

## Traveo™ ファミリマイコン S6J3200/S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/ S6J3510 シリーズの ADC 使用方法

著者: Yoshihiro Tsunokawa

関連製品ファミリ: Traveo ファミリ  
S6J3200/3300/3350/3360/3370/3400/3510 シリーズ関連ドキュメント: [関連ドキュメント](#)

AN211319 は S6J3200/S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/S6J3510 シリーズ MCU に搭載されている ADC の使い方・設定方法を、サンプルアプリケーションを用いて説明します。このドキュメントには、ソフトウェアトリガ・ハードウェアトリガ・グループ処理・レンジコンパレータ・パルスディテクション・診断・キャリブレーションの機能を使用するサンプルアプリケーションが含まれています。

### 目次

1	はじめに .....	1	7	パルスディテクション処理 .....	20
2	S6J3200/S6J3300/S6J3350/S6J3360/S6J3370/ S6J3400/S6J3510 シリーズ MCU の ADC 概要 .....	2	7.1	パルスディテクション設定 .....	21
3	ソフトウェアトリガ .....	2	7.2	A/D 変換とパルスディテクションの結果の出力 .....	22
3.1	ポート設定 .....	2	8	診断機能 .....	23
3.2	ADC グローバル設定 .....	4	8.1	ゼロトランジション電圧 の最大値とフルス ケールトランジション電圧の最小値の確認 ...	24
3.3	ソフトウェアトリガを使用する場合の論理チャ ネルの設定 .....	7	8.2	診断処理 .....	25
3.4	ソフトウェアトリガを使用した A/D 変換 .....	10	9	キャリブレーション機能 .....	27
4	ハードウェアトリガ .....	11	9.1	オフセット調整の説明 .....	27
4.1	ハードウェアトリガを使用する場合の論理チャ ネルの設定 .....	11	9.2	ゲイン調整の説明 .....	28
4.2	ハードウェアトリガを使用した A/D 変換 .....	13	9.3	キャリブレーション処理 .....	28
5	グループ処理 .....	14	9.4	オフセット調整処理 .....	30
5.1	グループ処理の場合の論理チャネルの設定 .....	14	9.5	ゲイン調整処理 .....	33
5.2	グループ処理の A/D 変換 .....	16	10	関連ドキュメント .....	35
6	レンジコンパレータ処理 .....	16	11	改訂履歴 .....	36
6.1	レンジコンパレータ設定 .....	17		セールス、ソリューションおよび法律情報 .....	37
6.2	レンジコンパレータを使用する場合の A/D 変換 .....	19			

## 1 はじめに

このアプリケーションノートは ADC インターフェースの使い方を説明します。対象の MCU は Cypress Traveo™ ファミリ S6J3200/S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/S6J3510 シリーズです。

このドキュメントに含まれるサンプルアプリケーションは S6J3200 シリーズで使用することを想定しています。しかし、その他のシリーズでも、そのサンプルアプリケーションを使用できます。その場合は、S6J3200 シリーズのハードウェアマニュアルとデータシートの代わりに対象商品の [ハードウェアマニュアルとデータシート](#) を参照してください。

## 2 S6J3200/S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/S6J3510 シリーズ MCU の ADC 概要

ADC は入力されたアナログ電圧をデジタル値に変換します。このシリーズ MCU の ADC は 8 個のレンジコンパレータ, 64 個のパルスディテクションユニット, そして 64 個の変換データレジスタを搭載しています。

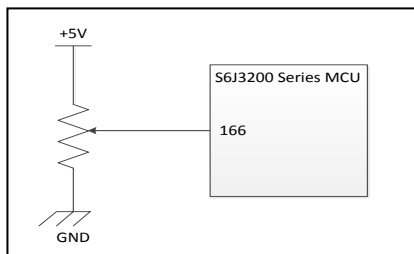
詳細は、[ハードウェアマニュアル](#)の 12/10/8-Bit Analog to Digital Converter の章を参照してください。

## 3 ソフトウェアトリガ

この章では、S6J3200L シリーズ MCU の 166 ピンに入力された電圧をデジタル値に変換し、その値を MCU の RAM に格納するアプリケーションを紹介します。A/D 変換は、main 関数内の無限ループで、ソフトウェアトリガにより繰り返し行われます。図 1 にハードウェアの構成を示します。

このアプリケーションノート内のすべてのアプリケーションは S6J3200L シリーズ MCU (216 ピン) を使うことを想定しています。S6J3200K シリーズ MCU (208 ピン) を使用する場合は、[ポート設定](#)のプロセスにおいて、表 1 の 'TEQFP216' 列の代わりに、'TEQFP208' 列を参照してください。

図 1. A/D 変換使用例



以下で説明するプロセスとサンプルコードにより、このアプリケーションを作成できます。

### 3.1 ポート設定

この章では、ピン番号 166 番に対応する MCU ポートをアナログ入力に設定にする方法を説明します。

S6J3200 シリーズ MCU のユーザは、使用するポートを汎用入出力、ペリフェラル入出力、ADC 入力のいずれかの機能に設定する必要があります。

ハードウェアマニュアルの Chapter 10: "Port Description," の "Package Pin Number TEQFP216" 列から、ピン番号 166 を見つけてください。表 1 によると、216 ピンのシリーズでは、166 ピンは Port3\_07 とアナログインプット 15 (AN15) に割り当てられています。これらの情報は、166 ピンをアナログインプットとして設定するソフトウェアを作成する上で必要です。

A/D 変換はアナログインプット (AN)の機能が割り当てられているピンでのみ行うことができます。ユーザは、最初にこの表をみて、使用するピンをアナログインプット(AN)の割り当てられているものの中から選択します。

表 1. ポート配置表

Port Name	Description	Package Pin Number		Remark
		TEQFP208	TEQFP216	
AN10	ADC Analog 10 input pin	98	102	
AN11	ADC Analog 11 input pin	99	103	
AN12	ADC Analog 12 input pin	100	104	
AN13	ADC Analog 13 input pin	101	105	
AN14	ADC Analog 14 input pin	102	106	
AN15	ADC Analog 15 input pin	160	166	
AN16	ADC Analog 16 input pin	161	167	
P3_03	General-Purpose I/O port	99	103	
P3_04	General-Purpose I/O port	100	104	
P3_05	General-Purpose I/O port	101	105	
P3_06	General-Purpose I/O port	102	106	
P3_07	General-Purpose I/O port	160	166	
P3_08	General-Purpose I/O port	161	167	
P3_09	General-Purpose I/O port	162	168	

以下のサンプルコードは、166 ピンをアナログインプット(AN)として設定するものです (コード 1)。  
 このサンプルコードは、ADC に供給されるクロック CLK\_LCP1A が 120 MHz であることを前提にしています。

コード 1. ポート設定のサンプルコード

```

/***** Port Settings For AN15 *****/
PPC_KEYCDR = 0x100000CE;
PPC_KEYCDR = 0x500000CE;
PPC_KEYCDR = 0x900000CE;
PPC_KEYCDR = 0xD00000CE;
PPC_PCFGR307 = 0x0000;

GPIO_KEYCDR = 0x2000003C;
GPIO_KEYCDR = 0x6000003C;
GPIO_KEYCDR = 0xA000003C;
GPIO_KEYCDR = 0xE000003C;
GPIO_DDCR3 = 0x00000080;
  
```

キーコードレジスタの操作

ポートのアナログ入力設定

キーコードレジスタの操作

ポートのアナログ入力設定

### 3.1.1 コード説明

ポートをアナログ入力に設定するときは、そのポートに対応する PPC\_PCFGR レジスタの PIE ビット、POF ビット、PDE ビット、PUE ビット、を 0 に設定します。さらに、そのポートに対応する GPIO\_DDR レジスタのビットを 0 に設定します。[S6J3200, HARDWARE MANUAL](#), Chapter 11, section 3.3, “Analog I/O Setting”を参照してください。

このサンプルコードでは、その他のビットは 0 に設定しています。これらはアナログ入力には影響しません。

### 3.1.2 キーコードレジスタの操作

レジスタ PPC\_PCFGR307 と GPIO\_DDCR3 はキーコードレジスタを操作しなくてはアクセスできません。キーコードレジスタの詳細については、[S6J3200, Traveo Family Hardware Manual Platform Part](#), Chapter 49, section 4.9, “GPIO Key Code Register (GPIO\_KEYCDR)”と section 4.11, “PPC Key Code Register (PPC\_KEYCDR)”を参照してください。キーコードレジスタに設定すべき値はターゲットになるレジスタのアドレスに依存します。

### 3.1.3 ポートのアナログ入力設定

PPC\_PCFGR307 は AN15 に対応する P3\_07 を設定するためのレジスタです。詳しくは [S6J3200, Traveo Family HARDWARE MANUAL Platform Part](#), Chapter 49, section 4.10, “Port Setting Register (PPC\_PCFGRijj)”を参照してください。

GPIO\_DDCR3 は GPIO\_DDR レジスタのクリアするためのレジスタです。詳しくは [S6J3200, Traveo Family HARDWARE MANUAL Platform Part](#), Chapter 49, section 4.3, “Data Direction Clear Register (GPIO\_DDCRi)”を参照してください。

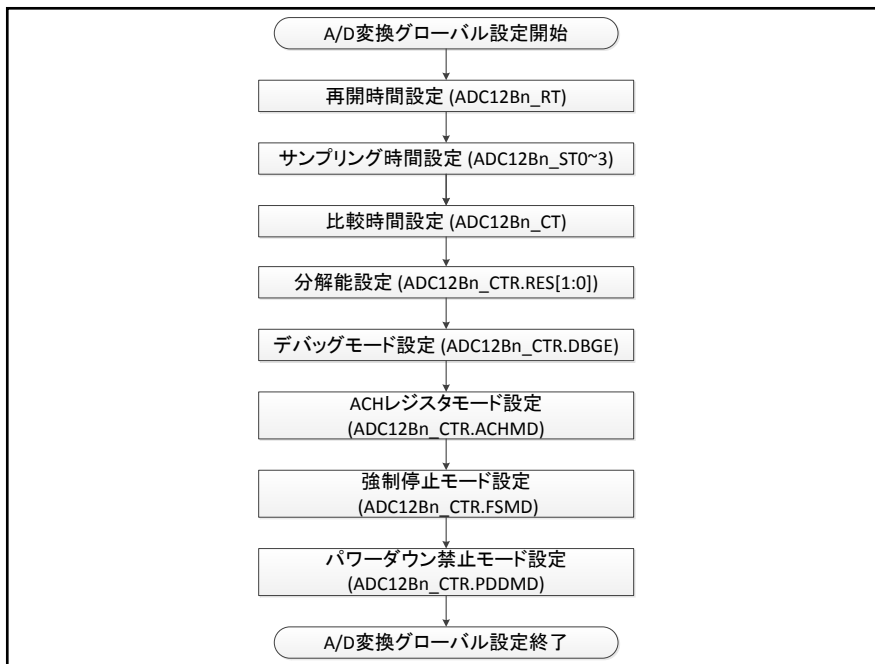
このサンプルコードでは、PPC\_PCFGR307 と GPIO\_DDCR3 の 7 ビット目は 0 に設定します。

## 3.2 ADC グローバル設定

この章では ADC ユニット全体の設定の方法と、そのサンプルコードを示します。

下のフローチャート(図 2)に従って、ADC グローバル設定を行ってください。

図 2. ADC グローバル設定フローチャート



ADC グローバル設定のサンプルコードは以下です(コード 2)。

## コード 2. ADC グローバル設定サンプルコード

/***** A/D Converter Global Settings *****/	
ADC12B0_RT = 120;	再開時間設定
ADC12B0_ST0 = 36;	サンプリグ時間設定
ADC12B0_ST1 = 36;	
ADC12B0_ST2 = 36;	
ADC12B0_ST3 = 36;	
ADC12B0_CT = 8;	比較時間設定
ADC12B0_CTRL_RES = 0;	分解能設定
ADC12B0_CTRL_DBGE = 0;	デバッグモード設定
ADC12B0_CTRL_ACHMD = 0;	ACH レジスタモード設定
ADC12B0_CTRL_FSMD = 0;	強制停止モード設定
ADC12B0_CTRL_PDDMD = 1;	パワーダウン禁止モード設定

### 3.2.1 コード説明

ADC に供給されるクロックは CLK\_LCP1A です。

### 3.2.2 再開時間設定

再開時間は以下の式により計算されます。ADC12B0\_RT はそのレジスタの値を、CLK\_LCP1A はそのクロックの周波数を示します。

$$\text{Resumption time[s]} = \text{ADC12B0\_RT} / \text{CLK\_LCP1A[Hz]} \quad \text{式 1}$$

