

# AN201701 STM32CubeMX と µC3/Compact の共存方法

Rev. 1. 0

September 29, 2017

# 改訂記録

		改訂内容				
Rev.	発行日	ページ	内容			
1.0	2017.09.29	_	初版			

# 目次

改訂記録	2
目次	3
1.   概要	4
2. STM32CubeMX と µC3/Compact 共存プロジェクト構築方法	6
3. サンプルプログラム(ボタン押下→LED 点滅)作成手順	8
3.1. フォルダ構成	8
<b>3.2.</b> STM32CubeMX を用いたプロジェクト作成	9
3.2.1. 新規プロジェクトの作成	9
3.2.2. Pinout 設定	10
<b>3.2.3.</b> クロック設定	13
3.2.4. GPIO 設定	14
3.2.5. 割り込み設定	17
3.2.6. ソースコード生成	18
3.3. µC3/Compact の設定	19
<b>3.4. EWARM</b> のプロジェクト設定	26
3.5. 自動生成コードの修正	35
3.6. 動作確認環境	41
4. サンプルプログラム(USB-HID)作成手順	42
<b>4.1.</b> フォルダ構成	42
<b>4.2.</b> STM32CubeMX を用いたプロジェクト作成	42
4.2.1. 新規プロジェクトの作成	42
4.2.2. Pinout 設定	42
<b>4.2.3</b> . クロック設定	45
4.2.4. GPIO 設定	46
4.2.5. USB 設定	47
4.2.6. 割り込み設定	49
<b>4.2.7</b> . ソースコード生成	49
<b>4.3</b> . μC3/Compact の設定	50
<b>4.4. EWARM</b> のプロジェクト設定	51
4.5. 自動生成コードの修正	53
4.6. 動作確認環境	57

#### 1. 概要

本ドキュメントは STM32CubeMX と µC3/Compact の共存方法を示すものです。ST マイク ロエレクトロニクスの提供する STM32CubeMX は、ST マイクロエレクトロニクス製マイ コンのデバイス周辺機能の設定を簡単に行うことができるツールです。また µC3/Compact を用いると RTOS の機能を各種マイコンに実装することができます。

STM32CubeMX と µC3/Compact を共存させることにより以下のメリットが得られます。

- µITRON 仕様のシステムコールが使えるようになります。これによりタスク管理、時間 管理、周期ハンドラ等の機能が実装され、CPU 負荷低減、間欠的な処理の作成、処理 の並列化等が行えるようになります。
- コンフィグレータを用いることで複雑なリアルタイム OS の機能をグラフィカルかつ 容易に設定することができます。

本ドキュメントではまず、STM32CubeMX が生成したファイルと µC3/Compact のコンフ ィグレータが生成したファイルからプロジェクトを構築する方法を示します。その後、具 体的な各サンプルプログラムの作成手順を説明します。



STM32CubeMX と μ C3/Compact の共存方法 1. 概要

#### 使用した動作確認環境

使用した動作確認環境を以下に示します。

#### 表1.1 使用した動作確認環境

項目	バージョン
μC3 パッケージ	μC3/Compact Cortex-M4 STM32F4 シリーズ EWARM 版 Release 2.1.9
STM32CubeMX	Version 4.21.0
コンパイラ	IAR Embedded Workbench for ARM 8.1.0.12863
評価ボード	STM32F4 Discovery kit for STM32F469 MCU

#### 作成するサンプルプログラム

本ドキュメントで作成するサンプルプログラムを以下に示します。

表1.2 作成するサンプルプログラム

章	内容
2章	LD1 が消灯時に USER ボタンを押すと点灯し、点灯時にボタンを押すと消灯する
	プログラム。
3章	USER ボタンを押すと PC 上のマウスポインタが移動するプログラム。
	※サンプルの一部に、ST マイクロエレクトロニクスの提供するソフトウェア
	STM32CubeF4のUSB Deviceのサンプルを使用しました。
	ダウンロードURL: <u>http://www.st.com/ja/embedded-software/stm32cubef4.html</u>
	参照元:
	STM32Cube_FW_F4_V1.16.0¥Projects¥STM32469I-Discovery¥Applications¥US
	B_Device¥HID_Standalone

TRON は"The Real-time Operation system Nucleus"の略称です。

 $\mu$  ITRON は"Micro Industrial TRON"の略称です。

μC3 はイー・フォース株式会社の登録商標です。

STM32は、STマイクロエレクトロニクスの登録商標です。

その他、本資料中の製品名やサービス名は全てそれぞれの所有者に属する商標または登録 商標です。

本書で記載されている内容は予告無く変更する場合があります。

STM32CubeMX の設定の詳細については STM32CubeMX のマニュアル等をご確認ください。

## 2. STM32CubeMX と µ C3/Compact 共存プロジェクト構築方法

STM32CubeMX と $\mu$  C3/Compact を共存させたプロジェクトの構築フローは次のようになります。

- ① STM32CubeMX を用いてデバイスのハードウェアに関する設定を行いスケルトンコー ドを生成。
- ② μ C3/Compactのコンフィグレータを用いて RTOS に関する設定を行いスケルトンコー ドを生成。
- ③ 統合開発環境のプロジェクトを作成し、コンパイル、リンク、デバッグに関するオプション設定を行う。
- ④ STM32CubeMX 生成ソースコードと µ C3/Compact 生成ソースコードの競合部分の調整のためソースコードの修正を行う。



図 2.1 STM32CubeMX と µ C3/Compact を共存させたプロジェクトの構築フロー



①において STM32CubeMX 上で割り込みサービスルーチンの設定を行う際の注意点として Preemption Priority の 0 (最も高い優先レベル) は、 $\mu$  C3/Compact のカーネルが占有する ことになるため、その他の割り込みに対しては Preemption Priority を 1 より大きい値とす る必要があります。

④でユーザーが自動生成されたコードを、エディタで修正する箇所は

- ・スタートアップルーチン( $\mu$  C3 が生成する prst.s79)
- ・割り込みサービスルーチンとタスク(µC3 が生成する main.c)
- ・割り込み優先度初期化処理(µC3 が生成する hw\_init.c)
   等です。



## 3. サンプルプログラム(ボタン押下→LED 点滅)作成手順

この章では、「LD1 が消灯時に USER ボタンを押すと点灯し、点灯時にボタンを押すと消灯 する」プログラムの作成手順を説明します。

#### 3.1. フォルダ構成

フォルダ構成を下記に示します。



次節から具体的な手順を示していきます。



#### 3.2. STM32CubeMX を用いたプロジェクト作成

#### 3.2.1. 新規プロジェクトの作成

STM32CubeMX を起動して、New Project を選択します。

Board Selector タブを選択し、

Type of Board->Discovery

MCU Series->STM32F4

#### Reference->STM32F469I-DISCO

を選択し、ダブルクリックします。

New Project ×							
VICU Selector Board Seli	ector						
Board Filter							
Verdor :		Tuner	of Bo	and: MCI	I Series :		
STMicmelectronice	/	Diem	ven		/B2E4		
0110000000101100		0.000	001 y	011			
Initialize all peripheral	s wit	h thei	r def	ault Mode			$\rightarrow$
Peripheral Selection				Boards List: 9 Items	;		
Peripherals	Nb	Max		Туре	Reference	MCU	
Accelerometer				Discoverv	STMB2F401C-DISCO	STM82F401 VCTx	
Analog I/O	0	0		Discoverv	STM32F411E-DISCO	STM32F411VETx	
Arduino Form Factor	0	32		Discoverv	STM32F4DISCOVERY	STM82F407VGTx	
Audio Line In	0	0	1	Discovery	STM32F4DISCOVERY	STM82F407VGTx	
Audio Line Out	0	1	1	Discovery	STM32F429I-DISCO	STM32F429ZITx	
Button	0	1		Discovery	STM32F429I-DISC1	STM32F429ZITx	
CAN	0	0		Discovery	STM32F469I-DISCO	STM32F469NIHx	
🕘 Camera				Discovery	STM32F412G-DISCO	STMB2F412ZGTx	
Compass				Discovery	STM32F413H-DISCO	STM32F413ZHTx	
Custom Form Factor	0	66					
Digital I/O	0	128					
Eeprom							
Ethernet							
Flash Memory		0					
Fredom Form Factor	0	0					
Gyroscope							
IrDA							
Joystick							
Lod Display (Graphi							
Lod Display (Segment)							
	0	4	$\sim$				



