



# Atmelソフトウェア枠組み(ASF)プロジェクトの ASFv3.3からASFv4への移植

### 序説

この応用記述はAtmel Start ASFv3プロジェクトからASFv4プロジェクトへの移植の検討導入を提供します。

ASFv4のコート 密度に対する最適化と追加された柔軟性のため、2つの枠組みは相互排他です。そしてASF v3.37<sup>°</sup>ロジェクトがASF v4.0によって構築することができないことを意味します。これはフ<sup>°</sup>ロジェクト移植に"根本的"手法が必要です。

### ASFv3

ASF 3基本構造は主な目標としてコート・量、性能、低電力の最適化で開発されました。周辺機能ト・ライハ・とミト・ルウェアは ASFが許されたプロジェクト使用時の設計段階減少を提供するために開発されました。

ASFの意図はお客様の設計時間を減らすためにAtmelの熟練者によって開発された実績のあるドライバとコード単位部の豊富な一式を提供することです。これはハードウェアと高価値ミドルウェアに対する抽象化を提供して、マイクロ コントローラの使い方を簡単化します。

ASFはソースコード単位部とこれらの使い方を実演する応用から成ります。

- ・**ドライハ**は周辺機能やデバイス特有機能をアクセスするための低位レジスタインターフェース関数を提供するdriver.cとdriver.h から成ります。サービスと部品はこのドライバをインターフェースします。
- ・サービスはUSBクラス、FATファイルシステム、基本構造最適化したDSPライブラリ、図画ライブラリなどのようなもっと応用指向ソフトウェアを提供する単位部型です。
- ・部品はメモリ(他えば、Atmel DataFlash<sup>®</sup>、SDRAM、SRAM、NANDフラッシュ)、表示器、感知器、無線などのような外部 ハートウェア部品をアクセスするためのソフトウェアトライバを提供する単位部型です。
- ・基板はAtmelの開発キットの各入出力ピンに対する全てのデジタルとアナログの周辺機能の割り当てを含みます。

### ASFv4

Atmel Startではドライバとソフトウェア階層がAtmelソフトウェア枠組み(Atmel Software Framework)の次のソフトウェア開発の一部 (ASFv4)として提供されます。この版は0から構築され、旧ASF版の使用者と貢献者によって報告された問題を解決す るためとAtmel Startウェブ使用者インターフェースでのより良い統合のためにこの枠組み全体の完全な再設計と実装です。 それにもかかわらず、ASFを経験した使用者に対してASFv4の親近感を保ち、新規使用者に対して未だ開始が容易 なことが目標でした。ASFv4でのいくつかの変更はこの版に対する必要条件に合致するために必要で、最も重要な変 更はAtmel Start使用者の手引きの「ASFv4対ASFv3評価基準」で一覧にされます。

ASFv4はAtmel Startに密接に統合され、そしてそれはASFv4コートが以前よりもよりもっと使用者の仕様に誂えることができることを意味します。例えば、許可/禁止コート部に対してC前処理条件式を使う代わりに、プロジェクトソースから禁止コート部を完全に取り去ることができ、コートを読むことをより綺麗でより容易にします。Atmel Startへの統合はソフトウェア構成設定がよりもっと使用者に友好的な環境で行われ、デバイスで設定された構成設定情報だけが生の周辺機能レジスタ内容で、それはファームウェアイメージをよりもっと簡潔にします。

我々が取り掛かった1つの重要な問題はASFに基づくコート、のメモリ量と性能です。ASFv3に基づくコート、を走らせるためのフラッシュメモリ必要条件は多くの使用者によって多すぎると思われていました。これはコート、生成を使って周辺機能が初期化される方法を変更することによって処理されました。報告されている性能の問題は代表的に大きな割り込み遅延/遅いコート、実行で、より小さくて複雑でない割り込み処理部を作ることによって解決されました。

### 移植

あいにく、ASFv3プロジェクトをASFv4プロジェクトへ直接移植する方法がありません。新しい基本設計とAPIで、移植の方法はオンラインのAtmel Startツール(start.atmel.com)で0から(元の)ASFv3プロジェクトを再構築するだけです。同時にこれは現在手応えのある量を示すかもしれず、ASFv3プロジェクトを用いるデバイス間移植よりもASFv4プロジェクトを用いるデバイス間の将来の移植はよりもっと容易でしょう。

本書は一般の方々の便宜のため有志により作成されたもので、Microchip社とは無関係であることを御承知ください。しおりの[はじめに]での内容にご注意ください。

### 例プロジェクト開始

移植処理を通して段階実行するには、例プロジェクトがASF v3.3例プロジェクト一覧から使われます。使われる例プロジェクトはSAMD21 Xpl ained Pro評価基板で使われるように選ばれます。

- 1. Atmel StudioでFile(ファイル)⇒New(新規)⇒Example Project(例プロジェクト)を選ぶことでASFv3プロジェクトを読み込んでください。
- 2. 図1で示されるように、Device Family(デバイス系統)メニューでSAMD21を選んでください。

図 1. ASEv31例ノロンエ		
New Example Project fro	m ASF or Extensions	
Device Family: SAMD21	Category: Drivers     Search for Example Projects	
All Projects Kit Category Technology Addon	<ul> <li>BST - Bosch Sensortec GmbH (0) 1.0.1</li> <li>Atmel - Atmel Corp. (73) 3.32.0</li> <li>Match Frequency Example for the SAM TC Driver - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM AC Driver (Callback) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM AC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM ADC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM ADC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM ADC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM ADC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM ADC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM ADC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM BOD Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM CRC32 Driver - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM D21/R21/D11/L21/DA1 SERCOM USART Driver with DMA - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM DAC Driver (Callback) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM DAC Driver (Callback) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM D21/R21/D11/L21/L22/DA1/C21 TC Driver (DMA) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM DAC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM DAC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM DAC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM DAC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM DAC Driver - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM DAC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM DAC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM DAC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM DAC Driver (Polled) - SAM D21 Xplained Pro</li> <li>Quick Start for the SAM EVENTS Driver (Interrupt Hook) - SAM D21 Xplained Pro</li></ul>	4 H
	K	

- 3. 全てのAtmelプロジェクトを表示するためにAll Projects(全プロジェクト)をクリックし、その後にQuick Start for the SAM ADC Driver (Callb ack) SAM D21 Xplained Pro(SAM ADCトライハ(呼び戻し)用即時開始 SAMD21 Xplained Pro)を選んでください。
- 4. 既定のAtmel Sudio作業空間でプロジェクトを作成するためにOKをクリックしてください。
- 5. 例プロジェクトをコンパイルして走らせてください。
- 6. 応用説明

例のADCプログラム流れ図が図2で示されます。例応用は試料の一式を収集するためにADCを 開始し、その後に一度データ集合が完了するのを永遠に待ちます。

応用は非常に単純ですが、移植に対して2つの段階に分断される必要があります。赤で外廓を描かれた段階1は後でAtmel Startで使うためにASFv3でシステムとADC初期化から必要とされる情報を抽出します。緑で外廓を描かれた段階2は応用で必要とされるAPIの機能的な確認で、Atmel StartプロジェクトがAtmel Studio内にインポートされた後で完了されます。

