアプリケーションノートを選択し、Import Project ボタンを押すと、Simplicity Studio 用のプロ ジェクトが生成されます。

AN0009 Getting Started with EFM32 AN0011 I2C Master and Slave Operation AN0012 GPI0 AN0013 Direct Memory Access
The EFM32 I2C module allows simple, robust and cost effective communication between integrated circuits using only one data and one clock line. This application note demonstrates how to use the EFM32 I2C module in multimaster mode. Two EFM32s are
Filter by selected product line
Import Project     Open Folder     Open     Close

◆ EFM32 ライブラリ(¥emlib フォルダ)

EFM32で共通に持つ各種機能の APIを提供しています。UART, I2C, タイマ, RTC, GPIO などが用意 されています。ヘッダおよびソースコードは、それぞれ下記フォルダに保存されています。

ヘッダ: ¥emlib¥inc

ソースコード: ¥emlib¥src

◆ EFM32 ドライバ (¥emdrv フォルダ)

EFM32 ライブラリ以外の機能の APIを提供しています。内蔵 Flash、スリープモード、SPI などが用意 されています。

APIの詳細情報については、Software Documentation(3-3 EFM32 の API 情報)から入手頂けます。 (オンラインでのみご使用頂けます)

EM\_Library にて、API について確認頂けます。各関数の引数が一覧できます。

Main Page	Modules	Da	ta Structures	Files	Documentation Home	sila	Q- Search
<ul> <li>EFM32 Leopard Ge</li> <li>Modules</li> <li>Parts</li> </ul>	cko Software Docum	e	void ADC_InitScar	n ( ADC_TypeDe const ADC_lı )	f* adc, hitScan_TypeDef* init		
▼ EM_Library ► ACMP ▼ ADC			Please refer to AD When selecting an other references,	<b>C_Start()</b> for stand n external refere the calibration i	arting scan sequence. nce, the gain and offset calibration i s updated with values defined durin	must be s g manufa	et explicitly (CAL register). For acturing.
<ul> <li>ADC_Init_</li> <li>ADC_Init_</li> </ul>	_TypeDef ScanInput_TypeDef		Note This functio	n will stop any c	ngoing scan sequence.		
<ul> <li>ADC_Inits</li> <li>ADC_Inits</li> </ul>	Scan_TypeDef Single_TypeDef		Parameters [in] adc P [in] init P	ointer to ADC po	eripheral register block. itialization structure.		
ADC_INIT	_DEFAULT SCAN_DEFAULT		Definition at line 6	579 of file em_a	lc.c.		

Data Structures にて、データ構造について確認頂けます。

Main Page Modules	Data Structures         Files         Documentation Home         sila q: Search
Data Structures Data Structu	re Index Data Fields
<ul> <li>EFM32 Leopard Gecko Software Docume</li> <li>Modules</li> </ul>	Data Structures
Data Structures	Here are the data structures with brief descriptions:
Data Structures	C ACMP_CapsenseInit_TypeDef
ACMP_CapsenseInit_TypeDef	C ACMP_Init_TypeDef
ACMP_Init_TypeDef	C ACMP_TypeDef
ACMP_TypeDef	c ADC_Init_TypeDef
ADC_Init_TypeDef	C ADC_InitScan_TypeDef
ADC_InitScan_TypeDef	C ADC_InitScanInput_TypeDef
ADC InitScanInput TypeDef	C ADC_InitSingle_TypeDef
	C ADC_TypeDef
> ADC_INIGINGIC_TypeDet	C AES_TypeDef
► AEC TypeDef	C ARING_TypeDef Defines segment COM and BIT fields for A-wheel (suited for

◆ キット別サンプルアプリケーション(¥kits フォルダ)
 EFM32 ライブラリ/ドライバを使った簡易アプリを提供しています。

### TecStar :

#### 7-3 ソースコードの追い方

Simplicity IDE でソースコードを追うための方法を紹介します。

◆ 変数や関数を定義している記述を探す



③変数や関数の定義にジャンプ

#### 7-4 割り込みハンドラ

EFM32の割り込みコントローラは、Cortex-M系コアに内包されるNVICいうブロックに含まれており、市場にある多くのCortex-Mコア搭載品と動作に差がありません。

EFM32 のリファレンスマニュアルの NVIC の関連事項のほか、ARM 社のドキュメント内の NVIC の記述が参照頂けます。

参考) 日本語の Web サイトとしては、APS ACADEMY の Cortex-M 入門編 第13回で、非常に判り やすく説明されています。「APS 例外/割り込み処理」で検索してみてください。

7-5 ピン設定、ペリフェラル設定の流れ

ピン設定は、Configuratorを使用して行うのが簡単です。1 つのピンを複数のペリフェラルで共有し ている場合も多いですので、競合が生じないような設定を、直感的に行うことができます。ソースコード も自動生成されます。