[S6J3200 Series, Datasheet](#) を参照してください(表 2)。この表によると、Resumption time (再開時間)の最大値は 1 μs です。これは、1 μs 以上の再開時間が必要であることを示しています。また、再開時間は仕様の範囲内で最小の値に設定することが推奨されます。このことと、CLK\_LCP1A が 120 MHz であることを考慮すると、式 1 は、

$$1[\mu\text{s}] \leq \text{ADC12B0\_RT} / 120[\text{MHz}]$$

になります。この式を整理すると、

$$\text{ADC12B0\_RT} \geq 120$$

です。ADC12B0\_RT の適切な値は 120 ということがわかります。

### 3.2.3 サンプリグ時間設定

サンプリグ時間は以下の式により計算されます。ADC12B0\_STx はそのレジスタの値を、CLK\_LCP1A はそのクロックの周波数を示します。

$$\text{Sampling time[s]} = \text{ADC12B0\_STx} / \text{CLK\_LCP1A[Hz]} \quad \text{式 2}$$

[S6J3200 Series, Datasheet](#) を参照してください(表 2)。この表によると、Sampling time (サンプリグ時間) の最小値は 0.3 μs です。サンプリグ時間は仕様の範囲内で最小の値に設定されることが推奨されます。このことと、CLK\_LCP1A が 120 MHz であることを考慮すると、式 2 は、

$$0.3[\mu\text{s}] \leq \text{ADC12B0\_STx} / 120[\text{MHz}]$$

になります。この式を整理すると、

$$\text{ADC12B0\_STx} \geq 36$$

です。ADC12B0\_STx の適切な値は 36 です。

ADC のサンプリグ時間は、ADC グローバル設定の中で 4 種類設定できます。その 4 種類から、どのレジスタを使用するかを論理チャネルごとに選べます。

### 3.2.4 比較時間設定

比較時間は以下の式により計算されます。ADC12B0\_CT はそのレジスタの値を、CLK\_LCP1A はそのクロックの周波数を示します。

If [ADC12B0\_CT value < 4]:

$$\text{Compare time[s]} = (\text{ADC12B0\_CT} \times 12 + 4) / \text{CLK\_LCP1A[Hz]} \quad \text{式 3}$$

If [ADC12B0\_CT value ≥ 4]:

$$\text{Compare time[s]} = (\text{ADC12B0\_CT} \times 13) / \text{CLK\_LCP1A[Hz]} \quad \text{式 4}$$

[S6J3200 Series, Datasheet](#) を参照してください(表 2)。この表によると、Compare time(比較時間)の最小値は 0.8 μs です。比較時間は仕様の範囲内で最小の値を設定することが推奨されます。このことと、CLK\_LCP1A が 120 MHz であることを考慮すると、式 3、と式 4 より

$$0.8[\mu\text{s}] \leq (\text{ADC12B0\_CT} \times 13) / 120[\text{MHz}]$$

になります。この式を整理すると、

$$\text{ADC12B0\_CT} \geq 7.38$$

です。ADC12B0\_CT の値の適切な値は 8 です。

表 2. ADC データシート

Parameter	Symbol	Pin Name	Value			Unit	Remarks
			Min	Typ	Max		
Resolution	-	-	-	-	12	bit	
Total Error	-	-	-	-	±12	LSB	
Integral Non linearity	-	-	-	-	±4.0	LSB	
Differential Non linearity	-	-	-	-	±1.9	LSB	
Zero transition voltage	V <sub>ZT</sub>	AN0 to AN49	AVRL -11.5LSB	-	AVRL +12.5LSB	V	
Full-scale transition voltage	V <sub>FST</sub>	AN0 to AN49	AVRH -13.5LSB	-	AVRH +10.5LSB	V	
<u>Sampling time</u>	t <sub>SMP</sub>	-	0.3	-	-	μs	
<u>Compare time</u>	t <sub>CMP</sub>	-	0.8	-	28	μs	
A/D conversion time	t <sub>CNV</sub>	-	1.1	-	-	μs	
A/D trigger input time		ADTRG	4t <sub>CLK_LCP1A</sub>	-	-	ns	4t <sub>CLK_LCP1A</sub> ≥ 100ns
			100				4t <sub>CLK_LCP1A</sub> < 100ns
<u>Resumption time</u>	-	-	-	-	1	us	-
Analog port input current	I <sub>AIN</sub>	AN0 to AN17	-1.0	-	1.0	μA	V <sub>AVSS</sub> ≤ V <sub>AIN</sub> ≤ V <sub>AVCC</sub>
		AN18 to AN25	-2.0	-	2.0	μA	
		AN26 to AN49	-3.0	-	3.0	μA	
Analog input voltage	V <sub>AIN</sub>	AN0 to AN49	AVSS	-	AVRH	V	
Reference voltage	AVRH	AVRH5	4.5	-	5.5	V	AV <sub>CC</sub> ≥ AVRH
	AVRL	AVRL5/AVSS	-	0.0	-	V	
Power supply current	I <sub>A</sub>	AVCC	-	500	900	μA	
	I <sub>AH</sub>		-	1.0	100	μA	
	I <sub>R</sub>	AVRH	-	1.0	2.0	mA	
	I <sub>RH</sub>		-	-	5.0	μA	
Variation between channels	-	AN0 to AN49	-	-	4.0	LSB	

### 3.2.5 分解能設定

ADC12B0\_CTRL\_RES は ADC の分解能を選択するためのレジスタです。以下のように、8-bit, 10-bit, 12-bit から ADC の分解能を選択できます。

ADC12B0\_CTRL\_RES = 0 or 2: 12-bit mode

ADC12B0\_CTRL\_RES = 1: 10-bit mode

ADC12B0\_CTRL\_RES = 3: 8-bit mode

このサンプルアプリケーションでは、12-bit mode に設定します。

### 3.2.6 デバッグモード設定

この bit が 1 に設定された場合、デバッグモードになります。

デバッグモードでは、プロセッサがデバッグ状態の場合、ADC は現在行っている変換は完了させますが、それに続く変換は行いません。その状態で、この bit が 0 に設定された場合、止まっていたところから変換が再開されます。

### 3.2.7 ACH レジスタモード設定

Direct ACH レジスタモードか Latched ACH レジスタモードかを選択します。

ADC12B0\_CTRL\_ACHMD = 0: Direct ACH レジスタモード

ADC12B0\_CTRL\_ACHMD = 1: Latched ACH レジスタモード

Direct ACH レジスタモードでは、ADC12B0\_STAT\_ACH レジスタは現在の変換が完了した論理チャンネルの数を示します。

Latched ACH レジスタモードでは、ADC12B0\_STAT\_ACH レジスタは最後に変換が終了した論理チャンネルの番号を示します。

### 3.2.8 強制停止モード設定

強制停止モードを使用するかを選択します。

ADC12B0\_CTRL\_FSMMD = 1: 強制停止モード

ADC12B0\_CTRL\_FSMMD = 0: 強制停止モードでない

強制停止モードについては、[S6J3200, Hardware Manual](#) を参照してください。

このサンプルコードでは強制停止モードは使用しません。

### 3.2.9 パワーダウン禁止モード設定

パワーダウン禁止モードを使用するかを選択します。

ADC12B0\_CTRL\_PDDMD = 1: パワーダウン禁止モード

ADC12B0\_CTRL\_PPDMD = 0: パワーダウン禁止モードでない

パワーダウン禁止モードについては、[S6J3200, Hardware Manual](#) を参照してください。

このサンプルアプリケーションでは 1 に設定しています。

## 3.3 ソフトウェアトリガを使用する場合の論理チャンネルの設定

1 つの論理チャンネルにつき、1 つの A/D 変換結果レジスタがあります。ADC12Bn\_CHCTRLx\_ANIN レジスタにより、それぞれの論理チャンネルは任意のアナログ入力(AN)にアサインすることができます。

[図 3](#) はソフトウェアトリガを使用する場合の論理チャンネルの設定フローチャートです。



図 3. ソフトウェアトリガを使用する場合の設定フローチャート



コード 3 はソフトウェアトリガを使用した、ADC 論理チャンネル設定のサンプルコードです。

このサンプルコードでは論理チャンネル 0 が使用されていますが、適切なアナログ入力の設定をすれば、任意の論理チャンネルで、このアプリケーションを動作させることができます。

コード 3. ソフトウェアトリガを使用する場合の設定サンプルコード

```

/***** A/D Converter Ch0 Settings *****/

ADC12B0_CHCTRL0_ANIN = 15;  ← アナログ入力選択
ADC12B0_CHCTRL0_TRGTYP = 0; ← トリガタイプ設定
ADC12B0_CHCTRL0_CHPRI = 0;  ← 論理チャンネル優先度設定
ADC12B0_CHCTRL0_RSMRST = 1; ← 再開/再起動/停止モード設定
ADC12B0_CHCTRL0_DP = 0;    ← データ保護有効設定
ADC12B0_CHCTRL0_SMTIME = 0; ← サンプリング時間選択
ADC12B0_CDONEIRQE0_CDONEIRQE0 = 0; ← A/D 変換終了割り込み許可設定
ADC12B0_GRP_IRQE0_GRP_IRQE0 = 0; ← グループ割り込み許可設定
  
```

### 3.3.1 アナログ入力選択

今回使用する 166 ピンはアナログ端子 15 (AN15)に対応しているため、ADC 論理チャンネル 0 をアナログ端子 15 (AN15)に割り当てます。

### 3.3.2 トリガタイプ設定

トリガの種類を選択します。



トリガタイプはこのレジスタの値によって以下のように決定します。

TRGTYP = 0b00: ソフトウェアトリガのみ

TRGTYP = 0b01: ソフトウェアトリガまたはハードウェアトリガ

TRGTYP = 0b10: 変換完了トリガ

TRGTYP = 0b11: Idle トリガ

このサンプルコードでは“ソフトウェアトリガのみ”に設定するため 0 に設定しています。

### 3.3.3 論理チャネル優先度設定

論理チャネルの優先度を設定します。

このレジスタ値 0 が最高優先度、15 が最低優先度に対応します。

このサンプルコードでは最高優先度の 0 に設定しています。

### 3.3.4 再開/再起動/停止 モード設定

RSMRST レジスタは、より高い優先度のチャネルの変換が終了したとき、グループ変換再開時の挙動を選択するためのレジスタです。

RSMRST = 0: 停止

RSMRST = 1: 再開

RSMRST = 2: 再スタート

サンプルコードでは再開の 1 に設定しています。今回は 1 チャネルしか使用していないため、この設定は影響を及ぼしません。

### 3.3.5 データ保護有効設定

このビット値が 1 の場合、データ保護機能が有効です。

データ保護機能が有効の場合、前回の A/D データが読み出されるまで次の変換は開始されません。

このサンプルコードでは無効の 0 に設定しています。

### 3.3.6 サンプリング時間選択

グローバル設定の行程で設定した、ADC12B0\_STx (x = 0~3)の中から、この論理チャネルで使用する設定を選択します。

このサンプルコードでは ADC12B0\_ST0 を選択しています。

### 3.3.7 A/D 変換完了割込み許可設定

0: 変換完了割込み禁止

1: 変換完了割込み許可

変換完了割込みは、このレジスタ bit が 1 であり、かつ、対応する割込みフラグ

ADC12B0\_CDONEIRQ1\_CDONEIRQ63~32 と ADC12Bn\_CHSTAT32~63.CDONEIRQ が 1 にセットされた時に発生します。