### 7. 応用構成部品

応用の機能はシステム(クロックなど)だけなくADC周辺機能の理解も必要です。図2の赤部分はどのプロジェクトに対しても必要とされるシステムトライバを含みます。いくつかの構成設定はシステムが周辺機能に対して必要なクロックと構成設定を提供することができることを保証するために必要とされます。ADCもこの部分に含まれ、応用に対して必要とされる構成設定を転送するための評価が必要です。



### システムとADCのASFv3構成設定抽出

### 1. クロック構成設定記録

ASFv3用システム クロック設定は例プロジェクトのconf\_clocks.hファイルに置くことができます。ここから、主クロック(GCLK\_0)が内部8MHz発振器によって供給されるのを見ることができます。

図3.はGCLK\_0が前置分周されれず、8MHzでSAMD21コアを走らせることを示します。これらの設定はADCプロジェクトのクロック樹形を構成設定するためにAtmel Studioで使われます。このプロジェクトのADC周辺機能に対して、クロック元としてGCLK\_0が使われます。これらの値は走行時に一般クロック発生器(GCLK)とシステム制御(SYSCTRL)下でI/Oウィントウでも見つけることができます。

5 SAM	ID21_ADC_QUICK_START_CALLBACK		•
onf_clock	sh ≄ ×		
			- 44
58			
	/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - I	nternal 8MHz oscillator */	
	<pre># define CONF_CLOCK_OSC8M_PRESCALER</pre>	SYSTEM_OSC8M_DIV_1	
61	# define CONF_CLOCK_OSC8M_ON_DEMAND	true	
62	<pre># define CONF_CLOCK_OSC8M_RUN_IN_STANDBY</pre>	false	
63 20 % •			Þ
63 00 % • 5AM	AD21_ADC_QUICK_START_CALLBACK - conf_clocks.h		•
63 00 % • SAN onf_clock	AD21_ADC_QUICK_START_CALLBACK - conf_clocks.h		+ = - <
63 00 % • SAM onf_clock 133	AD21_ADC_QUICK_START_CALLBACK - conf_clocks.h		• • • •
63 00 % • SAM onf_clock 133 134	AD21_ADC_QUICK_START_CALLBACK - conf_clocks.h sh → × /* Configure GCLK generator θ (Main Clock) */		- *
63 00 % • SAM 0nf_clock 133 134 135	<pre>AD21_ADC_QUICK_START_CALLBACK - conf_clocks.h sh → ×  /* Configure GCLK generator 0 (Main Clock) */ # define CONF_CLOCK_GCLK_0_ENABLE</pre>	- true	□ ~ ¢(
63 00 % • SAM onf_clock 133 134 135 136	AD21_ADC_QUICK_START_CALLBACK - conf_clocks.h sh + × /* Configure GCLK generator θ (Main Clock) */ # define CONF_CLOCK_GCLK_θ_ENABLE # define CONF_CLOCK_GCLK_θ_RUN_IN_STANDBY	- true false	• • • <b>*</b>
63 00 % • SAM onf_clock 133 134 135 136 137	AD21_ADC_QUICK_START_CALLBACK - conf_clocks.h sh + × /* Configure GCLK generator 0 (Main Clock) */ # define CONF_CLOCK_GCLK_0_ENABLE # define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY # define CONF_CLOCK_GCLK_0_CLOCK_SOURCE	true false SYSTEM_CLOCK_SOURCE_OS	) - - (0 .008M
63 00 % • SAM onf_clock 133 134 135 136 137 138	AD21_ADC_QUICK_START_CALLBACK - conf_clocks.h sh = * /* Configure GCLK generator 0 (Main Clock) */ # define CONF_CLOCK_GCLK_0_ENABLE # define CONF_CLOCK_GCLK_0_ENABLE # define CONF_CLOCK_GCLK_0_CLOCK_SOURCE # define CONF_CLOCK_GCLK_0_PRESCALER	true false SYSTEM_CLOCK_SOURCE_OS 1	- ¢¢

### 2. 周辺機能構成設定記録

ASFv3の周辺機能単位部は応用に対して周辺機能を初期化するのに構成設定構造体を使います。ASFv3は初期化のために主辺機能に書かれる既定値を含みます。これは周辺機能構成設定構造体が走行時に移入されることに注意することが重要です。 ASFv4は違う手法を取り、前処理部マクロで全ての応用構成設定を処理します。

この例応用ではconfigure\_adc()関数内でADCを初期化するのにadc\_config構造体が使われます。プロジェクト実行中に監視(Watch) ウィントウを用い、初期化パラメータは図4で捕獲されます0。構成設定値の命名規則を維持するためにASFv4開発で努力が払われました。そんな訳で、図4の中央列内の値はAtmel StartでADCを構成設定するのに使われます。

ame		Value	Type
e con	ifig adc	ω	struct adc config
	clock source	GCLK GENERATOR 0	enum gclk generator
9	reference	ADC_REFERENCE_INTVCC1	enum adc_reference
0	clock_prescaler	ADC_CLOCK_PRESCALER_DIV8	enum adc_clock_prescaler
0	resolution	ADC_RESOLUTION_12BIT	enum adc_resolution
	gain_factor	ADC_GAIN_FACTOR_1X	enum adc_gain_factor
	positive_input	ADC_POSITIVE_INPUT_PIN18	enum adc_positive_input
9	negative_input	ADC_NEGATIVE_INPUT_GND	enum adc_negative_input
9	accumulate_samples	ADC_ACCUMULATE_DISABLE	enum adc_accumulate_samples
9	divide_result	ADC_DIVIDE_RESULT_DISABLE	enum adc_divide_result
91	left_adjust	false	_Bool
9	differential_mode	false	Bool
e 1	freerunning	false	_Bool
91	run_in_standby	false	_Bool
91	reference_compensation_enable	false	_Bool
9	sample_length	0x00	uint8_t
Þ 🥥 (	window		struct adc_window_config
Þ 🤗 (	correction		struct adc_correction_config
9	event_action	ADC_EVENT_ACTION_DISABLED	enum adc_event_action
Þ 🥥 1	pin_scan		struct adc_pin_scan_config

図5のADCレシ、スタはASFv3で初期化後のレシ、スタ値を 表示し、Atmel StartでADC初期化を確認するのに使 われます。

3. ピン構成設定記録

選んだ例プロジェクトは応用に対して1つのADCチャネルを 使うだけです。この場合、ピン割り当ては見つけるのが 容易で、前に参照されたconfig\_adc構造体またはconfi gure\_adc()関数に配置されます。Xplained Proのような Atmelハートウェアを使うASFv3例プロジェクトはハートウェアが system\_board\_init()関数で初期化される場所でこの情 報を持つかもしれません。