ペリフェラル設定は、リファレンスマニュアルとアプリケーションノート(およびソースコード)を参考に、 行って頂くことになります。

#### 7-6 ピン設定

EFM32LG990を使用し、①USART0 (Asynchronous mode=UART)、②I2C0、③GPIO1 本を使用した場合の設定手順をご紹介します。もちろんゼロからコードを書くこともできますが、Configuratorを使用するのが簡単です。

- 1. Configurator をクリックし、「6-4 ピン設定やペリフェラル設定してみる」と同じ手順でプロジェクト を作成します。
- Configurator が起動したら、Default Mode Peripherals でペリフェラル設定をしていきます。まず① USART0 の設定を行います。USART0 にチェックを入れ、Outline に表示された USART0 を選択 し、Properties ウィンドウにて USART0 の設定を行います。ここでボーレートなどの設定を行うこと ができます。

	🗖 Properties 🔉 🏢 Peripheral Mapping	🛃 🗟 🗸 🗖 🗖
	Properties of USART0	
	USART 0	
	Property	Value
	Mode	
	USART Mode	Asynchronous Mode (UART)
	Asynchronous Settings	
	Baudrate	115200 (0×1C200)
	Databits in frame	8
	Parity bits	No parity
	Stop bits	1 stopbit
main.c 🗱 *myProjecthwconf 🕱	Oversampling	16x
	Majority vote	Enabled
DefaultMode Peripherals	PRS for USART Rx	Disabled
	IrDA modulator settings	
<b></b>	Enable IrDA mode	Disabled
	Invert RX signal before demodulator	Not inverted
	Enable filter before demodulator	No filter
	Pulse width fraction	3 fractions
	Enable PRS channel as input	Input is TX
	PRS triggering settings	
	Enable RX triggering through PRS	Disabled
	Enable TX triggering through PRS	Disabled
	Triggering channel	Channel 0
	Selected channel state	PBS peripheral is not enabled

3. ②I2C0の設定を行います。Default Mode Peripherals でI2C0 にチェックを入れ、Outline に表示さ れた I2C0 を選択し、Properties ウィンドウにて I2C0 の設定を行います。

	🔲 Properties 🗙 🛛 🗰 Peripheral I	Mapping 🛃 🐨 🗖 🗖
	Properties of I2C0	
	12C 0	
DefaultMode Peripherals	Property	Value
	Settings	
	Enable	Enable I2C after init
Communications *	Master/Slave	Master mode
	Low/High ratio	4:4 clock ratio
	I2C max bus frequency	1 MHz (fast mode +) 🗸
		10 kHz (low speed)
		100 kHz (standard)
UARTO UARTI <b>USARTO USARTI USART</b> 2		400 KHz (fast mode)
		T MINZ (Tast mode +)

4. ③GPIO の設定を行います。Default Mode Peripherals で GPIO にチェックを入れます。

DefaultMode Peripherals	
Other	*

 Default Mode Port IO に移動して、①USARTO からピン設定を行っていきます。Outline で Port IO を選択し、Peripheral Mapping ウィンドウで USARTO にチェックし、使用するピン(CLK, CS, RX, TX)にチェックします。すると、Default Mode Port IO にピン配置が反映されます。次図の場合で は、パッケージ左上の4本ピンに USARTO ピンが割り振られました。よく見ると、ピン下の信号名 が赤字で表示されています。これはピン設定がまだ完了していないことを示しています。



6. ピン設定を行っていきます。例として USARTO\_RX ピンをダブルクリックします。Property ウィンド ウにピンの設定画面が表示されますが、Pin mode に の表示があります。これが修正すべき 項目になります。Pin mode を選択して黒く反転させると、画面下に赤字で、どのように設定変更 すれば良いか?のアドバイスが表示されます。

Ĭ	🗖 Properties 🕱 🏾 🎆 Peripheral Mapping	🛃 🖪 🗸 🗖 🖸	
	Properties of PE11		
	Port Pin		
	Property	Value	
	Settings		
	Pin mode	🔞 Disabled	
	Pullup	Disabled	
	Custom pin name		
	Reserve	Not reserved	
	- T		
ect) • USART0_RX (PE11) pin mode must be Inpu	t		

USART0\_RXのPin modeをInput にするように指示されていますので、その通りに設定すると、 Default Mode Port IOの信号名(USART0\_RX)が赤字から黒字に変わりました。同じ要領で、他の3つのピンのPin modeをpush-pullに変更します。



 ②I2C0 のピン設定を行います。Outline で Port IO を選択し、Peripheral Mapping ウィンドウで I2C0 にチェックし、使用するピン(SCL, SDA)にチェックします。Default Mode Port IO のピン下の 信号名を見ると赤字で表示されていますので、まだ設定が完了していません。USART0と同じ要 領で、2 つのピンの Pin mode を Wired-and pullup filter に変更します。