このサンプルコードでは割込みを使用しないため 0 に設定します。

### 3.3.8 グループ変換中断割込み許可設定

0: グループ変換中断割込み禁止

1: グループ変換中断割込み許可

グループ変換中断割込みはこのビットが“1”であり、かつ、対応する割込みフラグ

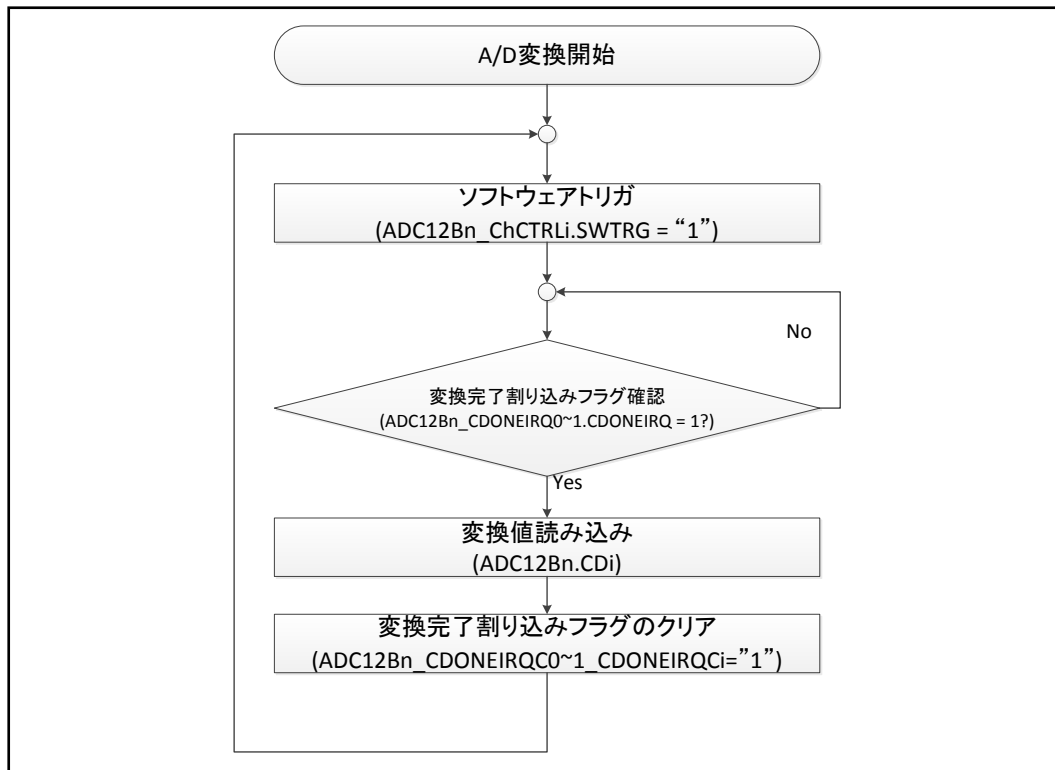
ADC12Bn\_GRP\_IRQ0\_GRP\_IRQ31~0 と ADC12Bn\_CHSTAT0~31.GRPIRQ が 1 にセットされた時に発生します。

このサンプルコードでは割込みを使用しないため 0 に設定します。

### 3.4 ソフトウェアトリガを使用した A/D 変換

図 4 はソフトウェアトリガを使用した、A/D 変換のフローチャートです。

図 4. ソフトウェアトリガを使用した A/D 変換のフローチャート



コード 4 はソフトウェアトリガを使用した、A/D 変換のサンプルコードです。

コード 4. ソフトウェアトリガを使用した A/D 変換のサンプルコード

```

/***** Endless main loop *****/
for (;;)
{
    /***** Software Trigger *****/
    ADC12B0_CHCTRL0_SWTRG = 1;    ← ソフトウェアトリガ
    /***** Wait A/D Convert Completion *****/
    while (ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1); ← 変換完了割り込みフラグの確認
    {
        ClearWatchdog();
    }
    /***** Get AD Value *****/
    Volume = ADC12B0_CD0;    ← 変換値の読み込み
    /***** Clear Conversion Interrupt Flag *****/
    ADC12B0_CDONEIRQC0_CDONEIRQC0 = 1; ← 変換完了割り込みフラグのクリア
}
  
```

```

/***** Clear hardware watchdog *****/
ClearWatchdog();
}

```

#### 3.4.1 チャネル 0 のソフトウェアトリガ

ADC12B0\_CHCTRL0\_SWTRG に 1 を書き込むことによって、ADC チャネル 0 にソフトウェアトリガがかかります。

#### 3.4.2 変換完了割込みフラグの確認

サンプルコード内では、この変換完了割込みフラグを見ることで、A/D 変換が終了したことを確認します。フラグが 1 に設定されていれば、while ループを抜けます。

#### 3.4.3 変換値の読み込み

この値を読むことで、A/D 変換データを変数 'Volume' へ格納します。

#### 3.4.4 変換完了割込みフラグのクリア

ADC12B0\_CDONEIRQ0\_CDONEIRQ0 に 1 を書き込むことで、A/D 変換完了割込みフラグをクリアします。

## 4 ハードウェアトリガ

S6J3200 シリーズ MCU の ADC のユーザは、ハードウェアトリガにより A/D 変換を開始させることができます。使用するハードウェアトリガは、外部入力ポート・リロードタイマ・アウトプットコンペア・ベースタイマから選択できます。

この章では、MCU ピン番号 166 に入力された電圧値をデジタル値に変換し、変換したデジタル値を MCU の RAM に格納するアプリケーションを紹介します。A/D 変換はリロードタイマのハードウェアトリガにより一定の時間間隔で繰り返されます。ハードウェアの構成はソフトウェアトリガの章の図 1 と同じです。

「ソフトウェアトリガ」の章の「ポート設定」と「ADC グローバル設定」を事前に行ってください。

以下で説明するプロセスとサンプルコードにより、このアプリケーションを作成できます。

このサンプルアプリケーションではリロードタイマ 0 チャネルを使用します。

### 4.1 ハードウェアトリガを使用する場合の論理チャネルの設定

ハードウェアトリガを使用した論理チャネルの設定フローチャートは、ソフトウェアトリガの章の図 3 と同様です。

コード 5 はハードウェアトリガを使用した論理チャネルの設定のサンプルコードです。

コード 5. ハードウェアトリガを使用した論理チャネル設定のサンプルコード

```

/***** A/D Converter Ch0 Settings *****/
ADC12B0_CHCTRL0_ANIN = 15;
ADC12B0_CHCTRL0_TRGTYP = 1; ← トリガタイプ設定
ADC12B0_CHCTRL0_CHPRI = 0;
ADC12B0_CHCTRL0_RSMRST = 1;
ADC12B0_CHCTRL0_DP = 0;
ADC12B0_CHCTRL0_SMTIME = 0;
ADC12B0_CDONEIRQ0_CDONEIRQ0 = 0;
ADC12B0_GRP_IRQ0_GRP_IRQ0 = 0;

```

#### 4.1.1 トリガタイプ設定

A/D 変換開始トリガの種類を選択します。

トリガタイプは TRGTYP レジスタの値によって以下のように決定します。

TRGTYP = 0b00: ソフトウェアトリガのみ

TRGTYP = 0b01: ソフトウェアトリガまたはハードウェアトリガ

TRGTYP = 0b10: 変換完了トリガ

TRGTYP = 0b11: Idle トリガ

このサンプルコードでは”ソフトウェアトリガまたはハードウェアトリガ”に設定するため 1 に設定しています。

その他の設定はソフトウェアの章に記載されているものと同じです。

このサンプルコードのその他の部分はソフトウェアトリガの章のサンプルコードと同じです。

#### 4.1.2 リロードタイマチャネル 0 のアンダフローを ADC チャネル 0 のトリガへ割り当てる

[S6J3200, Hardware Manual](#), Chapter 11, Port Configuration, 3.1 “Resource Input Configuration Module”を参照し、“Resource”カテゴリの“ADC12B\_HWTRG0”を探してください。(表 3)

表 3. リソース入力表

Register (Offset)	Resource	RESSEL [3:0] /PORTS EL[3:0]	Source for Resource Input							
			0	1	2	3	4	5	6	7
			8	9	10	11	12	13	14	15
RIC_RE SIN438 (0x036C)	ADC12B_HW TRG0	RESSEL (0-7)	PORT_P N	RLT0_U F SET	RLT1_U F SET	OCU0_O TD0	OCU1_O TD0	BT0_AD TOUT0	BT1_AD TOUT2	BT3_AD TOUT2
		RESSEL (8-15)	-	-	-	-	-	-	-	-
		PORTS EL (0-7)	-	-	-	-	-	-	-	-
		PORTS EL (8-15)	-	-	-	-	-	-	-	-

リロードタイマのアンダフローを ADC チャネル 0 のトリガとして使用する場合は RIC\_RESIN438 に 1 をセットします。

コード 6. ADC ハードウェアトリガの割り当てサンプルコード

```

/***** Assignment RLT0 to A/D Converter Ch0 HW Trigger *****/
RIC_KEYCDR = 0x1000036C;
RIC_KEYCDR = 0x5000036C;
RIC_KEYCDR = 0x9000036C;
RIC_KEYCDR = 0xD000036C;
RIC_RESIN438 = 0x0001;

```

← キーコードレジスタ操作

#### 4.1.3 コード説明

リソースインプット設定レジスタ RIC\_RESIN438 に 1 をセットします。

RIC\_RESIN438 はキーコードレジスタです。キーコードレジスタの詳細については、[S6J3200, Traveo Family Hardware Manual Platform Part](#), Chapter 49, Section 4.13 を参照してください。

キーコードレジスタに設定するべき値はターゲットになるレジスタのアドレスに依存します。

#### 4.1.4 割り当てられたリロードタイマの起動

ADC チャンネル 0 のトリガに割り当てられた、リロードタイマチャンネル 0 を設定しスタートさせます。

コード 7 はリロードタイマの設定しスタートさせるサンプルコードです。

コード 7. リロードタイマ開始サンプルコード

```
static void Rlt0_Start(void)
{
    /****** Reload Timer 0 Setting *****/
    RLTO_TMRLR = 14999;
    RLTO_TMCSR_CNTE = 1;
    RLTO_TMCSR_CSL = 2;
    RLTO_TMCSR_RELD = 1;
    RLTO_TMCSR_TRG = 1;
}
```

リロードタイマの設定の詳細については、[S6J3200, Traveo Family HARDWARE MANUAL Platform Part](#), Chapter 44 を参照してください。

## 4.2 ハードウェアトリガを使用した A/D 変換

コード 8 はハードウェアトリガを使用した、A/D 変換のサンプルコードです。

コード 8. ハードウェアトリガを使用した A/D 変換のサンプルコード

```
/****** Endless main loop *****/
for (;;)
{
    /****** Wait A/D Convert Completion *****/
    while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1);
    {
        ClearWatchdog();
    }
    /****** Get ADC Value *****/
    Volume = ADC12B0_CD0;
    /****** Clear Conversion Interrupt Flag *****/
    ADC12B0_CDONEIRQC0_CDONEIRQC0 = 1;
    /****** Clear hardware watchdog *****/
    ClearWatchdog();
}
```

#### 4.2.1 コード説明

このサンプルコードはソフトウェアトリガをかけていないこと以外は、ソフトウェアトリガの章のサンプルコードと同じです。

## 5 グループ処理

S6J3200 シリーズ MCU の ADC は、一度のトリガにより、複数端子を使用して、順番に変換を行う機能を持っています。また、使用する端子と変換する順番は、アナログ入力端子 (AN) として設定可能なポートから任意に選択できます。ただし、複数の端子を使用して同時に変換を行う機能は持っていません。

一度のトリガで順番に変換される ADC の論理チャネルの集合を「グループ」と呼びます。

ユーザはグループを、連番の論理チャネルのトリガタイプを設定することによって定義できます。グループの最初のチャネルのトリガタイプは「ソフトウェアトリガのみ」・「ハードまたはソフトウェアトリガ」・「Idle トリガ」のいずれかでなくてはなりません。もしそれに続くチャネルのトリガタイプが「変換完了トリガ」でないなら、グループはただ 1 つのチャネルしか含みません。そうでないならば、グループは連番のチャネルのトリガタイプが「変換完了トリガ」である限り続きます。

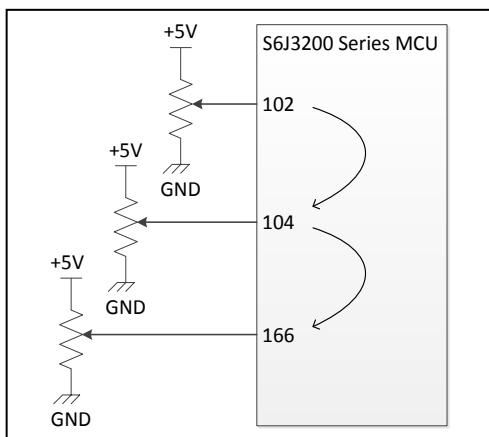
最初のチャネルにトリガがかかり変換が終了した後、その終了をトリガとしてグループ内の次のチャネルの変換が開始されます。これがそのグループ内の全チャネルの変換が行われるまで繰り返されます。

この章では「ソフトウェアトリガ」の章の「ポート設定」と「ADC グローバル設定」を事前に行ってください。

図 5 に示すように、一度のソフトウェアトリガで端子番号 102, 104, 166 の順番で電圧値を変換し、MCU 内部の RAM に格納するサンプルアプリケーションを示します。

「ソフトウェアトリガ」の章の「ポート設定」と「ADC グローバル設定」を事前に行ってください。

図 5. グループ変換設定例



以下で説明するプロセスとサンプルコードにより、このアプリケーションを作成できます。

### 5.1 グループ処理の場合の論理チャネルの設定

それぞれの論理チャネルの設定フローチャートはソフトウェアトリガのものと同じです。

表 1 に示すように、端子 102 はアナログ入力 10 (AN10), 端子 104 はアナログ入力 12 (AN12), 端子 166 はアナログ入力 15 (AN15)に対応します。