### 図5. ASFv3 ADCレジスタ値

Filter:		•	4
Name	Address	Value	Bits
CTRLA	0x42004000	0x02	
🛚 🛅 REFCTRL	0x42004001	0x02	
🖬 📄 AVGCTRL	0x42004002	0x00	
SAMPCTRL	0x42004003	0x00	
🛚 📄 CTRLB	0x42004004	0x0100	
WINCTRL	0x42004008	0x00	
🛚 🛅 SWTRIG	0x4200400C	0x00	
INPUTCTRL	0x42004010	0x00001806	
🗉 🔽 EVCTRL	0x42004014	0x00	
🛚 🗾 INTENCLR	0x42004016	0x00	
🛛 🏹 INTENSET	0x42004017	0x00	
🛚 🌄 INTFLAG	0x42004018	0x08	
🛛 🛅 STATUS	0x42004019	0x00	
🗉 📄 RESULT	0x4200401A	0x0000	
🛛 📄 WINLT	0x4200401C	0x0000	
🛚 🛅 WINUT	0x42004020	0x0000	
GAINCORR	0x42004024	0x0000	
OFFSETCO	0x42004026	0x0000	
🖬 🗈 CALIB	0x42004028	0x0388	
DBGCTRL	0x4200402A	0x00	
ocals 1/0 Watch 1	Watch 2 So	Jution Explore	ər

### Atmel START - システム初期化

1. 図6で示されるように、start.atmel.comに対して閲覧してその後にCreate New Project(新規プロジェクト作成)タブをクリックすることに よって新しいAtmel Startプロジェクトを作成してください。



2. Create New Projects(新規プロジェクト作成)ウィンドウで、Results(結果)項目下でSAM D21 Xplained Proを探してSAM D21 Xplained Proを選び、その後にCreate New Project(新規プロジェクト作成)をクリックしてください。図7.をご覧ください。

Atmel START					🗲 Return To Fi	ront Page	Help And Suppor
REATE NEW PROJECT							
ect device or board before creating a	new project. You can filte	er devices and boards	by what software you nee	d and also with hardwa	re requirements suc	h as memory sia	DES.
FILTERS		RESULTS					
HARDWARE	CO	SAM D21 Xplained	iPro 🛞 😡	Show all O Show	only boards O Sh	ow only devices	
Architecture	~	Name	Architecture	Package	Pins	Flash	SRAM
Flash size 0 8 0	• 2 MB 0	SAM D21 X	slained Pro				
RAM size 2 KB 0	• 512 KB 0						
hins 14 0 .	• 144 0						
Package All	~						
Atmel Data Protocol							
+ BLE							
Cloud							
Crypto Authentication Beta	0 0						
Ethernet PHY							
	•						
DRIVERS	() ()						
ADC							
		1 of 326 boards and	d devices		CREATE N	EW PROJECT	

3. 図8.で示されるように新規プロジェクトのDASHBOARD(指標表示板)が表示されます。

図8. 新規プロジェクト指標表示版	
Atmel START atsamd21j18a	+ Return To Front Page   Help And Support
{ }     view code     Save configuration	EXPORT PROJECT
MY SOFTWARE COMPONENTS	۲
Add software component       Add software component       Image: Second se	Show dependencies ① Show system drivers ① Show hardware ①
SELECTED BOARD: CUSTOM BOARD  You are using a custom board. Predefined boards can be selected from the front page when you create a new project.  SELECTED DEVICE: ATSAMD21J18A	

ASFv3でのように全てのASFv4プロジェクトでいくつかのシステムト・ライハ、がインクルートされることに注意してください。これらのト・ライハは開始構成設定で初めは非表示です。自動的にインクルート、されたシステムト、ライハ、、クロックを見るにはShow system drivers(システムト・ライハ、表示)指示部をクリックしてください。図9.で示されるように、4つの構成部品(SYSCTRL、DMAC、PM、GCLK)はDASHBOARD(指標表示板)画面でクロック、ハ、ス、NVM、DMAの設定の構成設定を許します。

	Atmel STAI	RT ATSAMD21J18A				🕂 Return To I	ront Page   Help And Support
	{}	VIEW CODE	ICt a		FIGURATION	D	EXPORT PROJECT
	MY SOFTWARE C	OMPONENTS					0
разнволер	Application Middleware Driver System driver			Add software c	omponent		Show dependencies D Show system drivers C Show hardware D
	F			MY PROJEC	* *		

- 4. DASHBOARD(指標表示板)タブで、プロジェクトに周辺機能単位部を追加するためにAdd software component(ソフトウェア部品追加)を クリックしてください。Add Software Components(ソフトウェア部品追加)タイアログは図10.で示されるようにSAMD21に対して利用可なトラ イバを表示します。
- 5. 全てのドライバを表示するためにName(名前)列の"+"をクリックし、その後にAdd(追加)列の"+"シンボルをクリックすることによってADCを 選んでください。Selected Components(選んだ部品)項目下にADC単位部が表示されます。
- 6. 選んだ部品をプロジェクトに追加するためにAdd Componet(s)(部品追加)をクリックしてください。

ilter				
Name	Description			Add
+ 🕏 Middleware				
- O Drivers				
O AC	Analog Comparator (AC).			$\oplus$
O ADC	Analog-to-digital converter (A	ADC).		$\oplus$
Analog Glue Function	No description available			Ð
CAN	No description available			( <del>+</del> )
SELECTED COMPONENTS				
Name		Count	Add/remove	Remove al
O ADC		1	$\odot \oplus$	Ē

7. 周辺機能単位部を構成設定してください。一旦ADC単位部が追加されると、図11.で示されるようにそれが(画面中央の)MY PRO JECT(私のプロジェクト)タブ下に表示されます。

図11.	周辺機能単	位部構成設定		
	Atmel S	TART atsamd21j18a		← Return To Front Page   Help And Support
		<pre>{ } view code</pre>	SAVE CONFIGURATION	EXPORT PROJECT
	MY SOFTWAR	RE COMPONENTS		۲
🕎 разнвоакр	Application     Middleware     Driver     System driver     Hardware		Add software component	Show dependencies ① A Show system drivers ① Show hardware ①
XNMNIA			ADC_0	
C) CLOCKS		GCLK	SYSCTRL O	DMAC 🔅

8. ADC\_0タブをクリックし、その後に構成設定任意選択を移入するためにASFv3例プロジェクトでadc\_config構造体から収集した情報を使い、その後に図12.で示されるようにADC入力に対してPA06(AIN/6)を許可してくだださい。

义	12.	ADC周辺機能構成設定(1/2)						
Γ		MY SOFTWARE COMPONENTS					0	Ð
	<b>VSHBOARD</b>		Analog to digita	ADC_0 I converter (ADC) in asynchronous mode using inte	rrupts		0	31
	a nu	GENERAL	COMPONEN	IT SETTINGS	SIGNALS			
	or i	i User guide	Driver:	HAL:Driver:ADC_Async ~	AIN/0:	PA02	 ~	
I	š		CLOCKS		AIN/1:	PA03	 $\sim$	
I	INNI	Kename component	ADC:	Generic clock generator 0 (8 MHz)	AIN/2:	P808	 $\sim$	
	õ	Remove component			AIN/3:	P809	 v	
Ľ					AIN/4:	PA04	 $\sim$	
I	ย				AIN/5:	PA05	 ~	
I	CLOC				AIN/6:	PA06	 ~	
	ē				AIN/7:	PA07	 ~	
					AIN/8:	P800	 ~	