 ③GPIO のピン設定を行います。今回は、A1(PE15)に push pull 出力を割り当ててみます。
 Default Mode Port IO にて、A1(PE15)をダブルクリックします。Property ウィンドウにピンの設定 画面で Pin mode を Push-pull に変更し、Custom pin name に信号名(下図では TEST)を入力しま す。Default Mode Port IO に反映されました。

main: 🔅 *nyProjecthwcont 🛛			🗖 maino 🛛 🗐 👘 🖬	ojecthwconf 🖂	
DefaultMode Port I/O: PORTIO			DefaultMode Por	rt I/O: PORTIO	
A1         A2         A3           PEIS         PEIA         PEIS           PAIS         PEIS         B3           PAIS         PEIS         PEIS           PAIS         PAD         PEIS           PEIS         ECOSOL         PEISTOR	Properties ℜ	ral Mapping R Value Value Push-pull TESTI Not reserved	A1 PEIS (TEST) B1 PAIS DA 200,SOL	A2 PEI 4 B2 PEI 3 USARTO_OS C2 PAO I200.SDA	PE12 USARTO,OLK B3 PE11 USARTO,RX C3 PE10 USARTO,TX

9. 設定が完了したら、Default Mode I/O ウィンドウか、Default Mode Peripheral ウィンドウの上で右 クリックし、Generate Sourceを選択します。ペリフェラル設定、ピン設定を含んだCコードが生成さ れます。

左下図はプロジェクトツリーですが、InitDevice.c に行った設定が反映されています。

Enter\_DefaultMode\_from\_RESET という関数の中で、USART0, I2C0, PortI/O の初期化関数を呼んでいます。それぞれの初期化関数も参考のために掲載します。



362	//
364	
3656	extern void I2C0 enter DefaultMode from RESET(void) {
366	// \$[12(0 initialization]
367	7/ plate init = 12C INIT DEFAULT:
368	Trefrance, Marcon Trefrance, Convert,
369	init_enable = 1:
370	init matter = 1:
371	init free = 12C EPEO STANDARD MAX.
372	init clbr = i2c(lock#RStandord)
373	Tic toit (T2C0 & init):
374	// [T2C0 initialization]
375	// [red interarticition]#
376	
377	3
1	*
540	extern void PORTIO_enter_DefaultMode_from_RESET(void) {
542	// \$[Port A Configuration]
543	
544	/* Pin PA0 is configured to Open-drain with pull-up and filter */
545	<pre>GPIO-&gt;P[0].MODEL = (GPIO-&gt;P[0].MODEL &amp; ~_GPIO_P_MODEL_MODE0_MASK)   GPIO_P_MODEL_MODE0_WIREDANDPULLUPFILTER;</pre>
546	
547	/* Pin PA1 is configured to Open-drain with pull-up and filter */
548	GPIO->P[0].MODEL = (GPIO->P[0].MODEL & ~_GPIO_P_MODEL_MODE1_MASK)   GPIO_P_MODEL_MODE1_WIREDANDPULLUPFILTER;
549	// [Port A Configuration]\$
550	
551	// «[Dart B Configuration]
552	// #prot 8 configuration]
554	// [For the contrigutation]
555	
556	// \$[Port C Configuration]
557	// [Port C Configuration]\$
558	
559	
560	// \$[Port D Configuration]
561	// [Port D Configuration]\$
562	
563	
564	// \$[Port E Configuration]
565	/* Die DE10 is serfiement to Duck sull */
566	/* Pin Pelo is configured to Push-puil */
568	drio-yr[4].hoben = (drio-yr[4].hoben & ~_drio_r_hoben_hobers_hask)   drio_r_hoben_hobers_hoshole;
569	/* Pin PE11 is configured to Input enabled */
570	GPIO->P[4].MODEH = (GPIO->P[4].MODEH & ~ GPIO P MODEH MODE11 MASK)   GPIO P MODEH MODE11 INPUT;
571	
572	/* Pin PE12 is configured to Push-pull */
573	<pre>GPIO-&gt;P[4].MODEH = (GPIO-&gt;P[4].MODEH &amp; ~_GPIO_P_MODEH_MODE12_MASK)   GPIO_P_MODEH_MODE12_PUSHPULL;</pre>
574	
5/5	/ FIN FLD IS CONTIGUED TO FUSH-PULL -/
570	deio-selel'unneu = (deio-selel'unneu « ~daio-a-lunneu-lunneis-lunse)   daio-a-lunneu-lunneis-hopen-generations
578	/* Pin PE15 is configured to Push-null */
579	GPIO->P[4],MODEH = (GPIO->P[4],MODEH & ~ GPIO P MODEH MODE15 MASK)   GPIO P MODEH MODE15 PUSHPULL:
580	// [Port E Configuration]\$
581	
582	
583	// \$[Port F Configuration]
584	// [Port F Configuration]\$
585	
586	
587	// \$[Route Configuration]
588	(* Fachla simple CCL CDA */
509	7' Endle Signals SCL, SDA ''
501	1200-70001E 14 120_0001E_SCHEM   120_0001E_SDAFEN;
502	/* Module PCNT0 is configured to location 1 */
503	$P(NTA_{2}) = P(NTA_{2}) = P(N$
594	For a substant of the substant
595	/* Enable signals CLK, CS, RX, TX */
596	USART0->ROUTE  = USART ROUTE CLKPEN   USART ROUTE CSPEN   USART ROUTE RXPEN
597	USART ROUTE TXPEN;
598	// [Route Configuration]\$
599	
600	
601	}

