

**Myway**

---

**SimCoder™**

---

**User's Guide**

**Powersim Inc.**

Mywayプラス株式会社

## SimCoder User's Guide

Version 3.2

Release 1

Copyright © 2008-2011 Powersim Inc., Myway Plus Corporation

All rights reserved. No part of this manual may be photocopied or reproduced in any form or by any means without the written permission of Powersim Inc and Myway Plus Corporation.

### *Disclaimer*

Powersim Inc. (Powersim) and Myway Plus Corporation (Myway) make no representation or warranty with respect to the adequacy or accuracy of this documentation or the software which it describes. In no event will Powersim and Myway or their direct or indirect supplies be liable for any damages whatsoever including, but not limited to, direct, indirect, incidental, or consequential damages of any character including, without limitation, loss of business profits, data, business information, or any and all other commercial damages or losses, or for any damages in excess of the list price for the license to the software and documentation.

### お問い合わせ先

Myway プラス株式会社

〒222-0022 神奈川県横浜市西区花咲町 6-145 横浜花咲ビル

Tel 045-548-8836, Fax 045-548-8832

Email: [sales@myway.co.jp](mailto:sales@myway.co.jp)

URL: <http://www.myway.co.jp/>

## 目次

1	SimCoder について .....	6
1.1	まえがき .....	6
1.2	ハードウェア .....	7
1.3	コード生成用の素子 .....	7
2	コード生成方法 .....	8
2.1	自動コード生成方法 .....	8
2.2	ハードウェア素子なしのシステム .....	9
2.3	連続系システム .....	9
2.4	離散系システム .....	9
2.5	ハードウェア素子付きシステム .....	10
2.6	C コード生成 .....	12
3	サブシステムのコード生成 .....	18
3.1	サブシステム .....	18
3.2	コード生成 .....	20
4	イベントコントロール .....	21
4.1	基本概念 .....	21
4.2	イベントコントロール素子 .....	22
4.3	イベント付きサブ回路の制限 .....	23
5	SimCoder ライブラリ .....	26
5.1	標準 PSIM ライブラリの素子 .....	26
5.1.1	パラメータファイルブロックを用いたグローバルパラメータ設定 .....	29
5.1.2	のこぎり波生成 .....	29
5.2	イベントコントロール素子 .....	29
5.2.1	入力イベント .....	29
5.2.2	出力イベント .....	30
5.2.3	デフォルトイベント .....	31
5.2.4	イベント接続 .....	31
5.2.5	イベントブロック初回実行フラグ .....	31
5.3	グローバル変数 .....	32
5.4	ハードウェア割り込み .....	34
6	TI F28335 Hardware Target .....	36
6.1	PWM 生成器 .....	38
6.2	Start PWM と Stop PWM .....	47
6.3	トリップゾーンとトリップゾーンステート .....	47

6.4	A/D 変換器 .....	49
6.5	デジタル入力とデジタル出力 .....	53
6.6	UP/DOWN カウンタ .....	54
6.7	エンコーダとエンコーダステート .....	54
6.8	キャプチャとキャプチャステート .....	56
6.9	シリアル通信インターフェース (SCI) .....	56
6.9.1	SCI 設定 .....	57
6.9.2	SCI 入力 .....	57
6.9.3	SCI 出力 .....	58
6.10	シリアルペリフェラルインターフェース(SPI) .....	58
6.10.1	SPI 設定 .....	59
6.10.2	SPI デバイス .....	59
6.10.3	SPI 入力 .....	61
6.10.4	SPI 出力 .....	62
6.11	DSP 設定 .....	63
6.12	ハードウェアボードコンフィギュレーション .....	63
6.13	プロジェクト設定とメモリ配置 .....	64
7	PE-PRO/F28335 Hardware Target .....	67
7.1	PWM 生成器 .....	67
7.2	Start PWM と Stop PWM .....	70
7.3	A/D 変換器 .....	70
7.4	D/A 変換器 .....	72
7.5	デジタル入力/エンコーダ/トリップゾーン .....	73
7.6	デジタル出力/単純 PWM .....	74
8	PE-Expert3 Hardware Target .....	76
8.1	PEV ボード .....	76
8.1.1	PWM 発生器 .....	76
8.1.2	A/D 変換器 .....	79
8.1.3	デジタル入力・キャプチャ・カウンタ .....	80
8.1.4	デジタル出力 .....	81
8.1.5	エンコーダ .....	82
8.2	PE-Expert3 ランタイムライブラリ .....	82
9	General Hardware Target .....	83
9.1	PWM 発生器 .....	83
9.2	A/D 変換器 .....	85
9.3	D/A 変換器 .....	88