図13.	ADC周辺機能構成該	设定(2/2)			
	MY SOFTWARE COMPO	NENTS			0
ARD	BASIC CONFIGURATION		ADVANCED CONFIGURATION		Enable: 🗸 📩
BBHB	Conversion resolution:	0 12-bit V	Run in standby:		
M N	Reference Selection:	1/2 VDDANA (only for VDDANA > 2.0V)	Debug Run: (		
B	Prescaler configuration:	Peripheral clock divided by 8     V	Left-Adjusted Result: (		
	Free Running Mode:	•	Reference Buffer Offset		
ğ	Differential Mode:	•	Digital Correction Logic Enabled:		
NN N	Positive Mux Input Selection:	ADC AIN6 pin      V	Offset Correction Value:	0	0
	Negative Mux Input Selection:	Internal ground ~	Gain Correction Value:	0	0
	EVENT CONTROL	Enable:	Gain Factor Selection:	0 1x	~
			Adjusting Result / Division Coefficient:	0	0
SC SC			Number of Samples to be Collected:	1 sample	~
E E			Sampling Time Length:	0	0
$\odot$			Window Monitor Mode:	No window mode	Ý
			Window Monitor Lower Threshold:	0	0
			Window Monitor Upper Threshold:	0	0
			Number of Input Channels Included in Scan:	0	0
			Positive Mux Setting Offset:	0	0
					_

### 9. ピン配置構成設定

ASFv3プロジェクトから情報を収集する時に、ADCはこのプロジェクトで1つのピンだけを使い、それはDASHBOARD(指標表示板)上の ADC構成設定で構成設定されます。けれども、汎用入出力(GPIO)やピンの名前付けはPINMUX構成設定部で行われます。図14. をご覧ください。PINMUX構成設定部で変更することを望むピンを選ぶことにより、パラメータを設定するために画面の下近くに使用 者ウィントウが開きます。



### 10. クロック樹形構成設定

クロック樹形はStartツールで、システムトライバ単位部を通したDASHBOARD(指標表示板)と、クロック構成設定部の2つの場所で構成設定することができます。クロック構成設定部はクロック樹形の図画的な表現を提供し、DASHBOARD(指標表示板)と同じ構成設定能力を提供します。この例についてはクロック構成設定部が使われます。

11. 図15.で示されるように、プロジェクトをエクスホートするため、Atmel Start画面の右上部分のExport Project(プロジェクトをエクスホート)タブを クリックすることによってプロジェクトをエクスポートしてください。これがこのエクスポートしたプロジェクトの使用を意図される唯一のIDEのため、 Atmel Studioだけが選ばれるべきです。Atmel Startからの出力はatzipファイルです。

図15.	図15. Atmel START - プロジェクトのエクスポート						
	Atmel START	🗲 Return To Front Page 📋 Help And Support					
	{ } VIEW CODE	CONFIGURATION					
	EXPORT PROJECT						
🕓 CLOCKS 🌐 PINMUX 🕎 DASHBOARD	DOWNLOAD YOUR CONFIGURED PROJECT         Download a generated pack containing all your configured software components.         Select which IDE or command line tool you want the pack to include support files for:         Atmel Studio:       IMR Embedded Workbench:         µVision from Kell:       Makefile (standalone):         Specify file name (optional):       My Project         Image: DOWNLOAD PACK       Image: Download pack	WHAT TO DO NEXT?         Use one of the selected IDEs and import your project as described in the user guide.         If you do not have any IDEs installed you can download and install <u>Atmel Studio 7.0</u> for free.         Note: The exported pack can also be used later if you want to import a configuration in Atmel START         How to open in IDEs					

### システム構成設定の確認

1. Atmel StudioでFile(ファイル)⇒Import(インホート)⇒Atmel Start Project(Atmel STARTプロジェクト)を選ぶことによってエクスホートしたatzip ファイルをインポートしてください。

tmel Start Importer		
Import Atmel Star	rt Project	
Atmel Start Project(.atzip):	C:\Downloads\StartProjects\SAMD21_START_ADC_Example	Browse
View project summary (CMSI	S package information)	
Project Name:	SAMD21_START_ADC_Example	
Location:	C:\Atmel Studio\7.0	Browse
Solution:	Create New Solution	×.
Solution Name:	SAMD21_START_ADC_Example	
View project import summary		
		OK Cancel

Atmel Startでのパラメーター式は都合の良いことにシステム初期 化に対してはconfigフォルダ、ピン割り当てに対してはatmel\_star t\_pins.h、周辺機能初期化に対してはdriver\_init.c/driver\_init. hに置かれます。図17をご覧ください。

- Config 前処理部デバイス構成設定に使われるAtmel Start からのレジスタ設定を含みます。
- atmel\_start.c MCU、ドライバ、ミドルウェアを初期化します。
- atmel\_start\_pins.h Atmel Start構成設定からMCUピン割り 当てと命名を保持します。
- driver\_init.c 周辺機能用の初期化関数を含みます。
- main.c システムを初期化するためにatmel\_start\_init()を呼びます。

### 図17. ASFv4 7ァイル構造



2. プロジェクトの構築と走行

新しいプロジェクトがどの応用コートも含まないため、Atmel Startはsystem\_init()で定義された周辺機能に対する初期化コートを出力します。入出力(I/O)ウィントウでレジスタ設定を見るためにプロジェクトを走らせて初期化手順後に停止してください。

入出力(I/O)ウィント・ウは入出力(I/O)アイコンをクリックすることによってAtmel Sudioで開くことで利用可能です。



3. 入出力(I/O)ウィンドウを開いてA/D変換器を選び、ASFv3から収集したそれらの結果を比較することによってASFv3とASFv4間のレジスタ設定を比較してください(図18.をご覧ください)。

ASFv3				ASFv4					
			• <del>•</del> ×	1/0			• 1		
Filter:		•	4	Filter:		- 1	4		
Name	Address	Value	Bits	Name	Address	Value	Bits		
CTRLA	0x42004000	0x02		🗖 🛅 CTRLA	0x42004000	0x02			
REFCTRL	0x42004001	0x02		REFCTRL	0x42004001	0x02			
AVGCTRL	0x42004002	0x00		AVGCTRL	0x42004002	0x00			
SAMPCTRL	0x42004003	0x00		SAMPCTRL	0x42004003	0x00			
CTRLB	0x42004004	0x0100		CTRL8	0x42004004	0x0100			
MINCTRL	0x42004008	0x00		WINCTRL	0x42004008	0x00			
SWTRIG	0x4200400C	0x00		SWTRIG	0x4200400C	0x00			
INPUTCTRL.	0x42004010	0x0F001806		🗖 🗈 🗈 INPUTCTRL	0x42004010	0x0F001806			
EVCTRL	0x42004014	0x00		EVCTRL	0x42004014	0x00			
INTENCLR	0x42004016	0x00		INTENCLR	0x42004016	0x01			
INTENSET	0x42004017	0x00		🗖 🎑 INTENSET	0x42004017	0x01			
INTFLAG	0x42004018	0x08		🖬 🛃 INTFLAG	0x42004018	0x08			
TATUS	0x42004019	0x00		STATUS	0x42004019	0x00			
RESULT	0x4200401A	0x0000		RESULT	0x4200401A	0x0000			
MINET .	0x4200401C	0x0000	*******	WINLT	0x4200401C	0x0000	******** *******		
WINUT	0x42004020	0x0000		WINUT	0x42004020	0x0000	*******		
GAINCORR	0x42004024	0x0000		GAINCORR	0x42004024	0x0000			
OFFSETC	0x42004026	0x0000		OFFSETC	0x42004026	0x0000			
CALIB	0x42004028	0x0388		CALIB	0x42004028	0x0000			
DBGCTRL	0x4200402A	0x00		DBGCTRL	0x4200402A	0x00			