Configurator が生成されたコードをベースに、リファレンスマニュアルとアプリケーションノート(および サンプルコード)を参考に、必要に応じて追記・修正を行ってください。

### TecStar -

#### 7-7 ペリフェラル設定

ペリフェラル設定の流れをご紹介します。

#### 7-7-1 USART (Asynchronous mode)

◆ 参考にするソースコード

Application Notes ⇒ AN0045 で入手できます。Import Project をすると Simplicity IDE からソースコ ードが開けますので、ソースコードを追うのが楽になります。

AN0040 Debug and Tu AN0045 USART or UA AN0045 USB Hardware AN0047 Interfacing Gra	RT Asynchronous mode Decige Quide anhieal Displays			×
This application note des asynchronous mode. An included software exa how to implement interru transceiver.	cribes how to configu mple for the EFM32GG pt driven receive and	re the EFM32 UAF G-DK3750 Giant G transmit, utilizin	RT or USART to o ecko Developm g the on-board I	perate in ent Kit hows RS-232
Filter by selected produc	t line			
?	Import Project	Open Folder	Open	Close

#### ◆ ヘッダファイル

"em\_gpio.h" と"em\_usart.h"を include してください。

```
◆ 初期化
```

```
以下は、初期化の例です。
       void usart1_init(void) 任意の関数名
       ł
                  USART_InitAsync_TypeDef uart_init = {
usartEnable, // Enable RX/TX when init completed
                             0,
115200.
                                                                       // Use current configured reference clock for configuring baudrate
                                                                       // 115200 bits/s
// 16x oversampling
                             usartOVS16,
                             usartDatabits8, // 8 databits
                                                                  // No parity
                             usartNoParity,
                             usartStopbits1, // 1 stopbit
                                                                    // Do not disable majority vote
// Not USART PRS input mode
// PRS channel 0
                              false,
                             false,
                             usartPrsRxCh0
usartPTsKxCn0 // PKS channel 0
};
// Using "TX" and "RX"
GPIO_PinModeSet(gpioPortD, 7, gpioModePushPull, 1);
GPIO_PinModeSet(gpioPortD, 6, gpioModeInput, 0);
// Configure USART for basic async operation
uart_init.enable = usartDisable;
USART_InitAsync(USART1, & uart_init);
// Prepare UART Rx and Tx interrupts
USART_IntClear(USART1, USART_IFC_MASK);
USART_IntClear(USART1, USART1, EX_IRQn);
NVIC_ClearPendingIRQ(USART1_TX_IRQn);
NVIC_EnableIRQ(USART1_TX_IRQn);
NVIC_EnableIRQ(USART1_TX_IRQn);
// Enable pins at default location
USART_ROUTE_LOCATION_LOC3;
// Finally enable it
USART_Enable(USART1, usartEnable);
}
```

割り込み処理 ٠ 割り込みハンドラは"system\_efm32lg.h"(EFM32LG の場合)でプロトタイプ宣言されています。 ¥¥SiliconLabs¥SimplicityStudio¥v2¥developer¥sdks¥efm32¥v2¥Device¥SiliconLabs¥EFM32LG¥Include 以下は、割り込み処理の例です。 void USART1\_RX\_IRQHandler(void) UART1 RX の割り込みハンドラ { uint8\_t rxdata; // Checking that RX-flag is set if (USART1->STATUS & USART\_STATUS\_RXDATAV) { rxdata = USART1->RXDATA; // 独自の処理 // Disable Rx interrupt USART1->IEN &= ~USART\_IEN\_RXDATAV; NVIC\_ClearPendingIRQ(USART1\_RX\_IRQn); NVIC\_DisableIRQ(USART1\_RX\_IRQn); } } UART1 TX の割り込みハンドラ void USART1\_TX\_IRQHandler(void) { // Checking that the USART is waiting for data if (USART1->STATUS & USART\_STATUS\_TXBL) { // Transmitting the next byte USART1->TXDATA = "user data" // 独自の処理 // Disable Tx interrupt USART1->IEN &= ~USART\_IEN\_TXBL; NVIC\_DisableIRQ(USART1\_TX\_IRQn); NVIC\_ClearPendingIRQ(USART1\_TX\_IRQn); } }