#### 3.2.2. Pinout 設定

Pinout タブを選択し以下のように設定します。

## USER ボタン押下を検知する割り込みの設定

PA0/WKUP を GPIO\_EXTIO に設定します。





## LD1 を点灯させるための設定

PG6の設定が GPIO\_Output となっていることを確認します。





#### RTOS を動かすための設定

左ペインの Configuration->Peripherals->SYS の設定で TimeBase Source を TIM1 と設定 します。

Per iphe	rals
🗎 🖶 🗛	ADC 1
🖻 🔔 A	ADC2
🖶 🔥 A	VDC3
🖻 🙆 C	SAN1
🗄 👰 C	SAN2
🗄 👰 C	RC
🕀 👰 🖸	)AC
🖻 😣 🛛	)CMI
🖻 🔍 🖸	)MA2D
🖻 – 💓 D	SIHOST
🕀 🖞 E	TH
🕀 🕀 F	MC
	201
	202
	203
	282
шт. 🐨 Т. ст. 🔞 Т	233 WDC
i in <u>·</u> · · · · · · · · · · · · · · · · · ·	RCG
	RNG
	RTC
🖶 🔥 S	SAI1
🖶 🚠 s	DIO
🖶 💿 S	SPI1
🖶 💿 S	SPI2
😐 😣 S	SPI3
🖶 🙆 S	SP14
🖶 📥 S	SPI5
🖶 庄 S	SPI6
🖻 📥 S	SYS
D	lebug Disable
	System Wake-Up
T	imebase Source [TIM1 🔹



## 3.2.3. クロック設定

Clock Configuration タブを選択し以下のように設定します。

#### 表 3.1 クロック設定

項目	設定
System CLK Mux	PLLCLK
PLL source	HSI
PLL N	336
PLL P	4
APB1 Prescaler	2

※STM32CubeMX上で背景赤色表示されている項目は、設定変更が必要な項目です。





## 3.2.4. GPIO 設定

次に、Configuration タブの GPIO ボタンを選択します。



PA0/WKUP と PG6 の設定を確認します(PG6 は GPIO mode と Maximaum output speed を 既定値から変更しています)。





#### 表 3.2 PA0/WKUP 設定

項目	設定
GPIO mode	External Interrupt Mode with Rising edge trigger detection
GPIO Pull-up/Pull-down	No pull-up and no pull-down

Pin Configuration ×							
GPIO Single	Mapped Signal	s					
Search Signa Search (Ci	Search Signals       Search (Crtl+F)     Show only Modified Pins						
Pin Name	Signal on Pin	GPIO outp	GPIO mode	GPIO Pull	Maximum o	User Label	Modified
PA0/WKUP	n/a	n∕a	External Int	No pull-up a	n∕a		$\checkmark$
PA3	n/a	Low	Output Pus	No pull-up a	Low	LCD BL CT	~
PB0	n/a	Low	Output Pus	No pull-up a	Low	EXT_RESET	~
PB2/BOOT1	n/a	Low	Output Pus	No pull-up a	Low	OTG_FS1_Po	~
PB7	n/a	n/a	External Int	No pull-up a	n/a	OTG_FS1_O	~
PD4	n/a	Low	Output Ope	No pull-up a	Low	LED2 [Oran	~
PD5	n/a	Low	Output Ope	No pull-up a	Low	LED3 [Red]	$\checkmark$
PE2	n/a	Low	Output Pus	No pull-up a	Low	AUDIO_RST	$\checkmark$
PE3	n/a	Low	Output Pus	No pull-up a	Low	SPKR/HP [	$\checkmark$
PG2	n/a	n/a	Input mode	No pull-up a	n/a	uSD_Detect	$\checkmark$
PG6	n/a	Low	Output Pus	No pull-up a	High	LED1 [Green]	$\checkmark$
PJ5	n/a	n/a	External Int	No pull-up a	n/a		$\checkmark$
РКЗ	n/a	Low	Output Ope	No pull-up a	Low	LED4 [Blue]	$\checkmark$
PAO/WKUP (	Configuration : ·						
GPIO mode External Interrupt Mode with Rising edge trigger detection						~	
GPIO Pull-u	ıp∕Pull−down	No pi	ull-up and no p	oull-down			$\sim$
User Label							
Group By Peripherals Apply Ok Cancel							

表 3.3 PG6 設定

項目	設定
GPIO output level	Low
GPIO mode	Output push-pull
GPIO Pull-up/Pull-down	No pull-up and no pull-down
Maximum output speed	High

Pin Configuration ×								
GPIO Single	Mapped Signa	als						
Search Signa Search (Cl	Search Signals       Search (Crtl+F)     Show only Modified Pins							
Pin Name	Signal on	GPIO outp	GPIO mode	GPIO Pull	Maximum o	User Label	Modified	
PA0/WKUP	n/a	n/a	External Int	No pull-up	n/a		$\checkmark$	
PA3	n/a	Low	Output Pus	No pull-up	Low	LCD BL CT	V	
PB0	n/a	Low	Output Pus	No pull-up	Low	EXT RESET	~	
PB2/BOOT1	n/a	Low	Output Pus	No pull-up	Low	OTG_FS1_P	~	
PB7	n/a	n/a	External Int	No pull-up	n/a	OTG_FS1_O	~	
PD4	n/a	Low	Output Ope	No pull-up	Low	LED2 [Oran	~	
PD5	n/a	Low	Output Ope	No pull-up	Low	LED3 [Red]	~	
PE2	n/a	Low	Output Pus	No pull-up	Low	AUDIO_RST	~	
PE3	n/a	Low	Output Pus	No pull-up	Low	SPKR/HP [	$\checkmark$	
PG2	n/a	n/a	Input mode	No pull-up	n/a	uSD_Detect	$\checkmark$	
PG6	n/a	Low	Output Pus	No pull-up	High	LED1 [Gree	~	
PJ5	n/a	n/a	External Int	No pull-up	n/a	LCD INT	$\checkmark$	×
- PG6 Configu	aration :							
GPIO outpu	t level			Low				$\sim$
GPIO mode				Output Push I	Pull			$\sim$
GPIO Pull-u	up/Pull-down			No pull-up and no pull-down				$\sim$
Maximum ou	itput speed			High				$\sim$
User Label	User Label LEDI [Green]							
Group By Peripherals Apply Ok Cancel								



## 3.2.5. 割り込み設定

次に Configuration タブの NVIC ボタンを押します。



以下のように設定します。

表 3.4 割り込み優先度設定

項目	設定
Priority Group	4bits for pre-emption priority 0 bits for subpriority
System tick timer	Enabled: Checked
	Preemption Priority: 15
	Sub Priority: 0
EXTI line0 interrupts	Enabled: Checked
	Preemption Priority: 15
	Sub Priority: 0
Time base: TIM1 update	Enabled: Checked
interrupt and TIM10 global	Preemption Priority: 1
interrupt	Sub Priority: 0

2章で述べたように Preemption Priority には0を設定しないようにしてください。



## 3.2.6. ソースコード生成

C:¥ LED\_BLINK というフォルダを作成します。STM32CubeMX のプルダウンメニュー Project->Setting を選択し以下のように設定します。