前の比較が行われる時に2つの違いが直ぐに明らかになります。それはADC割り込みが許可される場所とADCの校正設定です。 ASFv3では割り込みがADC作業関数内で許可される一方で、ASFv4は呼び戻し関数が割り当てられた時に割り込みを許可します。 加えて、ASFv4のADCトライバはADCに対して工場校正設定をインポートしません。これは使用者によって行われなければなりません。 この校正はもっと正確なADC測定値に対する偏りと直線性の設定を提供します。追加情報についてはSAM D21系統データシート(DS4 0001882)の「9.3.2. NVMYフトウェア校正領域割り当て」項を参照してください。ASFv47°Rシ´ェクトに構成をインポートするには「追補A:応用 コート・例」を参照してください。

### 応用の変換

### 1. 応用説明

ADCは初期化後に開始され、128個の12ビット試料を収集してそのデータを緩衝部に格納します。試料が収集されると、ADC呼び 戻しに移行して採取の完了を示すフラグが設定されます。

### **2**. API比較

ADC単位部の使用はAtmel Start構成設定で提供された以外にどんな追加の構成設定も必要としません。ASFv3は提供された既定の組と非常に似ていますが、初期化関数を作成して既定デルクトリを変更するのに開発者を必要とします。

例プログラムは定義した試料数で緩衝部を満たすのにadc\_read\_buffer\_jobを使います。この機能はASFv4での直接一致がありません。ASFv4でadc\_read\_buffer\_jobに最も近いのはadc\_async\_buffer\_start\_conversionです。けれども、これら2つの関数はかなり違う結果を生じます。ASFv3ではADC\_adc\_interrupt\_handlerが作業を完了させるように構成設定され、割り当てた緩衝部に要求した試料数でデータを入れます。ASFv4のADC処理部はもっと一般的な手法を取るように設計され、呼び戻して追加の採取を扱うために開発者を必要とします。ASFv3の機能に合わせるため、ASFv4のADC呼び戻しは配列内にADC結果を置いて次のADC作業を開始します。

完全な構成設定コートについては「追補A:応用コート」例」を参照してください。

### 3. 呼び戻し構成設定

一般的な応用に関して、主プロジェクト樹形内の例フォルダはプロジェクトで許可された周辺機能を構成設定するコードを含みます。この場合、例フォルダは応用に対して呼び戻しを許可するのに必要とされる全てのものを含みます。

ADC呼び戻しはASFv3とASFv4で同様に構成設定されます。違いは割り込みが合図された時のADC割り込み処理関数の方法です。ASFv3とASFv4間の違いを見るには\_adc\_interrup\_handlerを検索してください。ASFv4は低い付随負荷を提供し、呼び戻しで特別な機能を扱うことを開発者に許します。

完全な構成設定コートについては「追補A:応用コート」例」を参照してください。

## プロジェクトの拡張

次のようにASFv3プロジェクトに計時器制御(Timer Control)単位部を追加してください。

### 1. 応用説明

更新した応用はLEDに接続されたPWMチャネルのデューティサイクルを更新するのにAD Cの呼び戻しを使います。デューティサイクルはADCから読んだ値を用いて計算されま す。加えて、ADC値の塊を収集する単一作業を持つよりもむしろADCは100ms毎に PWMチャネルのデューティサイクルを更新するための継続的な繰り返しで開始されます。



- 2. プロジェクトの新しい機能を達成するにはASFウィサートから遅延サービスとTCC0トライハー単位部が追加されなければなりません。
- 3. ASFv3プロジェクトを開いて計時器制御(Timer Control)単位部を追加してください。

Atmel Studioに於いてProject下でASF Wizardを選んでください。一旦ウィンドウが開いたなら、変更されるべき例プロジェクトがプロジェクト引き落としメニューに入れられているのを確認してください。

Available Modules(利用可能な単位部)メニューで選択した呼び戻し(callback)任意選択と共にTC – Timer Counter (driver)(TC – 計時器/計数器(ドライベ))とDelay services(遅延サービス)単位部を選んでください。ウィントウの下部に配置されたApply(適用)をクリック することによって追加した単位部をプロジェクトに適用してください。TCドライハ゛と遅延サービスの両方がSelect Modules(単位部選択)メニューで表示されます。

ASFウィザート、によって一旦プロジェクトにTCCが追加されると、図20.で示されるようにプロジェクトでtcch、ライハ、フォルタを見つけることができ、応用で使うために適切なヘッタ、ファイルがasf.hに追加されます。



### ASFv3計時器構成の構成設定

### 1. 計時器構成の構成設定

プロジェクトがXplained Proで構築されている ため、PWM出力として使うために基板上の LEDにいくつかの定義が割り当てられていま す。この定義はプロジェクト内のsamd21\_xplain ed\_pro.hファイルに含められます。図21.をご覧 ください。

samd21\_xplained\_pro.hで定義されたマクロを 用いて図22.で示されるようにconfigure\_t cc 関数を作成してください。tcc\_instanceと呼ば れるt cc\_module構造体の実体を作成してく ださい。configure\_tcc関数の最初の行の1つ

# 図21. Xplained Pro定義

が既定TCC単位部の構成設定用の値一式を得るための関数呼び出しであることに注目してください。ASFv3は各周辺機能用の 既定設定一式を含みます。これらの既定はこのプロジェクト用の基準として使われ、必要とされる場所で変更されます。

### 2. TCC0レジスタ設定入手

ー旦TCC単位部が構成設定されてしまうと、Atmel Studioでコンパイルして走らせてください。ADCに対して前に入出力(I/O)ウィンドウを用いて行ったようにレジスタ設定を取り込んでください。これらの値はAtmel Startからの出力と比較するのに使われます。

### Atmel START - 計時器の追加

1. Atmel Startホーム頁を閲覧してLoad existing project(既存プロジェクト読み込み)任意選択を選ぶことによってAtmel Startプロジェクトを インポートしてください。その後、ダウンロートしたatzipファイルを検索してOPEN SELECTED FILE(選んだファイルを開く)をクリックしてくださ い。



2. ADD SOFTWARE COMPONENTS(ソフトウェア部品追加)をクリックし、その後にウィンドウでPWM部品を選ぶことによってプロジェクトに周辺機能単位部を追加してください。共にTCC単位部を使うとは言え、PWM部品はTCC部品と異なることに注意してください。

ADD SOFTWARE COMPONEN	NTS					
Filter						
Name	Description					
O PWM	Pulse-width modulation (PWM) digitally by controlling the amo connected peripheral.	Pulse-width modulation (PWM) to create an analog behavior digitally by controlling the amount of power transferred to the connected peripheral.				
O SPI	Serial Peripheral Interface (SPI)	Serial Peripheral Interface (SPI), synchronous serial				
SELECTED COMPONENTS						
Name		Count	Add/remove	Remove	all	
<b>A</b>		· · · ·	00	m		