#### ◆ 制御方法

"em\_usart.h"の API を使用して制御します。7-2「開発用のソースコードについて」をご参照ください。

Main Page	Modules	Data Structures	Files	Documentation Home	sila 🔍 Search
► RMU	(				Data Structures   Macros   Enumerations   Functions
► RTC					
► SYSTEM					
► TIMER		Universal Synchrono	ous/Asynchron	ous Receiver/Transmitter (USART) pe	eripheral API
USART	)				
▶ VCMP		. More			
► VERSION		Data Character			
▶ WDOG		Data Structur	res		
EM_Drivers		struct USART_Ini	itAsync_TypeD sTriggorIpit_T	ef meDef	
► USB		struct USART_Ini	itSync_TypeDe	f	
► BSP		struct USART_Ini	itlrDA_TypeDe	f	
Drivers		struct USART_Ini	itl2s_TypeDef		
Data Structures		Macros			
Files		#define USART_II	NITASYNC_DE	AULT	
Documentation Ho	me	✓ #define USART II	NITPRSTRIGGE	R DEFAULT	

#### 7-7-2 I2C

◆ 参考にするソースコード

Application Notes ⇒ AN0011 で入手できます。Import Project をすると Simplicity IDE からソースコ ードが開けますので、ソースコードを追うのが楽になります。

AN0000 Certics Starts of with FEMS AN0011 I2C Master and Slave Ope	ration			×
The EFM32 12C module allows simpl circuits using only one data and one EFM32 12C module in multimaster m slave and master mode.	e, robust and cost effe clock line. This appli ode. Two EFM32s are (	ctive communica cation note demo connected; each E	tion between inte nstrates how to u FM32 will operat	grated size the te in both
Filter by selected product line				
•	Import Project	Open Folder	Open	Close

#### ◆ ヘッダファイル

"em\_gpio.h" と"em\_i2c.h"を include してください。

◆ 初期化

```
以下は、初期化の例です。
                                         任意の関数名
void i2c0_init(void)
ł
        // default settings is 100KHz
        I2C_Init_TypeDef i2cInit = {
                                                            // Enable when init done
                true,
                true, // Enable when fint done
true, // Set to master mode
0, // Use currently configured reference clock
I2C_FREQ_STANDARD_MAX, // Set to standard rate assuring being within I2C spec
i2cClockHLRStandard, // Set to use 4:4 low/high duty cycle
        };
        // Using "SDA" and "SCL" pin
GPIO_PinModeSet(gpioPortE, 12, gpioModeWiredAndPullUpFilter, 1);
GPIO_PinModeSet(gpioPortE, 13, gpioModeWiredAndPullUpFilter, 1);
        // Initializing the I2C
I2C_Init(I2C0, &i2cInit);
        // Clear and enable interrupt
NVIC_ClearPendingIRQ(12C0_IRQn);
NVIC_EnableIRQ(12C0_IRQn);
        I2C_Enable(I2C0, 1);
}
         割り込み処理
```

割り込みハンドラは"system\_efm32lg.h" (EFM32LG の場合)でプロトタイプ宣言されています。

以下は、割り込み処理の例です。

```
void I2C0_IRQHandler (void) 割り込みハンドラ
```

```
// 独自の処理
```

NVIC\_DisableIRQ(I2C0\_IRQn);

{

}

### 制御方法 "em\_i2c.h"の APIを使用して制御します。7-2「開発用のソースコードについて」をご参照ください。 以下は、I2C ライト、I2C リードの例になります。 int i2c\_write(I2C\_TypeDef \*i2c, uint8\_t addr, uint8\_t wval) I2C ライトの制御例 { uint8\_t wdata[1]; I2C\_TransferSeq\_TypeDef seq; I2C\_TransferReturn\_TypeDef I2C\_Status; I2C\_TransferRefurn\_TypeDef I2C\_Status; seq.addr = addr; seq.flags = I2C\_FLAG\_WRITE\_WRITE; // write buffer setting wdata[0] = (uint8\_t) wval; seq.buf[0].data = wdata; seq.buf[0].len = 1; // Do a polled transfer I2C\_Status = I2C\_TransferInit(i2c, &seq); while (I2C\_Status == i2cTransferInProgress) { I2C\_Status = I2C\_Transfer(i2c); } return(I2C\_Status); } int i2c\_read(I2C\_TypeDef \*i2c, uint8\_t addr, uint8\_t \*p\_rval) I2C リードの制御例 uint8\_t rdata[1] = {0}; I2C\_TransferSeq\_TypeDef seq; I2C\_TransferReturn\_TypeDef I2C\_Status; seq.addr = addr; seq.flags = I2C\_FLAG\_READ; // register setting seq.buf[0].data = rdata; seq.buf[0].len = 1; // Do a polled transfer I2C\_Status = I2C\_TransferInit(i2c, &seq); while (I2C\_Status == i2cTransferInProgress) { I2C\_Status = I2C\_Transfer(i2c); $p_rval = rdata[0];$

return (I2C\_Status);