表 3.5 STM32CubeMX プロジェクト設定

タブ	設定
Project タブ	Project Name -> CubeMX
	Project Location -> C:¥ LED_BLINK
	Tool chain -> EWARM
Code Generator タブ	Copy only the necessary library files
	Keep User Code when re-generating
	Delete previously generated files when not re-generated

**OK** ボタンを押下すると、設定状態が「**C:¥ LED\_BLINK¥CubeMX¥CubeMX.ioc**」として保存されます。

**Project->Generate Code** を選択すると、**EWARM** のプロジェクトとして、ソースコードが「C:¥ LED\_BLINK ¥CubeMX¥」に生成されます。

以下のメッセージが表示されるので、「Close」で閉じます。

Code Generation						
The Code is successfully generated under C:/LED_BLINK/CubeMX						
Open Folder Open Project Close						



## 3.3. μ C3/Compact の設定

 $\mu$  C3/Compact のコンフィグレータを起動し、STM32F469NI を選択、ターゲットは、「STM32F469I-DISCO」とし、完了をクリックします。

<ul> <li>CPUの選択</li> </ul>		×
CPUを選択しま	説明	
STM32F439ZG           STM32F439ZI           STM32F446MC           STM32F446MC           STM32F446ME           STM32F446RC           STM32F446RC           STM32F446RC           STM32F446RC           STM32F446RC           STM32F446VC           STM32F446VE           STM32F446VE           STM32F446VE           STM32F446ZC           STM32F446ZE           STM32F446ZA           STM32F469AB           STM32F469AB           STM32F469AB           STM32F469BE	Core - ARM 32-bit Cortex-M4 CPU with FPU - Adaptive real-time accelerator - allowing 0-wait state execution performance from Flash memory - Frequency up to 180 MHz - 225 DMIPS/1.25 DMIPS/MHz (Dhrystone 2.1) - DSP instructions Memories - Up to 2 Mbyte of Flash memory - Up to 2 Mbyte of Flash memory - Up to 384+4 Kbytes of SRAM - Flexible external memory controller LCD parallel interface, 8080/6800 modes	<b>^</b>
STM32F469BG     STM32F469BI     STM32F469IE     STM32F469IG     STM32F469II     STM32F469II     STM32F469NE     STM32F469NG	Board - STM32F469I-DISCO	
STM32F469NI	追加アドオンまたはミドルウェアを選択し	
<ul> <li>STM32F469VG</li> <li>STM32F469VI</li> <li>STM32F469ZE</li> <li>STM32F469ZG</li> <li>STM32F469ZI</li> <li>STM32F470AG</li> </ul>	Middleware Descripton	
	< 戻る(B) 完了	キャンセル



次の画面が開いた後、カーネル全般タブの FPU を使用するにチェックを入れます。

uC3Configurator			_	×
: ファイル(F) 表示(V) プロジェ	クト(P) ツール(T) ヘルプ(H)			
: 🔁 🚰 🚺 🛸 😭 🎕 (	00			
Kernel 🗸 🗙				^
◎ カーネル全般	カーネル割込みレベ 	0		 _
<ul> <li>マスシ</li> <li>★** セマフォ</li> <li>▲ イベントフラグ</li> </ul>	最大タスク優先	8		 _
<ul> <li>愛 データキュー</li> <li>         ダメールボックス     </li> </ul>	チック時	1		_
□□ 固定長メモリプール ◎ 周期ハンドラ	ユーザー初期化関			_
✓ 割込みサービスルーナン カーネル管理外割込み は、 #有スタック	アイドル関			_
	追加ヘッダーファイ			
	✓ FPUを使用する			
	システムスタックサイズ			
	タイムイベントハンドラ	1024		
	システムハンドラ(HSTACK)	1024		
	<			> ×
	出力			-
				^
	<			>
Kernel STM32F469	▲ ■ ソース生成			₽
(RAM使用量) システム:60, スタッ	ック :2048, メモリプール :0, トレース機能	能 :0, 合計 :2108 byte		

またチック時が1であることを確認します。

※チック時を変更した場合、自動生成コードで修正する箇所が発生します。その箇所については **3.5** で記述致します。



次に、μ C3/Compact が生成するソースコードから、Cortex-M3,M4の周辺機能である、シ ステムタイマ (SysTick) を利用しなくするための設定を実施します。コア全般の「Systick を使用する」のチェックを外します。

🔹 uC3Configurator		
: ファイル(E) 表示(⊻) プロジェク	ト(E) ツール(I) ヘルプ(H)	
: 💦 💕 💦 🔌 🕋 🕼 🙆 🕕		
STM32F401RE -		
□ □ 7 全般	例外処理	Systick
	Pre-emption bit 4	Systickを使用する
CI GPIOA	例外ハンドラ	割込み番号 15
CP GPIOB		割込みレベル 240 📩
C1 GPIOC	100/h 100/h-9/1 ノ 起動番地 2 NMT	
CP GPIOD	3 Hard fault	
	4 Memory management fault	
th GPIOG	6 Usage fault	低消费雷力
C GPIOH	12 Debug monitor	
w USART1		□ アイドル時スリープモードに入る
📟 USART2		
w USART3		
USART4		
USART5		
■ 050 ACTO ● トレース機能		
	カゴナロル共会ナナ	
	ダノを切り谷えよう	
出	动 / / / / / / / / / / / / / / / / / / /	•
		A
	<u> </u>	۲.
Kernel STM32F401RE	□ ソース生成	4
(RAM使用量) システム:60, スタック:	:2048, メモリプール :0, トレース機能 :0, 合計 :2108 byte	



次に、 $\mu$  C3/Compact の割込みサービスルーチンを登録します。STM32CubeMX が生成した「C:¥ LED\_BLINK ¥CubeMX¥Src¥stm32f4xx\_it」に下記割込みハンドラの関数が登録されていますので、これを $\mu$  C3/Compact のコンフィグレータ上で登録します。これはSTM32CubeMX が生成した割込み処理を $\mu$  C3/Compact の RTOS が管理する割込みサービスルーチンとするための設定です。

#### 表 3.6 割り込みサービスルーチン設定

割り込み番号	起動番地
15	SysTick_Handler
22	EXTI0_IRQHandler
41	TIM1_UP_TIM10_IRQHandler

iuC3Configurator				- 🗆	×
: ファイル(F) 表示(V) プロジェク	クト(P) ツール(T) ^	Jルプ(H)			
i 🔁 🚅 💽 i 🕸 🚰 i 🎕 (	0 0				
Kernel 👻					^
◎ カーネル全般	割込みサービスル・	-チンー		-	
🛅 9 <b>ス</b> ク	割込み番号	起動番地	拡張情報	追加	
<b>*</b> セマフォ	15	SysTick_Handler			
▶ イベントフラグ	22	EXTIO_IRQ Handler			
😨 データキュー	41	TIM1_UP_TIM10_IRQHandler			
🔊 メールボックス					
💷 固定長メモリプール					
🔯 周期ハンドラ					
🕐 割込みサービスルーチン				上へ	
♪ーネル管理外割込み					
🔀 共有スタック				下へ	
				削除	
					~
	出力				-
					~
					$\sim$
	<				>
Kernel STM32F469	↓ □ ソース生成	戈			₽
(RAM使用量) システム:60, スタッ	ック :2048, メモリプール	:0, トレース機能:0, 合計:2108 b	yte		



次は、タスク部分です。以下のタスクを登録します。

#### 表 3.7 タスク設定

ID	関数名	タスク属性(RTOS 起動時のタスク状態)	スタックサイズ
			(Byte)
ID_TASKMAIN	TaskMain	TA_ACT(実行可能状態)	256
ID_TASKLED	TaskLed	休止状態	256