ADC グループ処理の論理チャネル設定サンプルコードを[エラー! 参照元が見つかりません。](#)に示します。このサンプルコードはチャネル 0, 1, 2 を使用していますが、チャネル番号が連続していれば任意のチャネルを使用できます。

### コード 9. グループ処理の ADC 論理チャネル設定サンプルコード

```

/***** A/D Converter Ch0 Settings *****/
ADC12B0_CHCTRL0_ANIN = 10;  ← アナログ入力選択
ADC12B0_CHCTRL0_TRGTYP = 0; ← トリガタイプ設定: ソフトウェアトリガ
ADC12B0_CHCTRL0_CHPRI = 0;
ADC12B0_CHCTRL0_RSMRST = 1;
ADC12B0_CHCTRL0_DP = 0;
ADC12B0_CHCTRL0_SMTIME = 0;
ADC12B0_CDONEIRQE0_CDONEIRQE0 = 0;
ADC12B0_GRPIRQE0_GRPIRQE0 = 0;

/***** A/D Converter Ch1 Settings *****/
ADC12B0_CHCTRL1_ANIN = 12;  ← アナログ入力選択
ADC12B0_CHCTRL1_TRGTYP = 2; ← トリガタイプ設定: 変換完了トリガ
ADC12B0_CHCTRL1_CHPRI = 0;
ADC12B0_CHCTRL1_RSMRST = 1;
ADC12B0_CHCTRL1_DP = 0;
ADC12B0_CHCTRL1_SMTIME = 0;
ADC12B0_CDONEIRQE0_CDONEIRQE1 = 0;
ADC12B0_GRPIRQE0_GRPIRQE1 = 0;

/***** A/D Converter Ch2 Settings *****/
ADC12B0_CHCTRL2_ANIN = 15;  ← アナログ入力選択
ADC12B0_CHCTRL2_TRGTYP = 2; ← トリガタイプ設定: 変換完了トリガ
ADC12B0_CHCTRL2_CHPRI = 0;
ADC12B0_CHCTRL2_RSMRST = 1;
ADC12B0_CHCTRL2_DP = 0;
ADC12B0_CHCTRL2_SMTIME = 0;
ADC12B0_CDONEIRQE0_CDONEIRQE2 = 0;
ADC12B0_GRPIRQE0_GRPIRQE2 = 0;

```

#### 5.1.1 アナログ入力選択

グループに割り当てたいアナログインプットピンの設定をします。このサンプルでは、論理チャネルのアナログインプットは、チャネル 0 が端子 102 (AN10)、チャネル 1 が端子 104 (AN12)、チャネル 2 が端子 166 (AN15) にそれぞれ割り当てられます。

#### 5.1.2 トリガタイプ設定

グループ処理を行う場合は、グループの最初のチャネルのトリガタイプを'変換完了トリガ'以外に設定します。このサンプルコードではソフトウェアトリガによりグループ変換を開始します。したがって、グループの最初のチャネルである論理チャネル 0 のレジスタ値を 0 に設定しています。



その他の連番の論理チャネルを'変換完了トリガ'に設定します。これにより、論理チャネル 0 の変換完了が、論理チャネル 1 のトリガです。同様に、論理チャネル 1 の変換完了が論理チャネル 2 のトリガです。

## 5.2 グループ処理の A/D 変換

コード 10 はグループ処理の A/D 変換のサンプルコードです。

コード 10. グループ処理の A/D 変換サンプルコード

```

/***** Endless main loop *****/
for (;;)
{
    /***** Software Trigger *****/
    ADC12B0_CHCTRL0_SWTRG = 1;
    /***** Wait A/D Convert Completion *****/
    while(ADC12B0_STAT_BUSY); ← ADC Busy フラグの確認
    {
        ClearWatchdog();
    }
    /***** Get ADC Value *****/
    Volume0 = ADC12B0_CD0;
    Volume1 = ADC12B0_CD1;
    Volume2 = ADC12B0_CD2;
    /***** Clear hardware watchdog *****/
    ClearWatchdog();
}

```

### 5.2.1 ADC Busy フラグの確認

このグループ処理のサンプルアプリケーションでは、ADC Busy フラグを読むことでグループ変換の完了を確認しています。その他のコードに関しては、ソフトウェアトリガの章と同じです。

BUSY フラグはすべての変換要求が処理されると 0 にクリアされ、再開時間が経過し、かつ、次の変換要求を受け付けるまで、クリアされたままになります。

## 6 レンジコンパレータ処理

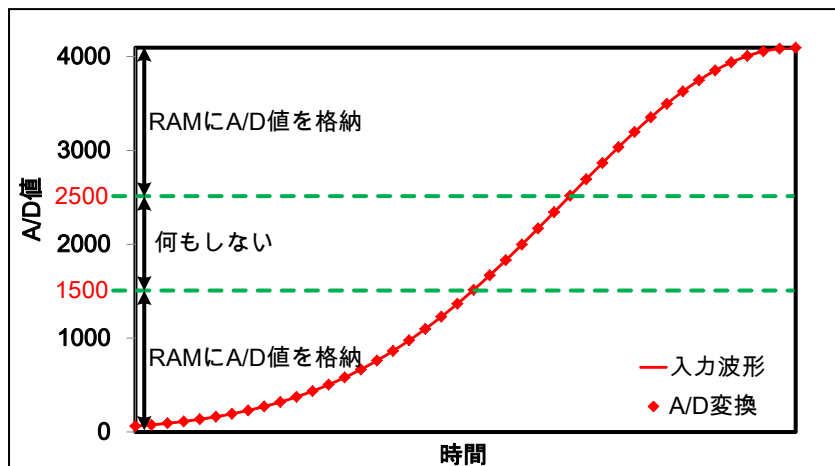
閾値には上限閾値と下限閾値があり、ユーザがその値を設定できます。上限・下限閾値を設定するレジスタはそれぞれ 8 個ずつあり、ユーザは、64 個の論理チャネルに、個別に閾値設定の 1 つを割り当てられます。

この機能は電圧の異常を監視するために使用されます。

この章では、MCU の端子番号 166 に与えられた電圧をデジタル値に変換するアプリケーションを紹介します。**エラー! 参照元が見つかりません。**に示すように、もし A/D 変換値が'2500'よりも大きい、または'1500'よりも小さかったとき、その値を RAM に格納します。A/D 変換はソフトウェアトリガにより行われ、main 関数内の無限ループで繰り返されます。ハードウェアの構成はソフトウェアトリガの章の図 1 と同じです。

「ソフトウェアトリガ」の章の「ポート設定」と「ADC グローバル設定」を事前に行ってください。

図 6. レンジコンパレータアプリケーションの振る舞い

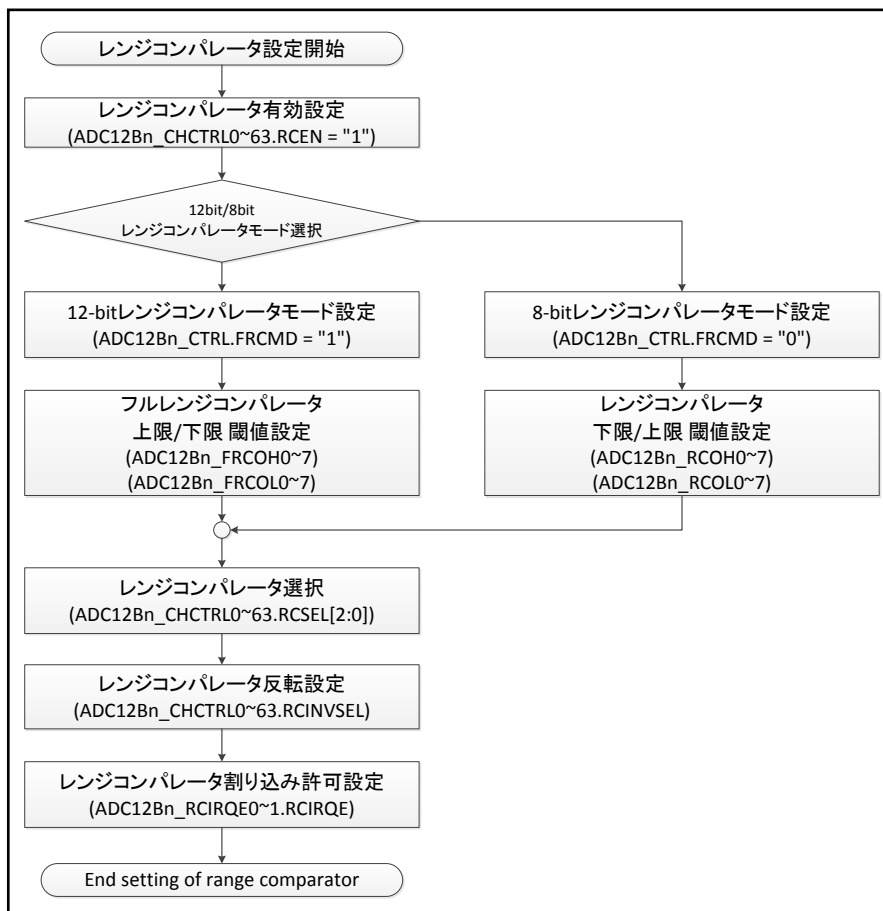


以下で説明するプロセスとサンプルコードにより、このアプリケーションを作成できます。

## 6.1 レンジコンパレータ設定

図 7 はレンジコンパレータ設定のフローチャートです。

図 7. レンジコンパレータ設定フローチャート



コード 11 はレンジコンパレータ設定のサンプルコードです。

コード 11. レンジコンパレータ設定サンプルコード

```

/***** A/D Converter Range Global Settings *****/
/***** A/D Converter Range Ch0 Settings *****/
ADC12B0_CHCTRL0_RCEN = 1; ← レンジコンパレータ有効設定
ADC12B0_CTRL_FRCMD = 1; ← 12-bit/8-bit レンジコンパレータモード設定
ADC12B0_FRCOH0 = 2500;
ADC12B0_FRCOL0 = 1500;
ADC12B0_FRCOH1 = 4095;
ADC12B0_FRCOL1 = 0;
ADC12B0_FRCOH2 = 4095;
ADC12B0_FRCOL2 = 0;
ADC12B0_FRCOH3 = 4095;
ADC12B0_FRCOL3 = 0;
ADC12B0_FRCOH4 = 4095;
ADC12B0_FRCOL4 = 0;
ADC12B0_FRCOH5 = 4095;
ADC12B0_FRCOL5 = 0;
ADC12B0_FRCOH6 = 4095;
ADC12B0_FRCOL6 = 0;
ADC12B0_FRCOH7 = 4095;
ADC12B0_FRCOL7 = 0;
ADC12B0_CHCTRL0_RCSEL = 0; ← レンジコンパレータ選択
ADC12B0_CHCTRL0_RCINVSEL = 0; ← レンジコンパレータ反転設定
ADC12B0_RCIRQE0_RCIRQE0 = 0; ← レンジコンパレータ割込み許可設定

```

フルレンジコンパレータの上限・下限閾値設

#### 6.1.1 レンジコンパレータ有効設定

ADC12B0\_CHCTRL0\_RCEN は、レンジコンパレータ許可・禁止を選択するためのレジスタです。

0: レンジコンパレータ禁止

1: レンジコンパレータ許可

#### 6.1.2 12-bit/8-bit レンジコンパレータモード設定

ADC12B0\_CTRL\_FRCMD は、上限・下限閾値データのビット数モードを選択するためのレジスタです。

0: 8 ビットレンジコンパレータモード

1: 12 ビットレンジコンパレータモード

このサンプルでは、12 ビットコンパレータモードをつかうため 1 に設定します。

#### 6.1.3 フルレンジコンパレータの上限・下限閾値設定

ADC12B0\_FRCOHx レジスタは 12 ビットレンジコンパレータの上限閾値を設定するためのレジスタです。また、ADC12B0\_FRCOLx レジスタは 12 ビットレンジコンパレータの下限閾値を設定するためのレジスタです。

12 bit レンジコンペアモード (ADC12B0\_CTRL\_FRCMD = 1) では、これらのレジスタが使用されます。8 bit レンジコンパレータモード (ADC12B0\_CTRL\_FRCMD = 0) では、ADC12B0\_RCOHx・ADC12B0\_FRCOLx レジスタがこれらの代わりに使われます。

このサンプルでは、FRCOH0 は 2500 に FRCOL0 は 1500 にセットされます。FRCOHx, FRCOLx はこのサンプルでは使用しませんが、それぞれ 4095, 0 にセットしています。