3. 図25.と図26.で示されるように、新しく追加したPWM\_0単位部を選んで構成設定するためにASFv3プロジェクトから収集した値を用いることによって周辺機能単位部を構成設定してください。

凶25	. Atmel START - PWM構成設	定(1/2)							
	MY SOFTWARE COMPONENTS							0	
ASHBOARD	PWM_0 PWM functionality using a TCC peripheral								
	GENERAL	COMPONENT	T SETTINGS		SIGNALS				
Car.	1 User guide	Driver:	HAL:Driver:PWM	~	WO/0:	P830	PWM out		
×		Instance:	TCC0	~	WO/1:		~	~	
INMU	Rename component	CLOCKS			W0/2:		v	~	
à.	Remove component	TCC:	Generic clock generator 0 (8 MHz)	~	WO/3:		~	~	
					WO/4:		~	~	
					W0/5:		~	~	
SOC					W0/6:		~	~	
E					WO/7:		~		
မ									

	MY SOFTWARE COMPONI	ENTS			0
g	BASIC SETTINGS		PWM WAVEFORM OUTPUT SETTINGS		-
180 €	TCC0 Prescaler:	Divide by 8 🗸	TCC0 Waveform Period Value (uS):	Oxff	
DASI	TCC0 Period Value:	0xff	TCC0 Waveform Duty Value (0.1%):	0x7f	
Ð	ADVANCED SETTINGS		TCC0 Waveform Channel Select:	0x0	
	Run in standby:	· 🗌			
MUX	TCC0 Prescaler and Counter Synchronization Selection:	Reload or reset counter on next GCLK			
2	TCC0 Waveform Generation Selection:	Single-slope PWM V			
	TCC0 Auto Lock:	·			
	TCC0 Capture Channel 0 Enable:				
~	TCC0 Capture Channel 1 Enable:				
Š	TCC0 Capture Channel 2 Enable:	•			
5	TCC0 Capture Channel 3 Enable:	•			
$\Theta$	TCC0 Lock update:	•			
	TCC0 Debug Running Mode:	• 🗌			

### 4. クロック樹形の構成設定

このプロジェクトの初版は一般クロック生成器0(GCLK\_0)へ行く8MHzを持っていました。ADC\_0とWPM\_0の両方がGCLK\_0によってクロック信号を供給されることを確実にしてください。PWM\_0部品がGCLK\_0によって供給されない場合、COMPONENTS(部品)部の 設定アイコンをクリックしてTCCクロック元をGLK\_0に設定してください。

### 図27. Atmel START - TCC0クロック構成設定部 CLOCK CONFIGURATOR 1 Reset clock settings OSCILLATORS SOURCES External Crystal Oscillator 0.4-32MHz (XOSC) Generic clock generator 0 CPU ¢ Frequency: 8 MHz Frequency: 8 MHz ۵ 32kHz External Crystal Oscillator (XOSC32K) 0 COMPONENTS ③ 0 Generic clock generator 1 PWM\_0 32kHz High Accuracy Internal Oscillator (OSC32K) ۶. TCC 8 MHz ¢ Using: TCC0 Generic clock generator 2 8MHz Internal Oscillator (OSC8M) ADC.0 0 ≥-Generic clock generator 3 \$ Frequency: 8 MHz ADC 8 MHz ¢ Using: ADC 0 Digital Frequency Locked Loop (DFLL48M) Generic clock generator 4 C) CLOCKS 0 Fractional Digital Phase Locked Loop (FDPLL96M) Generic clock generator 5 32kHz Ultra Low Power Internal Oscillator (OSCULP32K) Generic clock generator 6 ۵ Frequency: 32.768 kHz Generic clock generator 7

### システム構成設定の確認

- 1. 以前に記述されたようにプロジェクトをインポートして構築することによってASFv3とASFv4間のレジスタ設定を比較してください。
- 2. TCC0周辺機能が初期化されて許可されてしまうまでプロジェクトを走らせてください。図28.はASFv3とASFv4間の初期化の違いを 表示します。図28.での比較から、TCC0単位部の初期化はASFv4で構成設定されます。



## 応用の更新

1. 応用説明

PWMデューティサイクル計算用の入力を提供するにはADC呼び戻しでの算法が実装されることだけが必要です。ADCは単一12ビット 試料収集に構成設定され、その後に呼び戻しを起動します。ADC呼び戻しでは、ADCの結果(RESULT)レジスタが直接読まれて8 ビットの結果を提供するために位置移動されます。この8ビット値はPWMのデューティサイクルに書かれます。PWMは8ビット周期を持つ ように構成設定され、故にデューティサイクルは適切に尺度調整されます。

### 2. API比較

TCC0単位部の使用はAtmel Start構成設定から提供されたもの以外にどんな追加の構成設定も必要としません。ASFv3は既定の一式で提供されたものと非常に似ていますが、初期化関数を作成して既定を直接変更するために関発者を必要とします。

ASFv4は枠組み全体を通してより多くの使用事例で駆動される模式を持ちます。これはTCC部品がASFv3で追加され、PWM部品がAt mel startで追加された時から見ることができます。ASFv4でのPWM部品用APIはpwm\_set\_parametersでこれをもっと良く表示します。ASFv3は同じ機能を達成するのにtcc\_set\_compare\_valueを使います。これらの関数で渡されるパラメータは劇的に違いませんが、ASFv4で達成される事例駆動手法を用いて関数を掘り下げてもっと薄くします。

### 3. 呼び戻し構成設定

このプロジェクトのTCC0単位部は呼び戻しを使いませんが、ADC呼び戻し関数でデューティサイクルが変更されます。

## 結び

ASFv4はコード量と応用駆動使用事例手法でのドライバ効率性での改良を含む新しい枠組みです。ASFv4の基本構造がASFv3のものと異なる一方で、デバイスに関する命名規則と基本的な側面は残ります。