それぞれのタスクは以下の役割を持ちます。

#### ID\_TASKMAIN

STM32CubeMX の自動生成する main()関数内で行っていた各種ペリフェラルの初期化。

#### ID\_TASKLED

USER ボタンの押下待ち状態を slp\_tsk()システムコールで行い、起床後 LED の点滅制御を 行う。

※ここではスタックサイズをデフォルトの 256 としてありますが、アプリケーションのよってはスタックサイズを増やす必要があります。

コンフィグレータでのタスクの実装例を以下に示します。

uC3Configurator										-		$\times$
: ファイル(F) 表示(V) プロジョ	:クト(P) ツール(T) ^	ルプ(H)										
E 📔 🌒 🔝 🗳	00											
Kernel 🗸												^
🍥 カーネル全般	タスクー											
	ID	状態	関数名	優先	スタックサ	記述言語	拡張情報	属性	共有スタ	追加	]	
F ビマフォ F イベントフラグ	ID_TASKMAIN	(実行可能状) 休止状態	TaskMain TaskLed	1	256	高級言語						
2 データキュー	ID_TROTCED	DV TT 17738	TaskLeu		200							
🔊 メールボックス												
□□□ 固定長メモリブール □□□ 国知いいドラ												
◎ 周期ハントン 沙 割込みサービスルーチン										1.0		
◙ カーネル管理外割込み												
🕌 共有スタック										下へ		
	<								>	削除	;	
												~
	出力											•
												$\sim$
												~
Kernel STM32E460	4 回 1/-7生6	<del>,</del>										2 D
(RAM使用量) システム:96. スタ	ック:2560. メモリプール	2) :0.トレース機能:	0. 合計 :2656	bvte								V



クロック、GPIO については、STM32CubeMX で設定するため $\mu$ C3 コンフィグレータ上で は設定しません。

ここで C:¥ LED\_BLINK ¥uC3 を作成した後、「ファイル」->「プロジェクトを保存する」を 選択してコンフィグレータの設定内容 uC3Project.3cf を C:¥ LED\_BLINK ¥uC3 に保存しま す。

🔹 uC3Configurator										- • ×
· ファイル(F) 表示(V) プロジェ	ェクト(P) ツー	ν(T) ∧ι	ノプ(H)							
新規作成(N)	Þ	]								
▶ 🚰 開<(0)	Ctrl+Shift+O									
プロジェクトを閉じる(T)										
□ プロジェクトを保存する(S)	Ctrl+S	*	態	関数名	優先度	スタック	記述言語	拡張情報	属性	追加
	carro	AIN 実	行可能状態	TaskMain	1	256	高級言語			
終了(X)	Alt+F4	FD (#	▼止状態	TaskLed	1	256	高級言語			
20 データキュー										
20 スールボックス										
1 周期ハンドラ										Lم ا
论 割込みサービスルーチン										
○ カーネル管理外割込み										
再 共有人ダック										
	-								÷.	削除
										,
	出力									•
										*
	4									-
Kernel STM32F469NI(S	ע ע דע-	-ス生成								Þ
(RAM使用量) システム :96, スタッ	ク :2560, メモリ	·プール :0,	トレース機能	:0, 合計 :2656 b	yte					

その後「プロジェクト」->「ソース生成」を選択し C:¥ LED\_BLINK ¥uC3 にソースコード 生成を実施します。

🔹 uC3Configurator									_ 0 <b>_ X</b>
· ファイル(F) 表示(V) プロジェイ	クト(P) ツール(T)	へルプ(H)							
i 🖒 🚅 💽 🔌 😁 🕼 😕 ソーン	ス生成(G)	F7							
Kernel 🚰 プロ,	パティ(P) Al	t+F7							
◎ カーネル全般	タスク一覧								_
1 970	ID	状態	関数名	優先度	スタック	記述言語	拡張情報	属性	追加
<ul> <li>セマフォ</li> <li>セマフォ</li> <li>イベントフラグ</li> <li>データキュー</li> <li>メールボックス</li> <li>回鹿長メモリブール</li> <li>回期ノンドラ</li> <li>割込みサービスルーチン</li> <li>カーネル管理外割込み</li> <li>共有スタック</li> </ul>	ID_TASKMAIN ID_TASKLED	実行可能状態 休止状態	TaskMain TaskLed		256 256	高級言語高級言語			<u>Ł</u> ۸ <u></u> F۸
			111					4	削除
	出力								
	*								ł
Kernel STM32F469NI(S	▲ □ ソース生成	•							4
(RAM使用量) システム :96, スタック	フ :2560, メモリプール	:0, トレース機能	:0, 合計 :2656 b	yte					

生成されたファイルの「C:¥LED\_BLINK¥uC3¥main.c」を main\_0.c にリネームします。



#### 3.4. EWARM のプロジェクト設定

この節では、STM32CubeMX が生成したソースコードと、μC3/Compact が生成したソー スコードを用いて EWARM のプロジェクトを作成する手順を示します。

「IAR Embedded Workbench」を起動します。最初に、「プロジェクト」→「新規プロジェ クトの作成」によりプロジェクトを作成します。この時、「空のプロジェクト」を選択し、 「OK」をクリックして空のプロジェクトを作成します。

新規プロジェクトの作成	×
ツールチェーン( <u>」</u> ): ARM ~	
	_
- B 外部ツールで作成した実行可能ファイル - S CMSISPack例	
・ CMSISPackプロジェクトを空にする	
à-%: 「空のプロジェクトを作成します。	
OK ++>>セル	

プロジェクトには、任意の名前を付け保存します。ここでは、 $\mu$ **C3** コンフィグレータが生成したファイルのあるフォルダに、プロジェクト名を「LED\_BLINK」にし「保存」をクリックします。

😢 名前を付けて保存							×
← → ∽ ↑ 📕	> PC > Windows (C:) >	LED_BLINK > uC3		~ Ŭ	uC3の検索		٩
整理 ▼ 新しいフォノ	レダー					•	?
名前	^	更新日時	種類	サイズ			
		検索条件に一致す	る項目はありません。				
ファイル名(N):	LED_BLINK						~
ファイルの種類(T):	プロジェクトファイル (*.ewp)						~
ヘ フォルダーの非表示					保存(S)	キャンセル	



プロジェクトへのファイルの登録は以下のようにします。STM32CubeMX が生成したベク ターテーブル startup stm32f469xx.s は登録しないことに注意してください。



STM32CubeMX と µ C3/Compact の共存方法
 3. サンプルプログラム(ボタン押下→LED 点滅)作成手順
 3.4. EWARM のプロジェクト設定

オプションの設定は以下のように行います。

カテゴリ「一般オプション」の「ターゲット」タブでは、「デバイス」をクリック
 し、さらにその右にあるボタンをクリックしてデバイスリストを表示させます。該当する
 デバイスを選択します。

ノード"LED_BLINK"のオプション					×
カテゴリ:	Library Options 2 ターゲット 出 プロセッサ選択 〇 コア(O) Cc ⑥ デパイス(D) 〇 CMSIS-Pack Nc エンディアンモード ⑥ リトル(L) 〇 ビッグ(B) 〇 BE32(3) ⑧ BE8(8)	カ rtex-M4 rSTM32F4 one 浮動小 FPU(F) Dレジン 〇 Adva	MISRA-C:2004 ライプラリ設定 469NI 数点演算の設定 0 VFPv4 sing スタ(R) 16 ~ anced SIMD (NEON)	MISRA-C:1998 ライブラリオプション	
				OK キャンセ	ιŀ