#### 6.1.4 レンジコンパレータ選択

RCSEL は、その論理チャンネルで、どの閾値レジスタを選択するかを選択するレジスタです。

000: ADC12B0\_FRCOH0 と ADC12B0\_FRCOL0 で閾値を定義されるレンジコンパレータ 0 を使用します。

001: ADC12B0\_FRCOH1 と ADC12B0\_FRCOL1 で閾値を定義されるレンジコンパレータ 0 を使用します。

010: ADC12B0\_FRCOH2 と ADC12B0\_FRCOL2 で閾値を定義されるレンジコンパレータ 0 を使用します。

011: ADC12B0\_FRCOH3 と ADC12B0\_FRCOL3 で閾値を定義されるレンジコンパレータ 0 を使用します。

⋮

111: ADC12B0\_FRCOH7 と ADC12B0\_FRCOL7 で閾値を定義されるレンジコンパレータ 0 を使用します。

このサンプルでは、ADC12B0\_FRCOH0 と ADC12B0\_FRCOL0 を使用するため 000 を設定します。

#### 6.1.5 レンジコンパレータ反転設定

RCINVSEL は、レンジコンパレータが範囲外を検出するか、または範囲内を検出するかを選択するためのレジスタです。

0: レンジコンパレータは範囲外を検出します。すなわち、A/D 変換結果が上限閾値より大きいまたは、下限閾値より小さい場合に割込みフラグがセットされます。

1: レンジコンパレータは範囲内を検出します。すなわち、A/D 変換結果が上限閾値以下かつ、下限閾値以上の場合に割込みフラグがセットされます。

このサンプル、では範囲外を検出するため 0 にセットします。

#### 6.1.6 レンジコンパレータ割込み許可設定

RCIRQE0 はレンジコンパレータ割込み許可・禁止を選択するためのレジスタです。

0: レンジコンパレータ割込み禁止

1: レンジコンパレータ割込み許可

このサンプルでは、割込みを使用しないため 0 に設定します。

### 6.2 レンジコンパレータを使用する場合の A/D 変換

コード 12 はレンジコンパレータを使用した A/D 変換のサンプルコードです。

## コード 12. レンジコンパレータを使用した A/D 変換サンプルコード

```

/***** Endless main loop *****/
for (;;)
{
    /***** Software Trigger *****/
    ADC12B0_CHCTRL0_SWTRG = 1;
    /***** Wait A/D Convert Completion *****/
    while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1);
    {
        ClearWatchdog();
    }
    /***** Clear Conversion Done Flag *****/
    ADC12B0_CDONEIRQC0_CDONEIRQC0 = 1;
    if(ADC12B0_RCIRQ0_RCIRQ0 == 1) ← レンジコンパレータ割込みフラグ確認
    {
        /***** Outside range *****/
        /***** Clear Range Compare Flag *****/
        ADC12B0_RCIRQC0_RCIRQC0 = 1;
        /***** Get ADC Value *****/
        Volume = ADC12B0_CD0;
    }
    /***** Clear hardware watchdog *****/
    ClearWatchdog();
}

```

### 6.2.1 コード説明

A/D 変換終了後、レンジコンパレータの割込みフラグを読みます。そして、その値が 1 の場合は、そのフラグをクリアし A/D 値を 'Volume' という名前の変数に格納します。

## 7 パルスディテクション処理

レンジコンパレータの比較結果はパルスディテクション機能によりフィルタをかけることができます。

それぞれの ADC 論理チャネルは、ポジティブイベントにより減少するリロードレジスタ(ポジティブカウンタ)と、ネガティブイベントにより減少するリロードカウンタ(ネガティブカウンタ)の両方を持ちます(ADC12Bn\_PCCTRL0 ~ 63.PCTPRL[7:0] and ADC12Bn\_PCCTRL0~63.PCTNRL[4:0])。

この機能は電圧を監視する際、ノイズなどによる電圧異常の誤検知を避けるために使用されます。

この章では、レンジコンパレータを使用したサンプルアプリケーションを示します。[レンジコンパレータ設定](#)の章での設定を使用し、A/D 変換結果が 2500 よりも大きいまたは、1500 よりも小さい場合をポジティブイベント、1500 以上かつ 2500 以下の場合をネガティブイベントと定義します。このサンプルアプリケーションは[図 8](#)に示すように、ポジティブイベントが 5 回連続で発生した場合に A/D 変換結果を RAM に格納し、それ以降、A/D 変換値を格納し続けます。また、[図 9](#)に示すように、ポジティブイベントの連続はネガティブイベントが 2 回連続で発生したとき初めて中断されます。すなわち、ネガティブイベントが 1 回だけ発生したとしても、ポジティブイベントの連続カウントは続きます。ハードウェアの構成はソフトウェアトリガの章の[図 1](#)と同じです。

[ポート設定](#)と[ADC グローバル設定](#)と[レンジコンパレータ設定](#)は事前に行ってください。

図 8. パルスディテクションアプリケーションの振る舞い 1

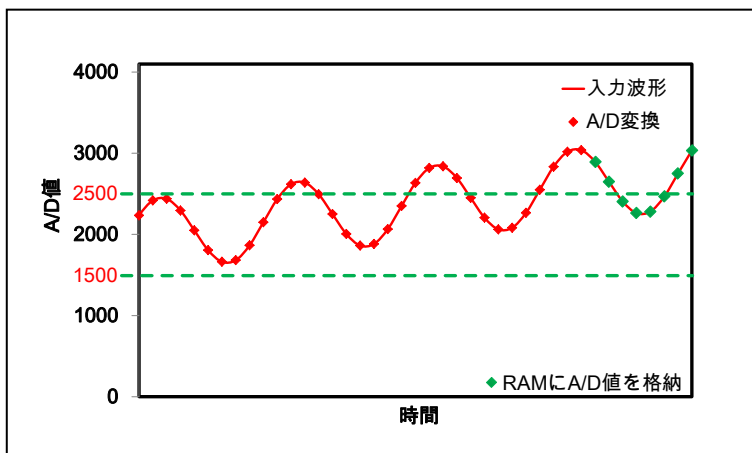
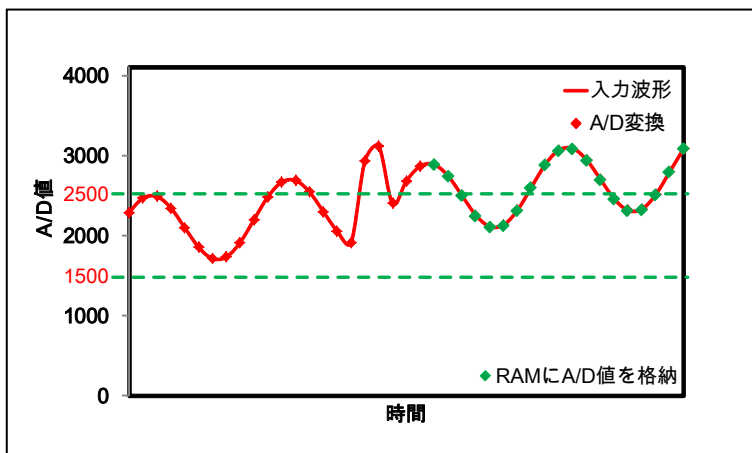


図 9. パルスディテクションアプリケーションの振る舞い 2

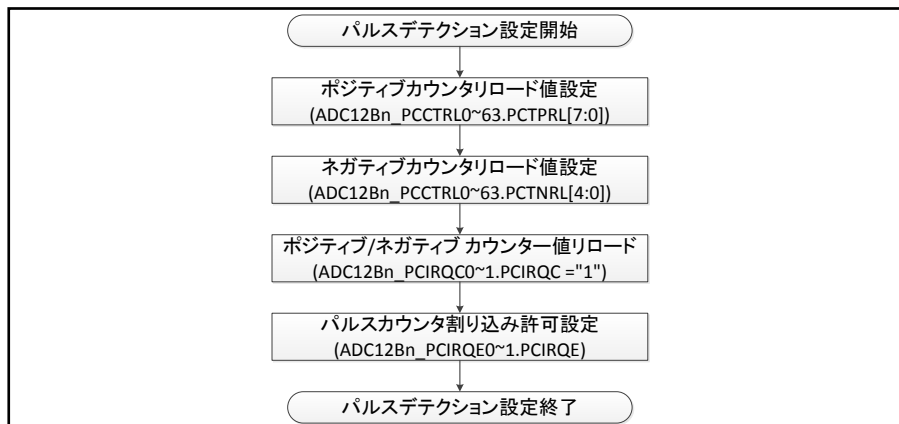


以下で説明するプロセスとサンプルコードにより、このアプリケーションを作成できます。

## 7.1 パルスディテクション設定

[図 10](#) はパルスディテクション設定のフローチャートです。

図 10. パルスディテクション設定フローチャート



コード 13 はパルスディテクションを使用する場合の 論理チャネルの設定例です。

このサンプルコードはレンジコンパレータ設定の後に実装してください。

コード 13. パルスディテクション設定サンプルコード

```

/***** A/D Converter Pulse Ch0 Settings *****/
ADC12B0_PCCTRL0_PCTPRL = 5;  ← ポジティブカウンタリロード値設定
ADC12B0_PCCTRL0_PCTNRL = 2;  ← ネガティブカウンタリロード値設定
ADC12B0_PCIRQC0_PCIRQC0 = 1; ← ポジティブ/ネガティブカウンタ値リロード
ADC12B0_PCIRQE0_PCIRQE0 = 0; ← パルスカウンタ割り込み許可設定
  
```

#### 7.1.1 ポジティブカウンタリロード値設定

ADC12B0\_PCCTRL0\_PCTPRL は、レンジコンパレータのポジティブイベントをカウントするの使用するポジティブカウンタのリロード値を設定するためのレジスタです。

このサンプルでは 5 に設定します。

#### 7.1.2 ネガティブカウンタリロード値設定

これらのレジスタは、レンジコンパレータのネガティブイベントをカウントするのに使用されるネガティブカウンタ PCTNCT のリロード値を設定するためのレジスタです。このサンプルでは 2 に設定します。

#### 7.1.3 ポジティブ/ネガティブカウンタ値リロード

ポジティブ・ネガティブカウンタのリロード値を設定した後、このレジスタに一度 1 を書き込んでください。

#### 7.1.4 パルスカウンタ割り込み許可設定

パルスカウンタの割り込み許可・禁止を禁止に設定します。

### 7.2 A/D 変換とパルスディテクションの結果の出力

コード 14 はパルスディテクションを使用した A/D 変換のサンプルコードです。

コード 14. パルスディテクションの結果を出力するサンプルコード

```

/***** Endless main loop *****/
for (;;)
{
    /***** Software Trigger *****/
  
```



```

ADC12B0_CHCTRL0_SWTRG = 1;

/***** Wait A/D Convert Completion *****/
while(ADC12B0_STAT_BUSY);
{
    ClearWatchdog();
}

/***** Clear Conversion Done Flag *****/
ADC12B0_CDONEIRQ0_CDONEIRQ0 = 1;
if(ADC12B0_PCCTRL0_PCTNCT == 1) ← パルスカウンタ割込みフラグ確認
{
    /***** Get AD Value *****/
    Volume = ADC12B0_CD0;
}

/***** Clear hardware watchdog *****/
ClearWatchdog();
}
}

```

### 7.2.1 コード説明

A/D 変換後にパルスカウンタ割込みフラグを確認します。このフラグの値が 1 の時 A/D 変換値を変数'Volume'に格納します。このフラグが 1 であるということは、A/D 変換結果が 5 回連続で範囲外だったことを示します。

## 8 診断機能

この章では、ADC 診断のフローチャートとサンプルコードを説明します。

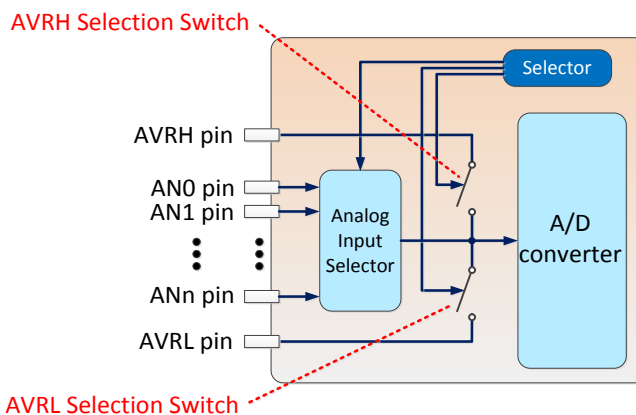
図 11 に示すように、ADC に入力する電圧を参照電圧 AVRH と AVRL から選択できます。