7
,

Main Page	Modules	Data Structures	Files	Documentation Home	sila 🔍 Search
▶ EBI	6				Data Structures   Macros   Enumerations   Functions
▶ EMU		I2C			
► GPIO		EM_LIDrary			
► 12C		Inter-integrated Cir	rcuit (I2C) Periph	eral API	
▶ INT					
▶ LCD		. More			
▶ LESENSE					
▶ LETIMER		Data Structu	ires		
▶ LEUART		struct I2C_Init_1	TypeDef		
▶ MPU		struct I2C_Trans Master m	sferSeq_TypeDe lode transfer me	ef essage structure used to define a com	plete I2C transfer sequence (from start to
▶ MSC		stop). Mo	re		
▶ OPAMP					
▶ PCNT		Macros			
▶ PRS		#define I2C_FRE Standar	<b>Q_STANDARD_I</b> d mode max fre	MAX 92000 quency assuming using 4:4 ratio for N	llow:Nhigh. More
▶ RMU		▼I #define I2C FRF		92157	

#### 7-7-3 タイマ

◆ 参考にするソースコード

Application Notes ⇒ AN0014 で入手できます。Import Project をすると Simplicity IDE からソースコ ードが開けますので、ソースコードを追うのが楽になります。

AN0013 Direct Ma AN0014 TIMER AN0016 Watchdog AN0016 Oscillator	mory Access	×
This application not explanations on how input capture, outpu	e gives an overview of the EFM32 TIMER module, followed by v to configure and use its primary functions, including up/down it compare and PWM.	count,
Filter by selected p	roduct line	
?	Import Project Open Folder Open	Close

#### ◆ ヘッダファイル

"em\_timer.h"を include してください。

◆ 初期化

以下は、初期化の例です。

<pre>static void start_timer1(void) 任意の関数名 {     // Enable clock for TIMER1 module     CMU_ClockEnable(cmuClock_TIMER1, true); 初期設定     // Select TIMER1 parameters     TIMER_Init_TypeDef timerInit = TIMER_INIT_DEFAULT;    // Enable overflow interrupt     TIMER_IntEnable(TIMER1, TIMER_IF_OF);     // Enable TIMER1 interrupt vector in NVIC     NVIC_EnableIRQ(TIMER1_IRQn);     // Set TIMER1 Top value     TIMER_TopSet(TIMER1, CMU_ClockFreqGet(cmuClock_HFPER) /     // Configure TIMER1     TIMER1, &amp;timerInit); }</pre>	<pre>{     enable = true,     debugRun = false,     prescale = timerPrescale1024,     clkSel = timerClkSelHFPerClk,     fallAction = timerInputActionNone,     riseAction = timerInputActionNone,     mode = timerModeUp,     dmaClrAct = false,     oneShot = false,     sync = false,     }; 1000); </pre>
<ul> <li>割り込み処理</li> <li>割り込みハンドラは"system_efm32lg.h"(EFM32LG の場合)て</li> <li>以下は、割り込み処理の例です。</li> <li>void TIMER1_IRQHandler(void) 割り込みハンドラ         <ul> <li>( 独自の処理</li> </ul> </li> </ul>	ミプロトタイプ宣言されています。

// Clear flag for TIMER1 overflow interrupt TIMER\_IntClear(TIMER1, TIMER\_IF\_OF);

```
}
```

#### ◆ 制御方法

"em\_timer.h"の API を使用して制御します。7-2「開発用のソースコードについて」をご参照ください。

Main Page	Modules	Data Structures	Files	Documentation Home	sila 🔍 Search
▶ LEUART	(				Data Structures   Macros   Enumerations   Functions
MPU					
► MSC					
► OPAMP		Timer/Counter (TIME	R) Peripheral /	API. More	
▶ PCNT					
PRS		Data Structure	es		
▶ RMU		struct TIMER_Init	_TypeDef		
► RTC		struct TIMER_Init	CC_TypeDef		
► SYSTEM		struct TIMER_Init	DTI_TypeDef		
TIMER		Macros			
▶ TIMER_Init	t_TypeDef	#define TIMER_IN	IT_DEFAULT		
TIMER_Init	tCC_TypeDef	#define TIMER_IN	ITCC_DEFAUL	т	
► TIMER_Init	DTI_TypeDef	#define TIMER_IN	ITDTI_DEFAUI	LT	
TIMER_INI	T_DEFAULT	Enumerations	5		
TIMER INI	TCC DEFAULT	enum TIMER CCM	Mode TypeDe	f{	

#### 7-7-4 タイマ (X 秒タイマの作り方)

**TecStar** 

TIMER\_Init\_TypeDef 型のメンバーである".prescale"と".clkSel"と、"TIMER\_TopSet()"の値を変えて 任意の時間のタイマを作ります。

AN0014 の"main\_timer\_up\_count.c"では2秒タイマを作っており、これを例に説明します。

タイマへ供給されるクロックは、下図のような構成になっています。(AN0014 Fig.2.1)



"main\_timer\_up\_count.c"では、".prescale"と".clkSel"は以下の設定になっています。

TIMER\_Init\_TypeDef timerInit =



つまりタイマへの供給クロックは、14 MHz(周波数) ÷ 1024(prescale) = 13671 Hz となります。 2 秒タイマを作るには、13671 Hz × 2 = 27342 のカウントが必要です。"TIMER\_TopSet()"を使 って、タイマの上限値を 27342 に設定します。