② カテゴリ「C/C++コンパイラ」の「最適化」タブでは、レベルをなしにします。
 ※この設定は必須ではありません。

ノード"F401RE_LEDonoff"のオプシ	/3V						×
カテゴリ:般オプション 静的)解析 ランタイム解析 C/C++コンパイラ アセンブラ 出力コンバータ カスタムビルド ビルドアクション リンカ デバッガ シミュレータ CADI CMSIS DAP GDBサーバ I-jet/JTAGjet J-Link/J-Trace TI Stellaris PE micro ST-LINK サードパーティドライバ	<ul> <li>         複数ファ 未使     </li> <li>         診断     </li> <li>         言語 1     </li> <li>         レベル(I)     </li> <li>         ● なし     </li> <li>         ● のよ     </li> <li>         ● のよ     </li> <li>         ● のよ     <td>イルのコンパ 用パブリック MISRA- 言語 2 E) (N) H) ランス ] サイズ制限</td><td><ul> <li>イル</li> <li>を破棄</li> <li>-C:2004</li> <li>コード</li> <li>3ード</li> </ul></td><td>MISRA-C: 最適化 使用可能 □ 共-プ □ 関 □ - ド □ 型 □ - ド □ 型 0 - ド 0 0 0 0 0 0 0 0</td><td>1998 出力 3な変換(I 部分式除う インライン化 参動 インライン化 クラスタ スケジューリ ル化</td><td>I I リスト ): た ス解析 ング</td><td>は加オプション プリプロセッサ</td></li></ul>	イルのコンパ 用パブリック MISRA- 言語 2 E) (N) H) ランス ] サイズ制限	<ul> <li>イル</li> <li>を破棄</li> <li>-C:2004</li> <li>コード</li> <li>3ード</li> </ul>	MISRA-C: 最適化 使用可能 □ 共-プ □ 関 □ - ド □ 型 □ - ド □ 型 0 - ド 0 0 0 0 0 0 0 0	1998 出力 3な変換(I 部分式除う インライン化 参動 インライン化 クラスタ スケジューリ ル化	I I リスト ): た ス解析 ング	は加オプション プリプロセッサ



STM32CubeMX と µ C3/Compact の共存方法 3. サンプルプログラム(ボタン押下→LED 点滅)作成手順 3.4. EWARM のプロジェクト設定

③ 「プリプロセッサ」タブで追加インクルードディレクトリに

\$PROJ\_DIR\$¥..¥CubeMX¥Inc

\$PROJ\_DIR\$4..4CubeMX4Drivers4STM32F4xx\_HAL\_Driver4Inc

\$PROJ\_DIR\$4..4CubeMX4Drivers4STM32F4xx\_HAL\_Driver4Inc4Legacy

\$PROJ\_DIR\$¥..¥CubeMX¥Drivers¥CMSIS¥Device¥ST¥STM32F4xx¥Include

\$PROJ\_DIR\$¥..¥CubeMX¥Drivers¥CMSIS¥Include

\$PROJ\_DIR\$¥.

を登録します。またシンボル定義には

USE\_HAL\_DRIVER

STM32F469xx

を登録します。

ノード"LED_BLINK"のオプション				×
カテゴリ:			I	場出荷時設定
一般オプション 🔨	□ 複数ファイルのコンパイル			
青鱼的解释析	未使用パブリックを破棄			
ランタイム解析	診断 MISRA-C:200	MISRA-C:1998	エンコード	追加オプション
C/C++コンパイラ	言語1 言語2 コード	最適化 出力	リスト	プリプロセッサ
アセンブラ 出力コンバータ カスタムビルド ビルドアクション リンカ デバッガ シミュレータ CADI CMSIS DAP GDBサーバ I-jet/JTAGjet	<ul> <li>一標準のインクルードディレ 追加インクルードディレクトリ</li> <li>\$PROJ_DIR\$¥¥CubeM3</li> <li>\$PROJ_DIR\$¥¥CubeM3</li> <li>\$PROJ_DIR\$¥¥CubeM3</li> <li>\$PROJ_DIR\$¥¥CubeM3</li> <li>\$PROJ_DIR\$¥¥CubeM3</li> <li>\$PROJ_DIR\$¥¥CubeM3</li> </ul>	クトリを無視(I) (A):(1行に1ディレクトリ) ¥Inc (¥Drivers¥STM32F4xx (¥Drivers¥STM32F4xx (¥Drivers¥CMSIS¥Dev (¥Drivers¥CMSIS¥Incl	-HAL_Driver -HAL_Driver vice¥ST¥STM ude	¥Inc ¥Inc32F4
J-Link/J-Trace TI Stellaris PE micro ST-LINK サードパーティドライバ	シンボル定義(D):(1行に1シ USE_HAL_DRIVER STM32F469xx	ンボル) コ マ #	マイルへのプリフ メントの保持(i lineディレクティ	<b>プロセッサ出力(F</b> こ) (ブ生成(G)
		L	OK	キャンセル



## 「診断」タブで診断を無効化に Pe170 を登録します。





5 カテゴリ「リンカ」の「設定」タブでは、リンカ設定ファイルをμC3 が生成した
 \$PROJ\_DIR\$¥STM32F4\_flash.icf に変更します。

ノード"F401RE_LEDonoff"のオプジ	VEV					×
カテゴリ:					E	口場出荷時設定
高いシンシン 静的解析 ランタイム解析 C/C++コンパイラ	リスト	#define	診断	チェックサム	エンコード	追加オプション
アセンブラ 出力コンパータ カスタムビルド	リンカ設	ライフラリ 定ファイル( <u>L</u> ) - ォルトのオーバラ	入)J ライド(O)	取週112	アトハンノ	
ビルドアクション リンカ デバッガ	\$P	ROJ_DIR\$¥S	TM32F4_fla	sh.icf		
シミュレータ CADI CMSIS DAP	設定ファ1	編集(E) /ルシンボルの5	官義( <u>C</u> )(1行	に1個)		
GDBサーバ I-jet/JTAGjet J-Link/J-Trace						^
TI Stellaris PE micro ST-LINK						
サードパーティドライバ						~
					OK	キャンセル

またリンカスクリプトをテキストエディタで開き HEAP 領域を確保します。

/*###ICF### Section handled by ICF editor, don't touch! ****/
/*-Sizes-*/
define symbolICFEDIT_size_cstack = 0x400;
define symbolICFEDIT_size_heap = 0x200;
/**** End of ICF editor section. ###ICF###*/
define block CSTACK with alignment = 8, size =ICFEDIT_size_cstack { };
define block HEAP with alignment = 8, size =ICFEDIT_size_heap { };
····
place at end of RAM_region { block order { block HEAP, block HSTACK, block CSTACK }};



⑥ 「ライブラリ」タブでは、追加ライブラリに *μ* **C3** が生成した

**\$PROJ\_DIR\$¥uC3cortexm4fl.a** を登録します(FPU を使用する場合のカーネル)。またエント リシンボルを ResetHandler とします。

ノード"F401RE_LEDonoff"のオプシ	νeν					×
カテゴリ:						E場出荷時設定
<ul> <li>一般オプション</li> <li>         静的解析         <ul> <li>             テンタイム解析         </li> </ul> </li> </ul>	1171	Hala Cons	=A.147	1	T L <sup>a</sup>	\ <u>←</u> ±□.±=°\\.
C/C++コンパイラ アセンブラ	リスト 設定	#define ライブラリ	診断 入力	テェックリム 最適化	エクコート アドバンス	追加17939
出力コンバータ カスタムビルド ビルドアクション	✓ 自動ラ 追加ライブ	ンタイムライブラ 'ラリ (L):(1行に	<b>ジ選択 (<u>A</u>)</b> [1個)	)		
リンカ デバッガ シミュレータ CADI CMSIS DAP	\$PROJ_[	DIR\$¥uC3cort	exm4fl.a			^
GDBサーバ I-jet/JTAGjet J-Link/J-Trace TI Stellaris	ダデフォル ● エン ○ エン	レトのプログラム ハトリシンボル(E ハトリ シンボルな	エントリをオ・ :) <mark>ResetH</mark> にし( <u>N</u> )	ーバライドする ( <u>(</u> landler	<u>)</u>	
PE micro ST-LINK サードパーティドライバ						
					ОК	キャンセル



「デバッガ」カテゴリは、使用状況によって変わりますが今回の環境では「設定」
 タブのドライバを「ST-LINK」に設定しました。それに合わせて「ST-LINK」カテゴリの「設定」
 定」タブでインターフェースを SWD にします。