ASFv3からASFv4への変換に取り組むための予め準備された解決策はありません。移植は0から完全に生じなければなりません。 この資料が2つの機能的な違いだけを網羅した一方で、機能的な移植を達成するために基本的な原則が提供されます。

```
追補A:応用コード例
例1:ASFv3プロジェクト拡張 - ADCとPWM
#define ADC_SAMPLES 1
uint16_t adc_result_buffer[ADC_SAMPLES];
struct adc_module adc_instance;
struct tcc_module tcc_instance;
volatile bool adc_read_done = false;void adc_complete_callback(struct adc_module *const module)
{
    uint32_t duty;
    duty = (adc_result_buffer[0] >> 4) & 0xFFFF;
    tcc_set_compare_value(&tcc_instance, (enum tcc_match_capture_channel)(CONF_PWM_CHANNEL), duty);
void configure_adc(void)
    struct adc_config config_adc;
    adc_get_config_defaults(&config_adc);
    config_adc.clock_prescaler = ADC_CLOCK_PRESCALER_DIV8;
    config_adc.reference
                               = ADC_REFERENCE_INTVCC1;
    config_adc.positive_input = ADC_POSITIVE_INPUT_PIN18;
    config_adc.resolution = ADC_RESOLUTION_12BIT;
    adc init(&adc instance, ADC, &config adc);
    adc_enable(&adc_instance);
static void configure_tcc(void)
    struct tcc_config config_tcc;
    tcc_get_config_defaults(&config_tcc, LED_0_PWM4CTRL_MODULE);
    config_tcc.counter.clock_prescaler = TCC_CLOCK_PRESCALER_DIV8;
    config_tcc.counter.period = 0xFE;
    config_tcc.compare.wave_generation = TCC_WAVE_GENERATION_SINGLE_SLOPE_PWM;
    config_tcc.compare.match[LED_0_PWM4CTRL_CHANNEL] = 0x7F;
    config_tcc.pins.enable_wave_out_pin[LED_0_PWM4CTRL_OUTPUT] = true;
    config_tcc.pins.wave_out_pin[LED_0_PWM4CTRL_OUTPUT]
                                                               = LED 0 PWM4CTRL PIN;
    config_tcc.pins.wave_out_pin_mux[LED_0_PWM4CTRL_OUTPUT]
                                                               = LED_0_PWM4CTRL_MUX;
    tcc_init(&tcc_instance, LED_0_PWM4CTRL_MODULE, &config_tcc);
    tcc_enable(&tcc_instance);
void configure_adc_callbacks(void)
    adc_register_callback(&adc_instance, adc_complete_callback, ADC_CALLBACK_READ_BUFFER);
    adc_enable_callback(&adc_instance, ADC_CALLBACK_READ_BUFFER);
int main(void)
ł
    system_init();
    delay_init();
    configure_adc();
    configure_adc_callbacks();
```

configure\_tcc(); system\_interrupt\_enable\_global(); while (1) { adc\_read\_buffer\_job(&adc\_instance, adc\_result\_buffer, ADC\_SAMPLES); delay\_cycles\_ms(100); }

## 例2:ASFv3移植 - ADCのみ

```
void adc_complete_callback(const struct adc_async_descriptor *const descr, const uint8_t channel)
ł
    static uint8_t i = 0;
    if (i < ADC_SAMPLES)
        adc_result_buffer[i++] = ADC->RESULT.reg;
        adc_async_start_conversion(&ADC_0);
    }else
        adc read done = true;
int main(void)
    system_init();
    adc_async_register_callback(&ADC_0, 0, ADC_ASYNC_CONVERT_CB, adc_complete_callback);
    adc_async_enable_channel(&ADC_0, 0);
    adc_async_start_conversion(&ADC_0);
    while (adc_read_done == false)
    ł
        /* 非同期ADC読み込み完了待機 */
    while (1)
        asm("nop");
```

### 例3:プロジェクト拡張 - ADCとPWM

```
#define PWM_PERIOD 254
static uint32_t pwm_duty;
static uint16_t adc_value;
static void adc_cb(const struct adc_async_descriptor *const descr, const uint8_t channel)
{
    adc_value = ADC->RESULT.reg;
    pwm_duty = (adc_value >> 4) & 0xFF;
    pwm_set_parameters(&PWM_0, PWM_PERIOD, pwm_duty);
}
int main(void)
{
    atmel_start_init();
}
```

```
adc_async_register_callback(&ADC_0, 0, ADC_ASYNC_CONVERT_CB, adc_cb);
ADC->CALIB.reg = ADC_CALIB_BIAS_CAL((*(uint32_t *)ADC_FUSES_BIASCAL_ADDR >>
ADC_FUSES_BIASCAL_Pos)) |
ADC_CALIB_LINEARITY_CAL((*(uint64_t *)ADC_FUSES_LINEARITY_0_ADDR >>
ADC_FUSES_LINEARITY_0_Pos));
adc_async_enable_channel(&ADC_0, 0);
```

adc\_async\_start\_conversion(&ADC\_0);delay\_ms(100);

while(1)

{

}

### Microchipデバイスでの以下のコード保護機能の詳細に注意してください。

- ・Microchip製品はそれら特定のMicrochipデータシートに含まれる仕様に合致します。
- ・Microchipは意図した方法と通常条件下で使われ使われる時に、その製品系統が今日の市場でその種類の最も安全な系統の1つであると考えます。
- コード保護機能を破るのに使われる不正でおそらく違法な方法があります。当社の知る限りこれらの方法の全てはMicrochipのデータシートに含まれた動作仕様外の方法でMicrochip製品を使うことが必要です。おそらく、それを行う人は知的財産の窃盗に関与しています。
- ・Microchipはそれらのコードの完全性について心配されているお客様と共に働きたいと思います。
- ・Microchipや他のどの半導体製造業者もそれらのコートの安全を保証することはできません。コート、保護は当社が製品を"破ることができない"として保証すると言うことを意味しません。

コート、保護は常に進化しています。Microchipは当社製品のコート、保護機能を継続的に改善することを約束します。Microchipのコート、保 護機能を破る試みはデジタル シニアム著作権法に違反するかもしれません。そのような行為があなたのソフトウェアや他の著作物に不正な アクセスを許す場合、その法律下の救済のために訴権を持つかもしれません。

デバイス応用などに関してこの刊行物に含まれる情報は皆さまの便宜のためにだけ提供され、更新によって取り換えられるかもしれま せん。皆さまの応用が皆さまの仕様に合致するのを保証するのは皆さまの責任です。Microchipはその条件、品質、性能、商品性、 目的適合性を含め、明示的にも黙示的にもその情報に関連して書面または表記された書面または黙示の如何なる表明や保証もし ません。Microchipはこの情報とそれの使用から生じる全責任を否認します。生命維持や安全応用でのMicrochipデバイスの使用は完 全に購入者の危険性で、購入者はそのような使用に起因する全ての損害、請求、訴訟、費用からMicrochipを擁護し、補償し、免責 にすることに同意します。他に言及されない限り、Microchipのどの知的財産権下でも暗黙的または違う方法で許認可は譲渡されま せん。

Microchipはその世界的な本社、アリゾナ州のチャントラーとテンヘ、オレゴン州グラシャムの設計とウェハー製 造設備とカリフォルニアとイントの設計センターに対してISO/TS-16949:2009認証を取得しました。当社 の品質システムの処理と手続きはPIC<sup>®</sup> MCUとdsPIC<sup>®</sup> DSC、KEELOQ<sup>®</sup>符号飛び回りデバイス、直列 EEPROM、マイクロ周辺機能、不揮発性メモリ、アナログ製品用です。加えて、開発システムの設計と製 造のためのMicrochipの品質システムはISO 9001:2000認証取得です。

### 商標

Microchipの名前とロゴ、Mcicrochipロゴ、AnyRate、AVR、AVRロゴ、AVR Freaks、BeaaconThings、BitCloud、CryptoMemory、CryptoR F、dsPIC、FlashFlex、flexPWR、Heldo、JukeBlox、KeeLoq、KeeLoqロゴ、Kleer、LANCheck、LINK MD、maXStylus、maXTouch、Med iaLB、megaAVR、MOST、MOSTロゴ、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32ロゴ、Prochip Designer、QTouch、Rig htTouch、SAM-BA、SpyNIC、SST、SSTロゴ、SuperFlash、tinyAVR、UNI/O、XMEGAは米国と他の国に於けるMicrochip Technology Incorporatedの登録商標です。