AVRH は上限参照電圧です。その A/D 値は 0xFFFF (= 4095) です。

AVRL は下限参照電圧です。その A/D 値は 0x000 (= 0) です。

これらは、ADC 診断と ADC キャリブレーションのための機能です。

図 11. ADC 入力の AVRH と AVRL 切り替え



## 8.1 ゼロトランジション電圧の最大値とフルスケールトランジション電圧の最小値の確認

初めに、[S6J3200 Series, Datasheet](#) をみてゼロトランジション電圧(VZT)の最大値とフルスケールトランジション電圧 (VFST) の最小値を確認してください。(表 4)

データシートによると、VZT の最大値は  $AVRL + 12.5 \text{ LSB}$  です。その A/D 値は 12.5 (0+12.5) です。

すなわち、ADC が正常な場合、AVRL の A/D 変換値は 12 以下です。(The AD value  $\leq 12 = 0x00C$ )

VFST の最小値は  $AVRH - 13.5 \text{ LSB}$  です。その A/D 値は 4081.5 (095 – 13.5) です。

すなわち、ADC が正常な場合、AVRH の A/D 変換値は 4082 以上です。(The AD value  $\geq 4082 = 0xFF2$ )

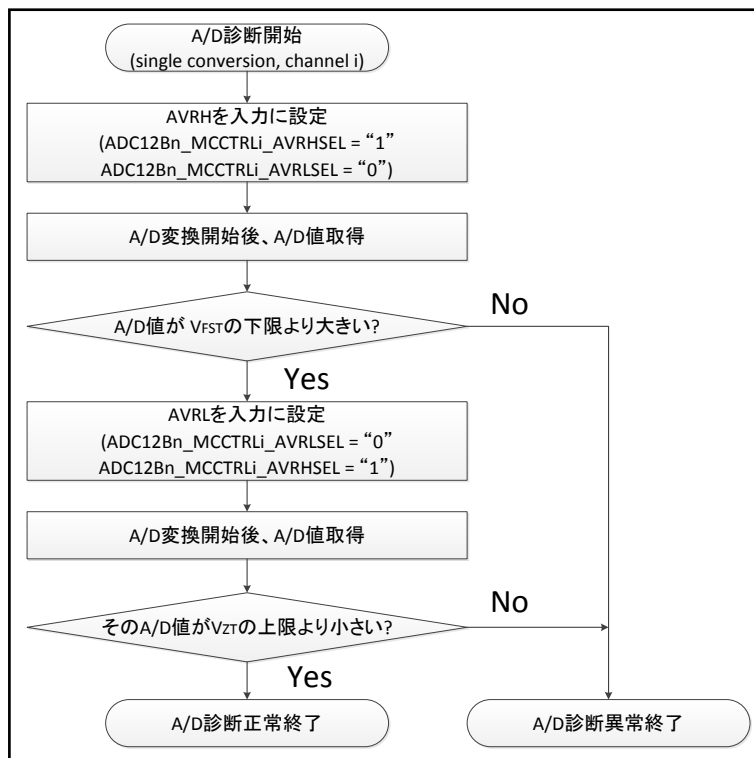
表 4. デバイスデータシートの VZT の最大値と VFST の最小値

Parameter	Symbol	Pin Name	Value			Unit	Remarks
			Min	Typ	Max		
Resolution	-	-	-	-	12	bit	
Total Error	-	-	-	-	$\pm 12$	LSB	
Integral Non linearity	-	-	-	-	$\pm 4.0$	LSB	
Differential Non linearity	-	-	-	-	$\pm 1.9$	LSB	
<u>Zero transition voltage</u>	V <sub>ZT</sub>	AN0 to AN49	AVRL -11.5LSB	-	AVRL +12.5LSB	V	
<u>Full-scale transition voltage</u>	V <sub>FST</sub>	AN0 to AN49	AVRH -13.5LSB	-	AVRH +10.5LSB	V	
Sampling time	t <sub>SMP</sub>	-	0.3	-	-	μs	
Compare time	t <sub>CMP</sub>	-	0.8	-	28	μs	
A/D conversion time	t <sub>CNV</sub>	-	1.1	-	-	μs	
A/D trigger input time		ADTRG	4t <sub>CLK_LCP1A</sub>	-	-	ns	4t <sub>CLK_LCP1A</sub> ≥ 100ns
			100				4t <sub>CLK_LCP1A</sub> < 100ns
Resumption time	-	-	-	-	1	us	-
Analog port input current	I <sub>AIN</sub>	AN0 to AN17	-1.0	-	1.0	μA	V <sub>AVSS</sub> ≤ V <sub>AIN</sub> ≤ V <sub>AVCC</sub>
		AN18 to AN25	-2.0	-	2.0	μA	
		AN26 to AN49	-3.0	-	3.0	μA	
Analog input voltage	V <sub>AIN</sub>	AN0 to AN49	AVSS	-	AVRH	V	
Reference voltage	AVRH	AVRH5	4.5	-	5.5	V	AV <sub>CC</sub> ≥ AVRH
	AVRL	AVRL5/AVSS	-	0.0	-	V	
Power supply current	I <sub>A</sub>	AVCC	-	500	900	μA	
	I <sub>AH</sub>		-	1.0	100	μA	
	I <sub>R</sub>	AVRH	-	1.0	2.0	mA	
	I <sub>RH</sub>		-	-	5.0	μA	
Variation between channels	-	AN0 to AN49	-	-	4.0	LSB	

## 8.2 診断処理

図 12 は ADC 診断のフローチャートです。

図 12. ADC 診断フローチャート



コード 15 は ADC 診断のサンプルコードです。

コード 15. ADC 診断のサンプルコード

```
#define AD_DIAGNOSIS_SUCCESS 0
#define AD_DIAGNOSIS_FAILURE 1
unsigned char AD_Diagnosis(void)
{
    unsigned short ADValue;
    /***** Switch analog input to AVRH *****/
    ADC12B0_MCCTRL0_AVRHSEL = 1;
    ADC12B0_MCCTRL0_AVRLSEL = 0; } ← ADC への入力を AVRH に設定
    /***** Software Trigger *****/
    ADC12B0_CHCTRL0_SWTRG = 1;
    /***** Wait A/D Convert Completion *****/
    while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1);
    {
        ClearWatchdog();
    }
    /***** Get AD Value *****/
    ADValue = ADC12B0_CD0;
    /***** Confirm the AD value is higher than VFSTmin *****/
    if(ADValue < 0xFF2) ← VFS の下限と比較
    {
        /***** finish AD diagnosis with abnormality *****/
        ADC12B0_MCCTRL0_AVRHSEL = 0;
        ADC12B0_MCCTRL0_AVRLSEL = 0; } ← ADC 診断異常終了
        return AD_DIAGNOSIS_FAILURE;
    }
    /***** Switch analog input to AVRL *****/
    ADC12B0_MCCTRL0_AVRHSEL = 0;
    ADC12B0_MCCTRL0_AVRLSEL = 1; } ← ADC への入力を AVRL に設定
    /***** Software Trigger *****/
    ADC12B0_CHCTRL0_SWTRG = 1;
    /***** Wait A/D Convert Completion *****/
    while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1);
    {
        ClearWatchdog();
    }
    /***** Get AD Value *****/
    ADValue = ADC12B0_CD0;
    /***** Confirm the AD value is higher than VZTmax *****/
    if(ADValue > 0x00C) ← VZT の上限と比較
    {
        /***** finish AD diagnosis with abnormality *****/
        ADC12B0_MCCTRL0_AVRHSEL = 0;
        ADC12B0_MCCTRL0_AVRLSEL = 0; } ← ADC 診断異常終了
        return AD_DIAGNOSIS_FAILURE;
    }
}
```

← A/D 変換値を取得

← A/D 変換値を取得

```

    }
    /***** finish AD diagnosis with success *****/
    ADC12B0_MCCTRL0_AVRHSEL = 0;
    ADC12B0_MCCTRL0_AVRLSEL = 0;
    return AD_DIAGNOSIS_SUCCESS;
  }

```

← ADC 診断正常終了

### 8.2.1 ADC への入力を AVRH に設定

ADC12B0\_MCCTRL0\_AVRHSEL は ADC への入力を、下限参照電圧 AVRH に選択するためのビットです。

0: AVRH 電圧が A/D 変換の入力として選択されない。

1: AVRH 電圧が A/D 変換の入力として選択される。

ADC12B0\_MCCTRL0\_AVRLSEL は ADC への入力を、下限参照電圧 AVRL に選択するためのビットです。

0: AVRL 電圧が A/D 変換の入力として選択されない。

1: AVRL 電圧が A/D 変換の入力として選択される。

AVRHSEL と AVRLSEL の両方が 1 にセットされた場合、AVRLSEL の設定が優先され AVRL 電圧が入力されます。

AVRHSEL と AVRLSEL のどちらかが 1 にセットされているとき、通常のアナログ入力の変換はできません。

ADC に AVRH を入力するために、ADC12B0\_MCCTRL0\_AVRHSEL を 1 に設定します。

ADC12B0\_MCCTRL0\_AVRLSEL は 0 に設定します。

### 8.2.2 A/D 変換値を取得

このサンプルコードではソフトウェアトリガを使用しています。

### 8.2.3 VFS の下限と比較

最大値はデータシートに記載されています (表 4)。

A/D 値が範囲外だった場合、診断を異常終了しエラーを返します。

### 8.2.4 ADC への入力を AVRL に設定

ADC に AVRL を入力するために、ADC12B0\_MCCTRL0\_AVRHSEL を 0 に設定、ADC12B0\_MCCTRL0\_AVRLSEL を 1 に設定します。

### 8.2.5 VZT の下限と比較

最小値はデータシートに記載されています (表 4)。

A/D 値が範囲外だった場合、診断を異常終了しエラーを返します。

## 9 キャリブレーション機能

ADC12Bn\_OCV と ADC12Bn\_GCV レジスタを使用して、ADC のオフセットとゲインを調整することができます。

この章では、ADC のキャリブレーションの方法を説明します。

### 9.1 オフセット調整の説明

ADC はオフセットのずれを補償するためのオフセット調整レジスタを搭載しています。ADC12B0\_OVC がそのレジスタです。

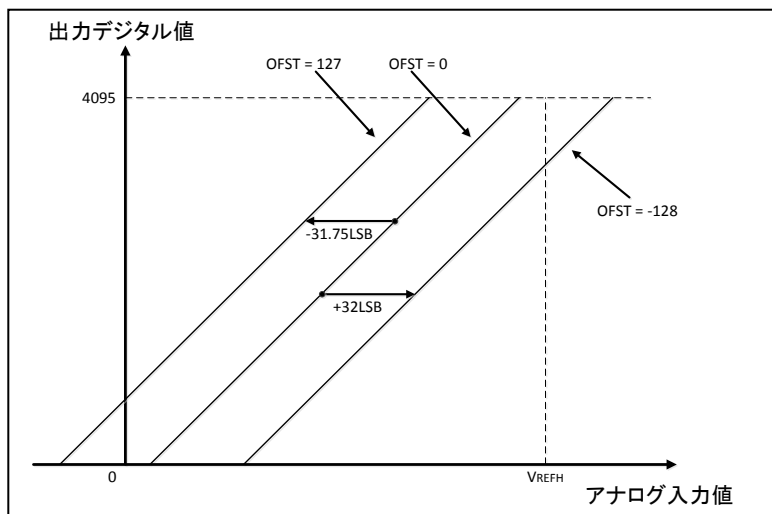
ADC12B0\_OVC には、+127 から -128 の値を設定できます。オフセットは 4 分の 1 LSB 単位で調整ができます。

出力デジタル値と ADC12B0\_OVC の値の関係式は以下です。OFST は ADC12B0\_OVC レジスタの値を示します。

$$\text{Output Digital Code} = \max\left(0, \min\left(4095, \text{floor}\left(\frac{V_{IN}}{V_{REFH}} \times 4096 + \frac{\text{OFST}}{4}\right)\right)\right)$$