ノード"LED_BLINK"のオプション		×
ノード"LED_BLINK"のオプション カテゴリ: 	設定 通信 ブレークポイント Iミュレータ ST-LINK v2 シリアル □ プローブを選択するプロンプトを常に表示します リセット(T) システム (デフォルト) ~ () JTAG ④ SWD 「フォルト ~ () 指定	工場出荷時設定
	OK	キャンセル



#### 3.5. 自動生成コードの修正

初期化に関する修正

C:¥ LED\_BLINK¥CubeMX¥Src¥main.c を開き以下の3カ所に修正を加えます。

int main(void)関数内の while ループをコメントアウト。

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
//while (1)
//{
/* USER CODE END WHILE */
/* USER CODE BEGIN 3 */
```

```
//}
/* USER CODE END 3 */
```

int main(void)を void TaskMain(VP\_INT exinf)にリネーム。

void TaskMain(VP\_INT exinf)

#### インクルードの追加。

/\* USER CODE BEGIN Includes \*/
/\* {{UC3\_INCLUDE \*/
#include "kernel.h"
#include "kernel\_id.h"
#include "hw\_dep.h"
/\* }}UC3\_INCLUDE \*/
/\* USER CODE END Includes \*/



**またμC3側が自動生成したスタートアップ処理 (prst.s79)**でSTM32CubeMXの生成コードを 利用する部分は修正します。具体的にはクロックの設定と GPIO の設定を行う

「Idr r0, =REG\_RCC」から「; Enter the iar code」の手前の部分を以下のように修正します。

EXTER EXTER	RN SystemInit RNiar_program_start EXTERN SystemInit を追加。
PUBLI	C ResetHandler
ResetHandler:	
mrs	r0, control
ldr	r1, =SFE(CSTACK)
orr	r0, r0, #0x02
ldr	r2, =SFE(HSTACK)
bic	r0, r0, #0x01
msr	psp, r1
msr	msp, r2
msr	control, r0
ldr	r0, =SystemInit SystemInit を用いて初期化を行います。
blx	r0
; Enter the iar c	code
ldr	r0, =iar_program_start



bx

 $\mathbf{r}\mathbf{0}$ 



STM32CubeMX と µ C3/Compact の共存方法
 3. サンプルプログラム(ボタン押下→LED 点滅)作成手順
 3.5. 自動生成コードの修正

② 割り込みサービスルーチンとタスク

C:¥ LED\_BLINK¥uC3¥main\_0.c 内の割り込みサービスルーチンとタスクのスケルトンコードを コメントアウトします。

```
/******
       SysTick_Handler
 *****************************/
/* {{UC3_ISR(SysTick_Handler) */
//void SysTick_Handler(VP_INT exinf)
//{
//}
                                  STM32CubeMX 側のものを用いるのでコ
/* }}UC3 ISR */
                                  メントアウトします
(中略)
/*********
       TaskMain
****************************/
/* {{UC3_TASK(TaskMain) */
//void TaskMain(VP_INT exinf)
//{
//}
/* }}UC3_TASK */
```

Copyright © eForce Co.,Ltd. All Rights Reserved



また C:¥LED\_BLINK¥CubeMX¥Inc¥stm32f4xx\_it.h 内に以下のインクルード文を追加します。

/\* Includes -----\*/

/\* {{UC3\_INCLUDE \*/

#include "kernel.h"

#include "kernel\_id.h"

#include "hw\_dep.h"

/\* }}UC3\_INCLUDE \*/

さらに割り込みサービスルーチンの引数を void から VP\_INT exinf に変更します。 C:¥LED\_BLINK¥CubeMX¥Inc¥stm32f4xx\_it.h だけでなく C:¥LED\_BLINK¥CubeMX¥Src¥stm32f4xx\_it.c に対しても行います。

void SysTick\_Handler(VP\_INT exinf);

void EXTI0\_IRQHandler(VP\_INT exinf);

void TIM1\_UP\_TIM10\_IRQHandler(VP\_INT exinf);

これは µC3/Compactの割込みサービスルーチンの関数とするための修正です。



③ C:¥ LED\_BLINK¥CubeMX¥Src¥main.c に割り込みサービスルーチンとタスクの実 装を追加します。

TaskMain() 内の /\* USER CODE BEGIN 2 \*/~/\* USER CODE END 2 \*/の間で act\_tsk(ID\_TASKLED)をコールして LED タスクを起床させる処理を追加。



/\* USER CODE BEGIN 4 \*/~/\* USER CODE END 4 \*/の間に LED タスクの関数を記述します。



/\* USER CODE BEGIN 4 \*/~/\* USER CODE END 4 \*/の間にボタン押下を検出する処理を追 加します。



STM32CubeMX と µ C3/Compact の共存方法
 3. サンプルプログラム(ボタン押下→LED 点滅)作成手順
 3.5. 自動生成コードの修正

#### STM32CubeMXの SYSTICKの更新をµC3 側に反映させるため処理を追加します。

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    /* USER CODE BEGIN Callback 0 */
    if (htim->Instance == TIM1) {
        HAL_IncTick();
        isig_tim(); _______isig_tim()でSYSTICKの更新をµC3 側に反映します
    }
    /* USER CODE BEGIN Callback 1 */
}
```

※μC3/Compact のコンフィグレータで「Systick を使用する」のチェックを外した場合タイマ の割り込みサービスルーチンから isig\_tim()を呼ぶ必要があります。またμC3/Compact のコン フィグレータでチック時を1から変更した場合、タイマの割り込みサービスルーチンの周期を 変更する必要があります。具体的には

C:¥ LED\_BLINK¥CubeMX¥Src¥stm32f4xx\_hal\_timebase\_TIM.c 内の関数 HAL\_InitTick()に おいて行っている以下の処理を修正します。

```
htim1.Init.Period = (1000000 / 1000) - 1;
```

例えばチック時を2とした場合は

```
htim1.Init.Period = (1000000 * 2 / 1000) - 1;
```

とします。



④ C:¥ LED\_BLINK¥CubeMX¥Src¥main.cの void SystemClock\_Config(void)の先頭に HAL\_RCC\_Delnit()を追加します。

<pre>void SystemClock_Config(void)</pre>	
{	HAL_RCC_Delnit()を追加します
HAL_RCC_DeInit();	

これにより SystemClock\_Config()が複数回コールされる度にクロックの再設定が行えるようになります。

ビルドして実行すると、USER ボタンを押下する度に LED が点滅します。

また以下のようにプログラムを修正すると10sごとにLEDが点滅するようになりTICKの更新周期を確認することができます。

```
void TaskLed(VP_INT exinf)
{
    while(1){
        //slp_tsk();
        dly_tsk(10000);
        HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
    }
}
```

## 3.6. 動作確認環境

動作確認にはボード上の以下のコンポーネントを使用します。

ボード上のコンポーネント	備考
CN1 ST-LINK/V2-1	電源供給及びデバッグのため、USB2.0 ケーブル(mini-B タイ
	プ)の mini-B 側を接続します(A 側は PC に接続します)。
B2 Wake-up button	外部割込みのため使用します。
LD1	緑色 LED

記号については ST マイクロエレクトロニクス提供 User manual UM1932: Discovery kit with STM32F469NI MCU(<u>http://www.st.com/resource/en/user\_manual/dm00218846.pdf</u>)を参照して ください。



## 4. サンプルプログラム(USB-HID)作成手順

この章では、「USER ボタンを押すと PC 上のマウスポインタが移動する」プログラムの作成手順を説明します。3章の説明と重複する箇所については簡略的な説明にしてあります。