#### 7-7-5 CMU (ペリフェラル・クロックの周波数)

TecStar =

ペリフェラルへの供給クロックの周波数を、どのように設定するか紹介します。

リファレンスマニュアルのCMU - Clock Management Unitの章に、CMU Overview という図があります。 この図で、EFM32 のクロックツリーを確認する事ができます。下図は CMU Overview の一部抜粋になり ますが、元になるクロックソース(下図の青枠)が、セレクタや分周回路(下図の緑枠)を経て、ペリフェラ ル・クロック(下図の赤枠)になります。



シンプルな図にすると下図になります。色づけした箇所が、周波数に影響するブロックです。



クロックソースとして、HFRCO(内蔵の高速クロック)を選択した場合を例に説明します。

◆ HFRCOの周波数設定(クロックソース)

HFRCOの周波数は、CMU\_HFRCOCTRLのBANDビットで設定されています。

下図は Leopard Gecko のリファレンスマニュアルからの抜粋ですが、1 (実際には 1.2),7 (実際には 6.6), 11, 14, 21, 28MHz から選択ができます。デフォルトでは 14MHz に設定されています。

11.5.4 CMU\_HFRCOCTRL - HFRCO Control Register

11MHZ

14MHZ

21MHZ

28MHZ

Offset															В	t Po	siti	on														
0x00C	3	30	29	28	27	26	25	24	53	8	21	20	19	9	17	16	15	4	13	12	÷	10	თ	œ	~	- 9	2	4	e	~	-	•
Reset																		0×00					0X3	)	0×80							
Access																		RW					RV						RV			
Name																		SUDELAY					BAND						TUNING			
Bit								Re								De		iptio														
10:8	BAI	١D						0x3	5			R	W			HF	RCC	Ba	nd	Sele	ect											
	Wri the frec diffe	te th HFF Juen erent	is fie RCO cy, t t bar	ld to out he l ids o	o set put, HFTI can b	then hen UNI	e fre ice i NG ead	quer t is s valu from	afe t afe t e sh n the	oand to cl ioul De	l in w hang d als vice	hich e thi o be nfor	n the is se wr ma	e HF ettin ritter tion	RC( g ev n wh page	Dist en w en c e.	to op hile hang	erat the s ging	e. V sys the	Whe tem fre	is ru ig ru quei	angi innir ncy l	ng t ng or band	his s n the I. Th	etti Hi ne o	ting th FRC calibi	nere O. T rateo	will oer dtur	be n nsure ning	o gli an valu	tche accu e for	s on irate r the
	Val	ue			M	lode								[	Descr	iptior	1															
	0				1	MHZ	Z							1	MH:	z ban	d. No	DTE:	Als	o set	t the	TUNI	NG	/alue	(bit	its 7:0	) whe	en cl	nangir	ng ba	nd.	
	1				7	MHZ	2							7	MH:	z ban	d. NO	DTE:	Als	o set	t the	TUNI	NG	alue	(bit	its 7:0	) whe	en cl	nangir	ng ba	nd.	



11 MHz band, NOTE: Also set the TUNING value (bits 7:0) when changing band.

14 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band

21 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.

28 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band

◆ クロックソースの選択 (セレクタ)



クロックソースは、CMU\_CMD の HFCLKSEL で設定します。

11.5.10 CMU\_CMD - Command Register

Offset																t Po																
0x024	31	30	29	28	27	26	<u>ې</u>	24	23	22	21	20	19	9	17	16	15	4	<del>2</del>	12	=	10	ი	8	~	6	2	4	e	~	-	0
Reset																											0X0	0	0		8	
Access																											۲1	W1	W1		W1	
Name																											USBCCLKSEL	CALSTOP	CALSTART		HFCLKSEL	
2:0	HF	CLK	SEL					0x0				V	/1			HF	CLK	Sel	ect													
	Selects the clock source for HFCLK. Note that selecting an oscillator that is disabled will cause the system clock to stop. Check the status register and confirm that oscillator is ready before switching.																															
	Va	lue			M	ode								D	escr	iption																
	1				н	FRCC								S	elec	t HFR	CO a	as H	FCLI	К.												
	2				Н	FXO								S	elec	t HFX	O as	HFO	CLK.													
							_	_		_	_	_	_		_		_	_		_												_

Select LFXO as HFCLK

◆ HFCLK の決定 (分周 1)

LFXO

4



クロックソース(HFRCO)を分周して HFCLK にします。何分の 1 にするかは、CMU\_CTRL の HFCLKDIV で決定します。

HFCLK は、クロックソース(HFRCO)の 1/(HFCLKDIV+1) なります。1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8 から選択できます。HFCLKDIV = 0であればHFCLK = HFRCO、HFCLKDIV = 7であればHFCLK = HFRCO \* (1/8) になります。