ClockWorks、The Embedded Control Solutions Company、EtherSynch、Hyper Speed Control、HyperLight Load、IntelliMOS、mTou ch、Precision Edge、Quiet-Wireは米国に於けるMicrochip Technology Incorporatedの登録商標です。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BodyCom、chipKIT、chipKITロゴ、C odeGuard、CryptoAuthentication、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、 DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet¤ ゴ、Mindi、MiWi、motorBench、MPASM、MPF、MPLAB Certifiedロゴ、MPLAB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PureSilicon、QMatrix、RightTouchロゴ、REAL ICE、Ripple Blocker、SAM-IC E、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、View Sense、WiperLock、Wireless DNA、ZENAは米国と他の国に於けるMicrochip Technology Incorporatedの商標です。

SQTPは米国に於けるMicrochip Technology Incorporatedの役務標章です。

Silicon Storage Technologyは他の国に於けるMicrochip Technology Inc.の登録商標です。

GestICは他の国に於けるMicrochip Technology Inc.の子会社であるMicrochip Technology Germany II GmbH & Co. KGの登録商 標です。

ここで言及した以外の全ての商標はそれら各々の会社の所有物です。

© 2017年、Microchip Technology Incorporated、米国印刷、不許複製

日本語© HERO 2021.

本応用記述はMicrochipのAN2474応用記述(DS00002474A-2017年10月)の翻訳日本語版です。日本語では不自然となる重複する 形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部 加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には()内に英語表記や略称などを残す形で表記しています。

青字の部分はリンクとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。



# 世界的な販売とサービス

### 本社

2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術支援: http://www.microchip.com/ support ウェブ アトレス: www.microchip.com

米国

**7トランタ** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455

オースチン TX Tel: 512-257-3370

**ボストン** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088

**シカゴ** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075

**ý 77** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924

**デトロイト** Novi, MI Tel: 248−848−4000

**ヒューストン** TX Tel: 281-894-5983

インデ<sup>・</sup>アナホ<sup>°</sup>リス Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380

**ロサンセ<sup>\*</sup>ルス** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 ローリー NC

Tel: 919-844-7510 **–––7** NY Tel: 631-435-6000

サンホセ CA Tel: 408-735-9110 Tel: 408-436-4270 カナダ・-トロント Tel: 905-695-1980

Tel: 905-695-1980 Fax: 905-695-2078 亜細亜/太平洋

**亜細亜太平洋支社** Suites 3707-14, 37th Floor Tower 6, The Gateway Harbour City, Kowloon 香港

Tel: 852-2943-5100 Fax: 852-2401-3431 オーストラリア - シト゛ニー

Tel: 61-2-9868-6733 Fax: 61-2-9868-6755 中国 - 北京

Tel: 86-10-8569-7000 Fax: 86-10-8528-2104

**中国 - 成都** Tel: 86-28-8665-5511 Fax: 86-28-8665-7889

**中国 - 重慶** Tel: 86-23-8980-9588 Fax: 86-23-8980-9500

中国 - 東莞 Tel: 86-769-8702-9880 中国 - 広州 Tel: 86-20-8755-8029

中国 - 杭州 Tel: 86-571-8792-8115 Fax: 86-571-8792-8116 中国 - 香港特別行政区

Tel: 852-2943-5100 Fax: 852-2401-3431 **中国 - 南京** 

Tel: 86-25-8473-2460 Fax: 86-25-8473-2470 中国 - 青島

Tel: 86-532-8502-7355 Fax: 86-532-8502-7205

**中国 - 上海** Tel: 86-21-3326-8000 Fax: 86-21-3326-8021

**中国 - 瀋陽** Tel: 86-24-2334-2829 Fax: 86-24-2334-2393

**中国 - 深圳** Tel: 86-755-8864-2200 Fax: 86-755-8203-1760

中国 - 武漢 Tel: 86-27-5980-5300 Fax: 86-27-5980-5118 中国 - 西安

Tel: 86-29-8833-7252 Fax: 86-29-8833-7256 亜細亜/太平洋

中国 - 廈門 Tel: 86-592-2388138 Fax: 86-592-2388130 中国 - 珠海 Tel: 86-756-3210040 Fax: 86-756-3210049 イント・ーハンガロール Tel: 91-80-3090-4444 Fax: 91-80-3090-4123 イント - ニューテリー Tel: 91-11-4160-8631 Fax: 91-11-4160-8632 イント・ファネー Tel: 91-20-3019-1500 日本 - 大阪 Tel: 81-6-6152-7160 Fax: 81-6-6152-9310 日本 - 東京 Tel: 81-3-6880-3770 Fax: 81-3-6880-3771 韓国 - 大邱 Tel: 82-53-744-4301 Fax: 82-53-744-4302 韓国 - ソウル Tel: 82-2-554-7200 Fax: 82-2-558-5932 or 82-2-558-5934 マレーシア - クアラルンプール Tel: 60-3-6201-9857 Fax: 60-3-6201-9859 マレーシア ー ヘ・ナン Tel: 60-4-227-8870 Fax: 60-4-227-4068 フィリピン - マニラ Tel: 63-2-634-9065 Fax: 63-2-634-9069 シンガホール Tel: 65-6334-8870 Fax: 65-6334-8850 台湾 - 新竹 Tel: 886-3-5778-366 Fax: 886-3-5770-955 台湾 - 高雄 Tel: 886-7-213-7830 台湾 - 台北 Tel: 886-2-2508-8600 Fax: 886-2-2508-0102

Fax: 886-2-2508-0102 **91 - ה`גם'** Tel: 66-2-694-1351 Fax: 66-2-694-1350

### 欧州

オーストリア – ウェルス Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 テンマーク - コヘンハーケン Tel: 45-4450-2828 Fax: 45-4485-2829 フィンラント – エスホー Tel: 358-9-4520-820 フランス – パリ Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 フランス - サン クルー Tel: 33-1-30-60-70-00 トイツ – カ・ルトンク・ Tel: 49-8931-9700 ドイツ - ハーン Tel: 49-2129-3766400 トイツ - ハイルブロン Tel: 49-7131-67-3636 ドイツ – カールスルーエ Tel: 49-721-625370 ドイツ - ミュンヘン Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 ドイツ – ローセンハイム Tel: 49-8031-354-560 イスラエル - ラーナナ Tel: 972-9-744-7705 イタリア ー ミラノ Tel: 39-0331-742611 Fax: 39-0331-466781 イタリア ー パドバ Tel: 39-049-7625286 オランダ – デルーネン Tel: 31-416-690399 Fax: 31-416-690340 ノルウェー - トロンハイム Tel: 47-7289-7561 ホーラント – ワルシャワ Tel: 48-22-3325737 ルーマニア - ブカレスト Tel: 40-21-407-87-50 スペイン – マトリート Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 スウェーデン – イェーテホリ Tel: 46-31-704-60-40 スウェーデン – ストックホルム Tel: 46-8-5090-4654 イキ・リス – ウォーキンカ・ム Tel: 44-118-921-5800 Fax: 44-118-921-5820