#### 4.1. フォルダ構成

最上位のフォルダ名が LED BLINK→USB-HID となる以外は 3.1 の説明と同じになります。

#### 4.2. STM32CubeMX を用いたプロジェクト作成

#### 4.2.1. 新規プロジェクトの作成

3.2.1 で説明した手順と同じになります。

#### 4.2.2. Pinout 設定

Pinout タブを選択し以下のように設定します。

#### USER ボタン押下を検知する割り込みの設定。

3.2.2 と同じく PAO/WKUP を GPIO\_EXTIO に設定します。

#### RTOS を動かすための設定。

3.2.2 と同じく SYS の設定で TimeBase Source を TIM1 と設定します。



**STM32CubeMX** と µ C3/Compact の共存方法 4. サンプルプログラム(USB-HID)作成手順 4.2. **STM32CubeMX** を用いたプロジェクト作成

## USB を使用するための設定。

左ペインから Configuration->Peripherals->RCC を選択し、HSE を Crystal/Ceramic に設

定します。

次に Configuration->Peripherals->USB\_OTG\_FS を選択し、Mode を Device Only に設定

 $\sim$ 

し、Activate\_VBUS にチェックを入れます。



そして Configuration->MiddleWares->USB\_DEVICE を選択し、Class For FS IP を Human

Interface Device に設定します。





## 4.2.3. クロック設定

Clock Configration タブを選択し以下のように設定します。

表 4.1 クロック設定

項目	設定
System CLK Mux	PLLCLK
PLL source	HSE
Input frequency	8
PLL M	8
PLL N	100
PLL P	2
PLL Q	7
APB1 Prescaler	4
APB2 Prescaler	2
PLL48CLK Mux	PLLSAIP
PLLSAI N	384
PLLSAI P	8





STM32CubeMX と µ C3/Compact の共存方法 4. サンプルプログラム(USB-HID)作成手順 4.2. STM32CubeMX を用いたプロジェクト作成

#### 4.2.4. GPIO 設定

次に、Configuration タブの GPIO ボタンを選択します。

PA0/WKUPの設定を確認します。ここは 3.2.4 と同じ設定になります。

表 4.2 PA0/WKUP 設定

項目	設定
GPIO mode	External Interrupt Mode with Rising edge trigger detection
GPIO Pull-up/Pull-down	No pull-up and no pull-down



**STM32CubeMX** と µ C3/Compact の共存方法 4. サンプルプログラム(USB-HID)作成手順 4.2. **STM32CubeMX** を用いたプロジェクト作成

## 4.2.5. USB 設定

## Configuration タブの USB\_OTG\_FS ボタンを押します。

STM32CubeMX CubeMX ir	och STM32E469NIHy STM32				-		×
File Project Window Help	CC COMPLETERING STMDE	analog					0
	+ - 🦉 🖗				3	f 🔽 (	Þ×
Pinout Clock Configuration Con	figuration Power Consumption	Calculator					
Configuration Co	Multimedia	Connectivity LEBJ75 Harris	Heid LUBBOR	NVARUS ADE Harry System GPIO	Control		security

Low Power を Enabled に設定し、OK ボタンを押下します。

USB_OTG_FS Configuration				
🖋 Parameter Settings 🚽 User Constants 🚽 NVIC Settings 🚽 GPIO Settings				
Configure the below parameters :				
Search : Search (Crtl+F)	] ♥ ♠			
Speed	Full Speed 12MBit/s			
Endpoint O Max Packet size	64 Bytes			
Enable internal IP DMA	Disabled			
Low power 🔇	Enabled			
Link Power Management	Disabled			
VBUS sensing	Enabled			
Signal start of frame	Disabled			
Low power low_power_enable				
Restore Default	Apply Ok Cance	el 👘		



Configuration タブの GPIO ボタンを選択します。さらに USB\_OTG\_FS タブを選択し PA9 を以下のように設定します。

#### 表 4.3 PA9 設定

項目	設定
GPIO mode	Input mode
GPIO Pull-up/Pull-down	No pull-up and no pull-down

ISB_O	TG_FS Configuratior	ı					×
🕜 Parameti	er Settings 🚽 User (	Constants 🗹 NVIC Sett	ings √ GPIO Settings				
Search Sig Search (C	nals Ort/+F)					Shov	v only Modified Pins
Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pull-down	Maximum output speed	User Label	Modified
PA9	USB_OTG_FS_VBUS	n/a	Input mode	No pull-up and no pull-down	n/a	VBUS_FS1	✓
PA11	USB_OTG_FS_DM	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Very High	USB_FS1_N	$\checkmark$
PA12	USB OTG FS DP	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Very High	USB FS1 P	~
B40.0							
PA9 Coning	uration.						
GPIO mode	Э			Input mode			~
GPIO Pull-	-up/Pull-down			No pull-up and no pull-down			~
User Label				VBUS_FS1			
Restore I	Default					Apply 0	Ok Cancel

STM32CubeMX と µ C3/Compact の共存方法 4. サンプルプログラム(USB-HID)作成手順 4.2. STM32CubeMX を用いたプロジェクト作成

## 4.2.6. 割り込み設定

Configuration タブの NVIC ボタンを押し以下のように設定します。

表	4.	4	割	Ŋ	込み	⊁優·	先	度設	定
---	----	---	---	---	----	-----	---	----	---

項目	設定
Priority Group	4bits for pre-emption priority 0 bits for subpriority
Pendable request for	Enabled: Checked
system service	Preemption Priority: 15
	Sub Priority: 0
System tick timer	Enabled: Checked
	Preemption Priority: 15
	Sub Priority: 0
EXTI line0 interrupts	Enabled: Checked
	Preemption Priority: 15
	Sub Priority: 0
Time base: TIM1 update	Enabled: Checked
interrupt and TIM10 global	Preemption Priority: 1
interrupt	Sub Priority: 0
USB ON The Go FS	Enabled: Checked
wake-up interrupt	Preemption Priority: 5
though EXT line 18	Sub Priority: 0
USB ON The Go FS	Enabled: Checked
global interrupt	Preemption Priority: 5
	Sub Priority: 0

2章で述べたように Preemption Priority には 0 を設定しないようにしてください。

#### 4.2.7. **ソースコード生成**

フォルダ名とプロジェクト名を LED\_BLINK→USB-HID と変更する以外は 3.2.6 で説明した 内容と同じになります。 STM32CubeMX と µ C3/Compact の共存方法 4. サンプルプログラム(USB-HID)作成手順 4.3. µ C3/Compact の設定

#### 4.3. *μ* C3/Compact の設定

μ C3/Compact のコンフィグレータの起動、ターゲットの選択、FPU の使用、システムタ イマ (SysTick) を利用しなくするための設定は、3.3 と同じ手順となります。

割込みサービスルーチンの登録についてはSTM32CubeMX が生成した以下の割込みハンド ラの関数を μ C3/Compact のコンフィグレータ上で登録します。

表 4.5 割り込みサービスルーチン設定

割り込み番号	起動番地
15	SysTick_Handler
22	EXTI0_IRQHandler
41	TIM1_UP_TIM10_IRQHandler
58	OTG_FS_WKUP_IRQHandler
83	OTG_FS_IRQHandler

タスクは以下を登録します。

表 4.6 タスク設定

ID	関数名	タスク属性	スタックサイズ
		(RTOS 起動時のタスク状態)	(Byte)
ID_TASKMAIN	TaskMain	TA_ACT(実行可能状態)	512
ID_TASKUSBMOUSE	TaskUsbMouse	休止状態	256

それぞれのタスクは以下の役割を持ちます。

#### ID\_TASKMAIN

STM32CubeMX の自動生成する main()関数内で行っていた各種ペリフェラルの初期化。

#### ID\_TASKUSBMOUSE

USER ボタンの押下待ち状態を slp\_tsk()システムコールで行い、起床後マウスポインタを 動かす。

※ここでは TaskMain のスタックサイズを 512 としてありますが、アプリケーションのよってはさらにスタックサイズを増やす必要があります。

クロック、GPIO については、STM32CubeMX で設定するためμC3 コンフィグレータ上で は設定しません。



コンフィグレータの設定内容の保存方法、ソースコード生成についてはフォルダ名を USB-HID とする以外は3章と同じ内容となります。そして生成されたファイルの 「C:¥USB-HID¥uC3¥main.c」を main\_0.c にリネームすることも3章と同じになります。