この HFCLK は、CPU コアの動作周波数にも利用します。

Offset															Bi	t Po	ositi	on													
0x000	3	8	29	28	27	26	25	24	23	3	5	20	19	18	17	16	15	4	<del>1</del> 3	5	÷	9	ი	 7	9	2	4	e	~	-	0
Reset		0		0				8			0×0			0x3			0×0		-	1 0x0		0×3		0	0×1			2	200	2	2
Access		RV		RW			RW			RW			Md		RW		RW		ΝŇ	N N		RV		RW	RV			RW		RW	
Name		HFLE		DBGCLK				CLKOUTSEL1			<b>CLKOUTSEL0</b>		I EVOTIMEOUT		LFXOBUFCUR	HFCLKDIV		HFCLKDIV	LFX0B00ST	LUCOTOLL			HFAUIIMEOUI	HFXOGLITCHDETEN		HFXOBUFCUK		TOCOCOL	HFAUBOUSI	HEVONODE	
16:14	HFCLKDIV 0x0 RW HFCLK Division																														
	Us	Use to divide HFCLK frequency by (HFCLKDIV + 1).																													

#### 11.5.1 CMU\_CTRL - CMU Control Register

◆ HFPERCLK の決定 (分周 2)



HFCLK を分周して、HFPERCLK にします。何分の1にするかは、CMU\_HFPERCLKDIVの HFPERCLVDIV ビットで決定します。

分周は、1/1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256, 1/512 から選択できます。1/512 を選択した 場合には、HFPERCLK = HFCLK \* (1/512) になります。

#### 11.5.3 CMU\_HFPERCLKDIV - High Frequency Peripheral Clock Division Register

						_	_																									
Offset		Bit Position																														
0x008	31	30	29	28	27	z z z z z z z z z z z z z z z z z z z										e	8	-	0													
Reset																							-					0×0				
Access																									RW							
Name																								HFPERCLKEN							HFFERCLAUIV	
3:0	HFPERCLKDIV 0x0 RW						W	HFPERCLK Divider																								
	Specifies the clock divider for the HFPERCLK.																															
	Value Mode								Description																							
	0				н	FCL	ĸ							-	IFPE	RCL	K = H	IFCL	К.													
	1 HFCLK2								-	IFPE	RCL	K = H	IFCL	.K/2	2.																	
	2 HFCLK4								HFPERCLK = HFCLK/4.																							
	3 HFCLK8							HFPERCLK = HFCLK/8.																								
	4 HFCLK16							HFPERCLK = HFCLK/16.																								
	5 HFCLK32							HFPERCLK = HFCLK/32.																								
	6 HFCLK64							HFPERCLK = HFCLK/64.																								
	7 HFCLK128							HFPERCLK = HFCLK/128.																								
	8				н	FCL	K25	6						ŀ	IFPE	RCL	K = H	HFCL	.K/2	256.												
	9 HFCLK512							ŀ	IFPE	RCL	K = H	IFCL	.K/5	512.																		

◆ まとめ

TecStar =

以上をまとめますと、ペリフェラルに供給されるクロックの周波数は、下記の組み合わせになります。



HFRCO を分周し、通常ペリフェラルで使用した場合の事例を紹介しましたが、HFRCO ではなく HFXO(外付け高速クロック)/LFRCO(内蔵の低速クロック)/LFXO(外付け低速クロック)を使用した場 合や、通常ペリフェラルではなく Low Energy ペリフェラルを使用した場合にも、CMU Overview のクロッ クツリーから読み解いていくという考え方は共通です。

#### 改版履歴

Version	改定日	改定内容
1.0	2014年07月	•新規作成
1.13	2015年06月	・マクニカオンラインで公開
1.14	2015年08月	・誤植訂正(6-2章 M0+の breakpoint数)、4-1, 6-3, 6-6-4,
		6-7-3 章追記
1.15	2016年02月	・Pearl Gecko/Jade Gecko 追加。最新の Simplicity Studio
		に合わせて説明を一部変更。

#### 参考文献

- Silicon Labs 社 各種ドキュメント
- Silicon Labs 社 ナレッジベース、コミュニティフォーラム

#### 免責、及び、ご利用上の注意

弊社より資料を入手されましたお客様におかれましては、下記の使用上の注意を 一読いただいた上でご使用ください。

- 1. 本資料は非売品です。許可無く転売することや無断複製することを禁じます。
- 2. 本資料は予告なく変更することがあります。
- 3. 本資料の作成には万全を期していますが、万一ご不審な点や誤り、記載漏れなどお 気づきの点がありましたら、弊社までご一報いただければ幸いです。
- 4. 本資料で取り扱っている回路、技術、プログラムに関して運用した結果の影響につい ては、責任を負いかねますのであらかじめご了承ください。
- 5. 本資料は製品を利用する際の補助的なものとしてかかれたものです。製品をご使用 になる場合は、メーカーリリースの資料もあわせてご利用ください。

〒222-8561 横浜市港北区新横浜 1-6-3 TEL 045-470-9841 FAX 045-470-9844

本社