図 13 に OFST の値とオフセットの変化を表します。

図 13. OFST とオフセット変化の関係



## 9.2 ゲイン調整の説明

ADC はゲインエラーを補償するためのゲイン調整レジスタを搭載しています。ADC12B0\_GVC がそのレジスタです。

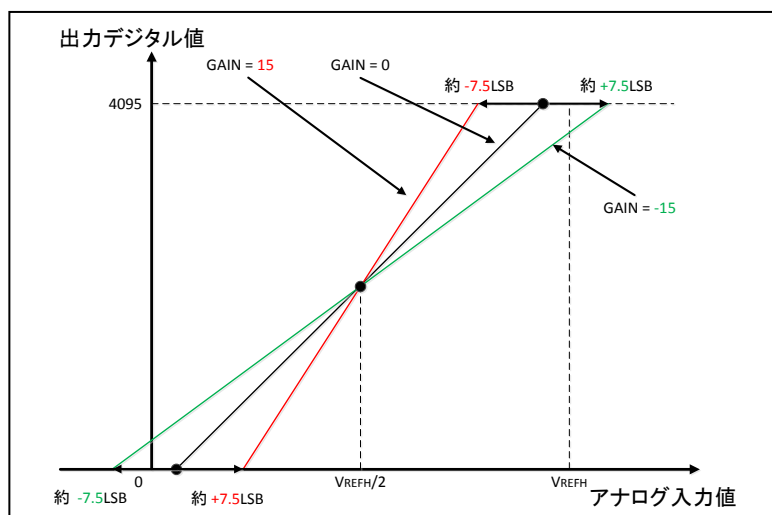
ADC12B0\_GVC レジスタには+15 から-15 までの値を設定できます。ゲインは約 1LSB 単位で調整できます。

出力デジタル値と、ADC12B0\_GVC の値の関係式は以下です。GAIN は ADC12B0\_GVC レジスタの値を示します。

$$\text{Output Digital Code} = \max \left( 0, \min \left( 4095, \text{floor} \left( \frac{4096 - \text{GAIN}}{4096} \times \left( V_{IN} - \frac{V_{REFH}}{2} \right) + 2048 \right) \right) \right)$$

図 14 に GAIN の値とゲインの変化の関係を示します。

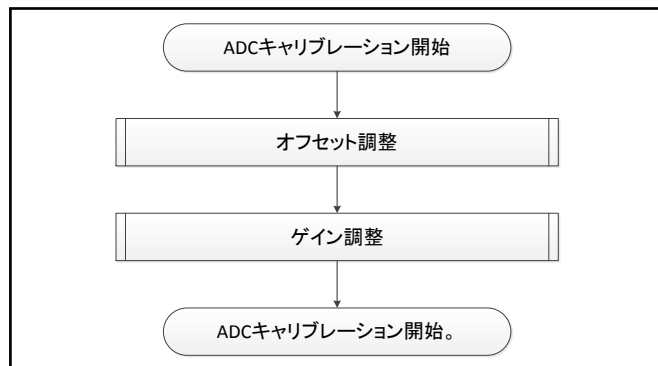
図 14. GVC とゲイン変化の関係



## 9.3 キャリブレーション処理

図 15 は ADC のキャリブレーションのフローチャートです。最初にオフセットの調整を行い、次にゲインの調整を行います。

図 15. キャリブレーションフローチャート



コード 16 は ADC キャリブレーションのサンプルコードです。

コード 16. ADC キャリブレーションのサンプルコード

```

#define ADC_CALIBRATION_SUCCESS 0
#define ADC_CALIBRATION_FAILURE 1
#define ADC_OFFSET_SUCCESS 0
#define ADC_OFFSET_FAILURE 1
#define ADC_GAIN_SUCCESS 0
#define ADC_GAIN_FAILURE 1
unsigned char ADC_Calibration(void)
{
    /***** Conduct Offset Adjustment *****/
    if(ADC_Offset_Adjustment() != ADC_OFFSET_SUCCESS) ← オフセット調整
    {
        /***** Finish Calibration with abnormality *****/
        return ADC_CALIBRATION_FAILURE; ← キャリブレーション異常終了
    }
    /***** Conduct Gain Adjustment *****/
    if(ADC_Gain_Adjustment() != ADC_GAIN_SUCCESS) ← ゲイン調整
    {
        /***** Finish Calibration with abnormality *****/
        return ADC_CALIBRATION_FAILURE; ← キャリブレーション異常終了
    }
    /***** Finish Calibration with success *****/
    return ADC_CALIBRATION_SUCCESS; ← キャリブレーション正常終了
}
  
```

#### 9.3.1 オフセット調整を行い正常終了したことを確認する

詳細については、[オフセット調整処理](#)の章を参照してください。

オフセット調整が成功した場合、その関数は"ADC\_OFFSET\_SUCCESS"を返し、プログラムは ADC ゲイン調整の処理に進みます。

オフセット調整に失敗した場合、エラーを報告し ADC キャリブレーションを終了します。

#### 9.3.2 ゲイン調整を行い正常終了したことを確認する

詳細については、[ゲイン調整処理](#)の章を参照してください。



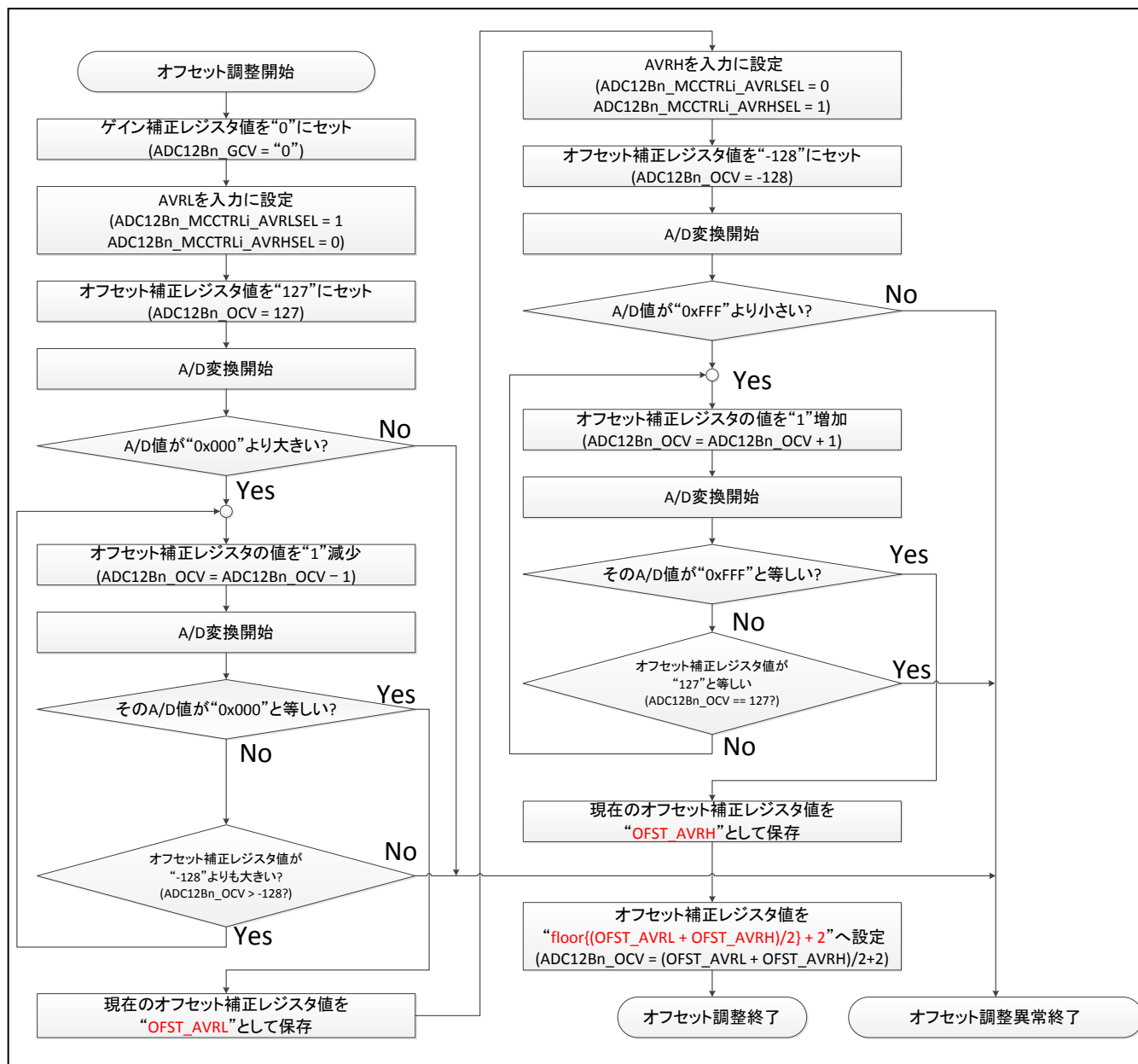
ゲイン調整が成功した場合、その関数は"ADC\_OFFSET\_SUCCESS"を返し、プログラムは ADC キャリブレーションを正常終了します。

ゲイン調整に失敗した場合、エラーを報告し ADC キャリブレーションを終了します。

## 9.4 オフセット調整処理

図 16 はオフセット調整のフローチャートです。

図 16. オフセット調整フローチャート



コード 17 はオフセット調整のサンプルコードです。

#### コード 17. オフセット調整のサンプルコード

```
#define OFFSET_REGISTER_VALUE_MAX (127)
#define OFFSET_REGISTER_VALUE_MIN (-128)
unsigned char ADC_Offset_Adjustment(void)
{
    signed short loop;
    unsigned short ADValue;
    signed char OFST_AVRL;
    signed char OFST_AVRH;

    /***** Set Offset Correction to "0" *****/
    ADC12B0_GCV = 0;
    /***** Switch analog input to AVRL *****/
    ADC12B0_MCCTRL0_AVRLSEL = 1;
    ADC12B0_MCCTRL0_AVRHSEL = 0;
    for(loop=OFFSET_REGISTER_VALUE_MAX;loop>=OFFSET_REGISTER_VALUE_MIN;loop--)
    {
        ADC12B0_OCV = loop;
        /***** Software Trigger *****/
        ADC12B0_CHCTRL0_SWTRG = 1;
        /***** Wait A/D Convert Completion *****/
        while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1)
        {
            ClearWatchdog();
        }
        /***** Get AD Value *****/
        ADValue = ADC12B0_CD0;
        if(loop == OFFSET_REGISTER_VALUE_MAX && ADValue == 0x000)
        {
            /***** Finish Offset Adjustment with abnormality *****/
            ADC12B0_MCCTRL0_AVRLSEL = 0;
            ADC12B0_MCCTRL0_AVRHSEL = 0;
            return ADC_OFFSET_FAILURE;
        }
        else if(ADValue == 0x000)
        {
            OFST_AVRL = loop;
            break;
        }
        if(loop == OFFSET_REGISTER_VALUE_MIN)
        {
            /***** Finish Offset Adjustment with abnormality *****/
            ADC12B0_MCCTRL0_AVRLSEL = 0;
            ADC12B0_MCCTRL0_AVRHSEL = 0;
            return ADC_OFFSET_FAILURE;
        }
    }
}
```

```

    }
}

/***** Switch analog input to AVRL *****/
ADC12B0_MCCTRL0_AVRLSEL = 0;
ADC12B0_MCCTRL0_AVRHSEL = 1;
for(loop=OFFSET_REGISTER_VALUE_MIN; loop<=OFFSET_REGISTER_VALUE_MAX; loop++)
{
    ADC12B0_OCV = loop;
    /***** Software Trigger *****/
    ADC12B0_CHCTRL0_SWTRG = 1;
    /***** Wait A/D Convert Completion *****/
    while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1);
    {
        ClearWatchdog();
    }
    /***** Get AD Value *****/
    ADValue = ADC12B0_CD0;
    if(loop == OFFSET_REGISTER_VALUE_MIN && ADValue == 0xFFFF)
    {
        /***** Finish Offset Adjustment with abnormality *****/
        ADC12B0_MCCTRL0_AVRLSEL = 0;
        ADC12B0_MCCTRL0_AVRHSEL = 0;
        return ADC_OFFSET_FAILURE;
    }
    else if(ADValue == 0xFFFF)
    {
        OFST_AVRH = loop;
        break;
    }
    if(loop == OFFSET_REGISTER_VALUE_MAX)
    {
        /***** Finish Offset Adjustment with abnormality *****/
        ADC12B0_MCCTRL0_AVRLSEL = 0;
        ADC12B0_MCCTRL0_AVRHSEL = 0;
        return ADC_OFFSET_FAILURE;
    }
}

/***** Set to floor{(OFST_AVRL + OFST_AVRH) / 2} + 2 *****/
ADC12B0_OCV = ((OFST_AVRL + OFST_AVRH) / 2) + 2;
/***** Finish Offset Adjustment with success *****/
ADC12B0_MCCTRL0_AVRLSEL = 0;
ADC12B0_MCCTRL0_AVRHSEL = 0;
return ADC_OFFSET_SUCCESS;
}