#### 4.4. EWARM のプロジェクト設定

この節では、STM32CubeMX が生成したソースコードと、μC3/Compact が生成したソー スコードを用いて EWARM のプロジェクトを作成する手順を示します。

「IAR Embedded Workbench」を起動します。最初に、「プロジェクト」→「新規プロジェクトの作成」によりプロジェクトを作成します。この時、「空のプロジェクト」を選択し、「OK」をクリックして空のプロジェクトを作成します。

プロジェクトには、任意の名前を付け保存します。ここでは、μ**C3** コンフィグレータが生成したファイルのあるフォルダに、プロジェクト名を「USB-HID」にし「保存」をクリックします。

プロジェクトへのファイルの登録は以下のようにします。



STM32CubeMX と µ C3/Compact の共存方法 4. サンプルプログラム(USB-HID)作成手順 4.4. EWARM のプロジェクト設定



オプションの設定内容で 3.4 と異なる箇所は、「プリプロセッサ」タブで追加インクルード ディレクトリに

\$PROJ\_DIR\$¥..¥CubeMX¥Inc

\$PROJ\_DIR\$¥..¥CubeMX¥Drivers/STM32F4xx\_HAL\_Driver/Inc

\$PROJ\_DIR\$\\_.\CubeMX\Drivers/STM32F4xx\_HAL\_Driver/Inc/Legacy

\$PROJ\_DIR\$¥..¥CubeMX¥Middlewares/ST/STM32\_USB\_Device\_Library/Core/Inc

\$PROJ\_DIR\$¥..¥CubeMX¥Middlewares/ST/STM32\_USB\_Device\_Library/Class/HID/Inc

\$PROJ\_DIR\$¥..¥CubeMX¥Drivers/CMSIS/Device/ST/STM32F4xx/Include

\$PROJ\_DIR\$\$..\$CubeMX\$Drivers/CMSIS/Include

\$PROJ\_DIR\$/.

を登録する箇所です。その他は3.4の設定と同じになります。



**STM32CubeMX** と µ C3/Compact の共存方法 4. サンプルプログラム(USB-HID)作成手順 4.5. 自動生成コードの修正

#### 4.5. 自動生成コードの修正

初期化に関する修正

C:¥ USB-HID¥CubeMX¥Src¥main.c を開き以下の修正を加えます。

**int main(void)関数内の MX\_USB\_DEVICE\_Init()**呼び出しをコメントアウト。 USB の初期化関数 MX\_USB\_DEVICE\_Init()は USB タスク TaskUsbMouse()から呼び出すので ここでの呼び出しはコメントアウトします。

int main(void)関数内の while ループをコメントアウト。 int main(void)を void TaskMain(VP\_INT exinf)にリネーム。 これらは 3.5 と同じ修正です。

インクルードの追加。

/\* USER CODE BEGIN Includes \*/

/\* {{UC3\_INCLUDE \*/

#include "kernel.h"

#include "kernel\_id.h"

#include "hw\_dep.h"

/\* }}UC3\_INCLUDE \*/

#include "usbd\_hid.h"

/\* USER CODE END Includes \*/

3.5 で記載したファイルの他、"usbd\_hid.h"が追加しています。

またμC3 側が自動生成したスタートアップ処理 (prst.s79)で修正する箇所も 3.5 と同じにな ります。



**STM32CubeMX** と µ C3/Compact の共存方法 4. サンプルプログラム(USB-HID)作成手順 4.5. 自動生成コードの修正

② 割り込みサービスルーチンとタスク

C:¥ USB-HID ¥uC3¥main\_0.c 内の割り込みサービスルーチンとタスクのスケルトンコードをコ メントアウトすること、C:¥ USB-HID ¥CubeMX¥Inc¥stm32f4xx\_it.h 内に以下のインクルード文 を追加することも 3.5 と同じです。

/\* Includes -----\*/

/\* {{UC3\_INCLUDE \*/

#include "kernel.h"

#include "kernel\_id.h"

#include "hw\_dep.h"

/\* }}UC3\_INCLUDE \*/

割り込みサービスルーチンの引数を void から VP\_INT exinf に変更することも 3.5 と同じです。

void SysTick\_Handler(VP\_INT exinf);

void EXTI0\_IRQHandler(VP\_INT exinf);

void TIM1\_UP\_TIM10\_IRQHandler(VP\_INT exinf);

void OTG\_FS\_WKUP\_IRQHandler(VP\_INT exinf);

void OTG\_FS\_IRQHandler(VP\_INT exinf);

③ C:¥ USB-HID¥CubeMX¥Src¥main.c に割り込みサービスルーチンとタスクの実装 を追加します。

TaskMain() 内 の /\* USER CODE BEGIN 2 \*/~/\* USER CODE END 2 \*/ の 間 で act\_tsk(ID\_TASKUSBMOUSE)をコールして USB MOUSE タスクを起床させる処理を追加。



# /\* USER CODE BEGIN 4 \*/~/\* USER CODE END 4 \*/の間に USB MOUSE タスクの関数を 記述します。





/\* USER CODE BEGIN 4 \*/~/\* USER CODE END 4 \*/の間にボタン押下を検出する処理を追 加します。

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == A0_Pin){
        iwup_tsk(ID_TASKUSBMOUSE);
        USER ボタンが押されると USB MOUSE タスクを起床
        します。
```

STM32CubeMX の SYSTICK の更新を µ C3 側に反映させるため処理の追加、C:¥ USB-HID ¥CubeMX¥Src¥main.c の void SystemClock\_Config(void)の先頭に HAL\_RCC\_DeInit()を追加す ることは 3.5 と同じです。

## Sleep 設定を変更。

C:¥ USB-HID ¥CubeMX¥Src¥ usbd\_conf.c の HAL\_PCD\_SuspendCallback()と HAL\_PCDEx\_LPM\_Callback()で行っている Sleep 設定において SCB\_SCR\_SLEEPONEXIT\_Msk を無効にします。

/\* SCB->SCR |=(uint32\_t)((uint32\_t)(SCB\_SCR\_SLEEPDEEP\_Msk | SCB\_SCR\_SLEEPONEXIT\_Msk)); \*/

SCB->SCR |= (uint32\_t)((uint32\_t)(SCB\_SCR\_SLEEPDEEP\_Msk));

SCB\_SCR\_SLEEPONEXIT\_Msk を無効にします。

ビルドして実行すると、USER ボタンを押下する度に PC 上のマウスポインタが移動します。



#### 4.6. 動作確認環境

動作確認にはボード上の以下のコンポーネントを使用します。

ボード上のコンポーネント	備考
CN1 ST-LINK/V2-1	電源供給及びデバッグのため、USB2.0 ケーブル(A mini-B
	タイプ)のmini-B側を接続します(A側はPCに接続します)。
B2 Wake-up button	外部割込みのため使用します。
CN13 USB OTG FS Connector	マウスポインタカーソル移動のため、USB2.0 ケーブル(A
	micro-B タイプ)の micro-B 側を接続します(A 側は PC に接
	続します)。

記号については ST マイクロエレクトロニクス提供 User manual UM1932: Discovery kit with STM32F469NI MCU(<u>http://www.st.com/resource/en/user\_manual/dm00218846.pdf</u>)を参照して ください。



STM32CubeMX と μ C3/Compact の共存方法





STM32CubeMX と µ C3/Compact の共存方法

#### AN201701 STM32CubeMX と µ C3/Compact の共存方法

2017年09月29日 Rev.1.0

イー・フォース株式会社 http://www.eforce.co.jp/ お問い合わせ support@eforce.co.jp Copyright (C) 2017 eForce Co.,Ltd. All Rights Reserved.