```

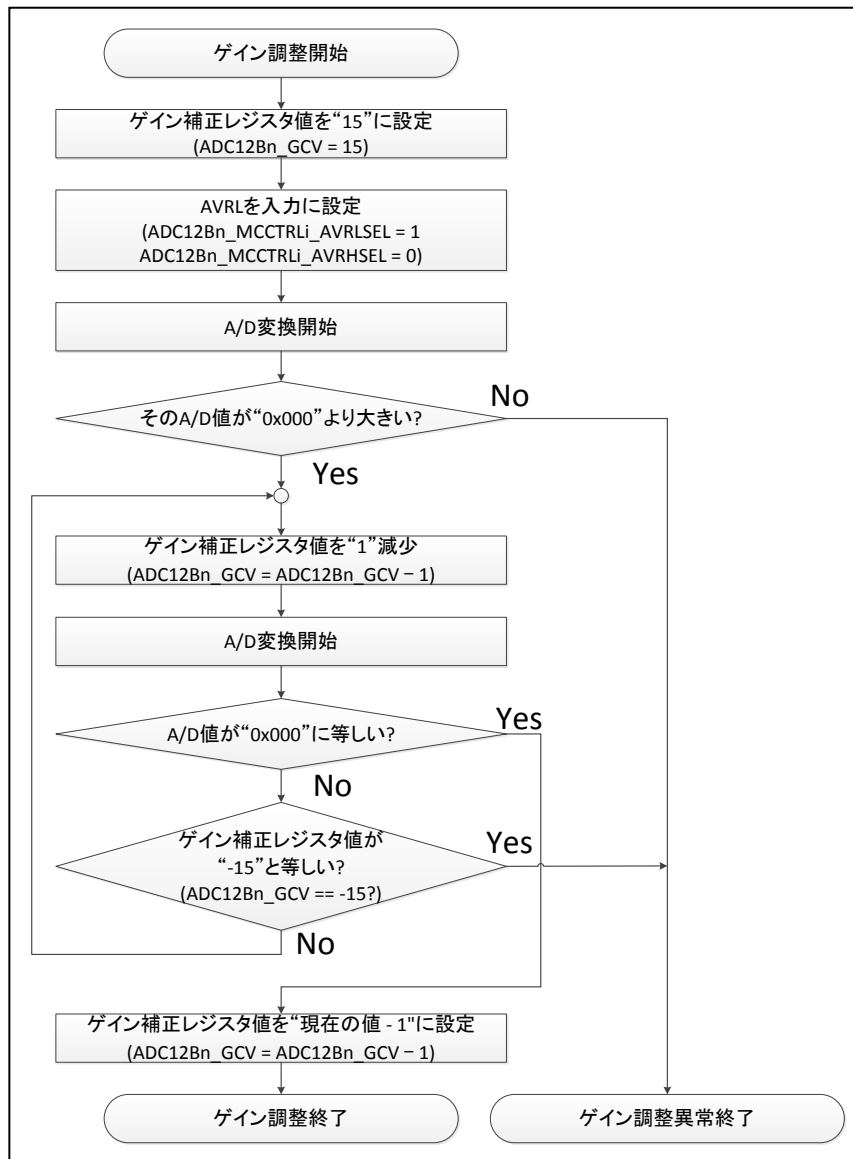
#### 9.4.1 コード説明

このサンプルコードはフローチャート (図 16) に従って、適切なオフセット値を見つけます。

## 9.5 ゲイン調整処理

図 17 はゲイン調整のフローチャートです。

図 17. ゲイン調整フローチャート



コード 18 はゲイン調整のサンプルコードです。

#### コード 18. ゲイン調整サンプルコード

```
#define GAIN_REGISTER_VALUE_MAX (15)
#define GAIN_REGISTER_VALUE_MIN (-15)
unsigned char ADC_Gain_Adjustment(void)
{
    signed short loop;
    unsigned short ADValue;
    /***** Switch analog input to AVRL *****/
    ADC12B0_MCCTRL0_AVRHSEL = 0;
    ADC12B0_MCCTRL0_AVRLSEL = 1;
    for(loop=GAIN_REGISTER_VALUE_MAX;loop>=GAIN_REGISTER_VALUE_MIN;loop--)
    {
        ADC12B0_GCV = loop;
        /***** Software Trigger *****/
        ADC12B0_CHCTRL0_SWTRG = 1;
        /***** Wait A/D Convert Completion *****/
        while(ADC12B0_CDONEIRQ0_CDONEIRQ0 != 1)
        {
            ClearWatchdog();
        }
        /***** Get AD Value *****/
        ADValue = ADC12B0_CD0;
        if(loop == GAIN_REGISTER_VALUE_MAX && ADValue == 0x000)
        {
            /***** Finish Gain Adjustment with abnormality *****/
            ADC12B0_MCCTRL0_AVRHSEL = 0;
            ADC12B0_MCCTRL0_AVRLSEL = 0;
            return ADC_GAIN_FAILURE;
        }
        else if(ADValue == 0x000)
        {
            break;
        }
        if(loop == GAIN_REGISTER_VALUE_MIN)
        {
            /***** Finish Gain Adjustment with abnormality *****/
            ADC12B0_MCCTRL0_AVRHSEL = 0;
            ADC12B0_MCCTRL0_AVRLSEL = 0;
            return ADC_GAIN_FAILURE;
        }
    }
    /***** Set Gain Compensation to "the current value - 1" *****/
    ADC12B0_GCV = loop - 1;
    /***** Finish Gain Adjustment with success *****/
    ADC12B0_MCCTRL0_AVRHSEL = 0;
```

```
ADC12B0_MCCTRL0_AVRLSEL = 0;  
return ADC_GAIN_SUCCESS;  
}
```

#### 9.5.1 コード説明

このサンプルコードはフローチャート(図 17)に従って、適切なゲイン値を見つけます。

## 10 関連ドキュメント

以下は Traveo ファミリシリーズのデータシートとハードウェアマニュアルのリンクです。

- [S6J3200 Series Datasheet \(Doc.No.002-05682\)](#)
- [S6J3200 Series Hardware Manual \(Doc.No.002-04852\)](#)
- [S6J32E/F/G Series Datasheet \(Doc.No.002-10689\)](#)
- [S6J32E/F/G Series Hardware Manual \(Doc.No.002-12500\)](#)
- [Traveo Family Hardware Manual Platform Part for S6J3200 Series \(Doc.No.002-04854\)](#)
- [S6J3310/20/30/40 Series Datasheet \(Doc.No.002-10635\)](#)
- [S6J3350 Series Datasheet \(Doc.No.002-10634\)](#)
- [S6J3310/20/30/40/50 Series Hardware Manual \(Doc.No.002-10185\)](#)
- [Trave Family HardwareManual Platform Part for S6J3310/3320/3330/3340/3350 Series \(Doc.No.002-07884\)](#)
- [S6J3360/70 Series Datasheet \(Doc.No.002-03359\)](#)
- [S6J3360/70 Series Hardware Manual \(Doc.No.002-18302\)](#)
- [Trave Family HardwareManual Platform Part for S6J3360/3370 Series \(Doc.No.002-07884\)](#)
- [S6J3400 Series Datasheet \(Doc.No.001-97829\)](#)
- [S6J3400 Series Hardware Manual \(Doc.No.002-09919\)](#)
- [Traveo Family Hardware Manual Platform Part for S6J3400 Series \(Doc.No.002-07884\)](#)
- [S6J3510 Series Datasheet \(Doc.No.002-18647\)](#)
- [S6J3510 Series Hardware Manual \(Doc.No.002-18642\)](#)
- [Traveo Family Hardware Manual Platform Part for S6J3510 Series \(Doc.No.002-07884\)](#)

## 11 改訂履歴

文書名: AN211319 - Traveo™ ファミリマイコン S6J3200/S6J3300/S6J3350/S6J3360/S6J3370/S6J3400/S6J3510 シリーズの ADC 使用方法

文書番号: 002-16936

Revision	ECN	変更者	発行日	変更内容
**	5481488	YOTS	10/20/2016	Initial Release これは英語版 002-11319 Rev. **を翻訳した日本語版 002-16936 Rev. **です。
*A	5975352	YOTS	11/24/2017	これは英語版 002-11319 Rev. *C を翻訳した日本語版 002-16936 Rev. *Aです。



## セールス、ソリューションおよび法律情報

### ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、[サイプレスのロケーション ページ](#)をご覧ください。

### 製品

ARM® Cortex® Microcontrollers	<a href="http://cypress.com/arm">cypress.com/arm</a>
車載用	<a href="http://cypress.com/automotive">cypress.com/automotive</a>
クロック&バッファ	<a href="http://cypress.com/clocks">cypress.com/clocks</a>
インターフェース	<a href="http://cypress.com/interface">cypress.com/interface</a>
IoT (モノのインターネット)	<a href="http://cypress.com/iot">cypress.com/iot</a>
メモリ	<a href="http://cypress.com/memory">cypress.com/memory</a>
マイクロコントローラ	<a href="http://cypress.com/mcu">cypress.com/mcu</a>
PSoC	<a href="http://cypress.com/psoc">cypress.com/psoc</a>
電源用 IC	<a href="http://cypress.com/pmic">cypress.com/pmic</a>
タッチ センシング	<a href="http://cypress.com/touch">cypress.com/touch</a>
USB コントローラー	<a href="http://cypress.com/usb">cypress.com/usb</a>
ワイヤレス	<a href="http://cypress.com/wireless">cypress.com/wireless</a>

### PSoC® ソリューション

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6](#)

### サイプレス開発者コミュニティ

[フォーラム](#) | [WICED IOT Forums](#) | [Projects](#) | [ビデオ](#) | [ブログ](#) | [トレーニング](#) | [Components](#)

### テクニカルサポート

[cypress.com/support](http://cypress.com/support)

Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.



Cypress Semiconductor  
 198 Champion Court  
 San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2015-2017. 本書面は、Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社 (以下「Cypress」という。) に帰属する財産である。本書面 (本書面に含まれ又は言及されているあらゆるソフトウェア若しくはファームウェア (以下「本ソフトウェア」という。)) を含む) は、アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し、本段落で特に記載されているものを除き、その特許権、著作権、商標権又はその他の知的財産権のライセンスを一切許諾しない。本ソフトウェアにライセンス契約書が伴っておらず、かつ Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意がない場合、Cypress は、(1) 本ソフトウェアの著作権に基づき、(a) ソースコード形式で提供されている本ソフトウェアについて、Cypress ハードウェア製品と共に用いるためにのみ、かつ組織内部でのみ、本ソフトウェアの修正及び複製を行うこと、並びに (b) Cypress のハードウェア製品ユニットに用いるためにのみ、(直接又は再販売者及び販売代理店を介して間接のいずれかで) 本ソフトウェアをバイナリーコード形式で外部エンドユーザーに配布すること、並びに (2) 本ソフトウェア (Cypress により提供され、修正がなされていないもの) が抵触する Cypress の特許権のクレームに基づき、Cypress ハードウェア製品と共に用いるためにのみ、本ソフトウェアの作成、利用、配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス (サブライセンスの権利を除く) を付与する。本ソフトウェアのその他の使用、複製、修正、変換又はコンパイルを禁止する。

**適用される法律により許される範囲内で、Cypress は、本書面又はいかなる本ソフトウェア若しくはこれに伴うハードウェアに関しても、明示又は黙示を問わず、いかなる保証 (商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない) も行わない。**適用される法律により許される範囲内で、Cypress は、別途通知することなく、本書面を変更する権利を留保する。Cypress は、本書面に記載のある、いかなる製品若しくは回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報 (あらゆるサンプルデザイン情報又はプログラムコードを含む) は、参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計、プログラム、かつテストすることは、本書面のユーザーの責任において行われるものとする。Cypress 製品は、兵器、兵器システム、原子力施設、生命維持装置若しくは生命維持システム、蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム、汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分としての使用、又は装置若しくはシステムの不具合が人身傷害、死亡若しくは物的損害を生じさせるようなその他の使用 (以下「本目的外使用」という。)) のためには設計、意図又は承認されていない。重要な構成部分とは、その不具合が装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できるような装置若しくはシステムのあらゆる構成部分をいう。Cypress 製品のあらゆる本目的外使用から生じ、若しくは本目的外使用に関連するいかなる請求、損害又はその他の責任についても、Cypress はその全部又は一部を問わず一切の責任を負わず、かつ Cypress はそれら一切から本書により免除される。Cypress は Cypress 製品の本来目的外使用から生じ又は本目的外使用に関連するあらゆる請求、費用、損害及びその他の責任 (人身傷害又は死亡に基づく請求を含む) から免責補償される。

Cypress, Cypress のロゴ, Spansion, Spansion のロゴ及びこれらの組み合わせ, WICED, PSoC, Capsense, EZ-USB, F-RAM, 及び Traveo は、米国及びその他の国における Cypress の商標又は登録商標である。Cypress のより完全な商標のリストは、[cypress.com](http://cypress.com) を参照すること。その他の名称及びブランドは、それぞれの権利者の財産として権利主張がなされている可能性がある。