9.4	デジタル入出力 .....	88
9.5	エンコーダ .....	89
9.6	キャプチャ .....	90
9.7	割り込み.....	91

### 1.1 まえがき

SimCoderは制御システムのコードを自動的に生成するためのPSIMシミュレータの追加モジュールです。SimCoderを利用して、PSIMシミュレータでシステムのシミュレーションができ、Myway プラス製の実機コントローラ PE-Expert3及びPE-PRO/F28335またはTI社製高速浮動小数点型 DSPTMS320F28335とその他汎用DSPのハードウェアプラットフォームに合わせたCコードを自動的に生成することができます。

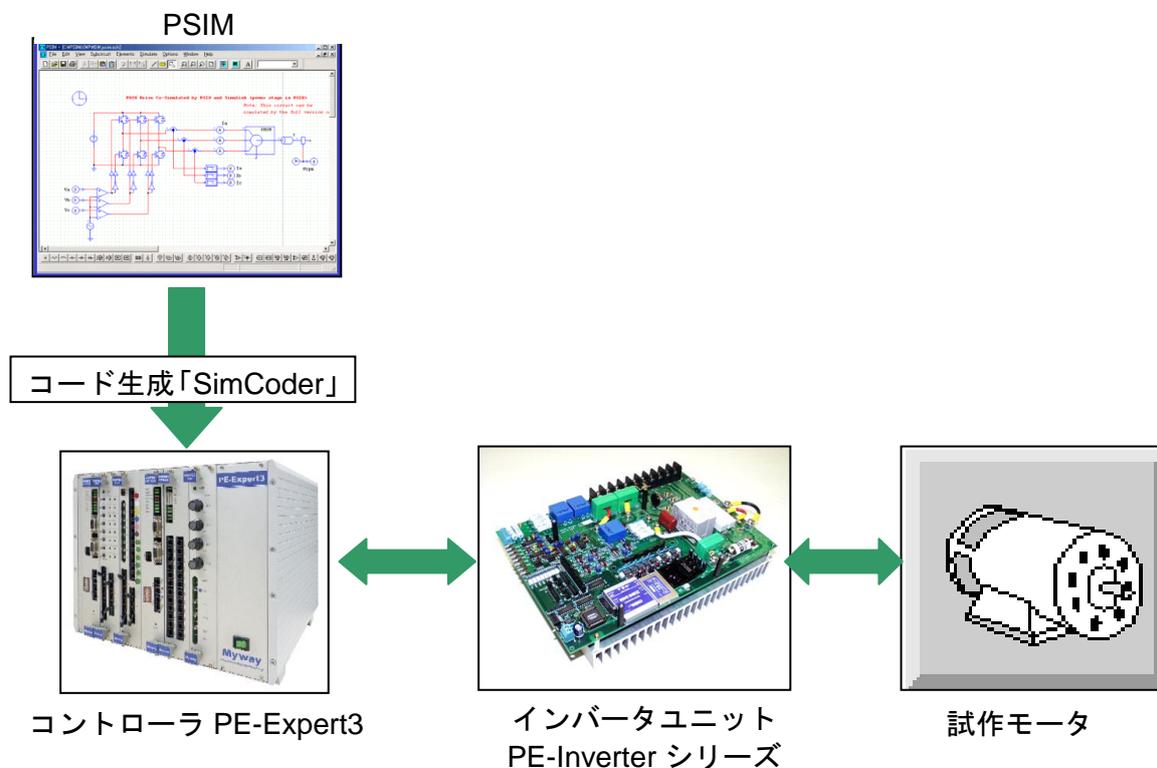


図 1.1 SimCoder と PE-Expert3 を利用した実験システム

SimCoderを利用して自動的にコードを作成すると、開発時間とコストを大幅に縮小することが可能になります。このマニュアルは、SimCoderを使用する方法について記述します。

## 1.2 ハードウェア

SimCoderは、下記のハードウェア向けのコードを生成することができます。

- **PE-Expert3**: Mywayプラス製のPE-Expert3 DSP開発プラットフォームで使用できるCコードを生成します。**Elements → SimCoder for Code Generation → PE-Expert3 Target**メニューを選択してPE-Expert3に対応している素子を選ぶことができます。（利用するためにはSimCoderモジュールに加えてPE-Expert3 Targetモジュールが追加オプションとして必要になります。）
- **PE-PRO/F28335**: Mywayプラス製のPE-PRO/F28335 DSP開発プラットフォームで使用できるCコードを生成します。**Elements → SimCoder for Code Generation → PE-Pro/F28335 Target**メニューを選択してPE-PRO/F28335に対応している素子を選ぶことができます。（利用するためにはSimCoderモジュールに加えてPE-PRO/F28335 Targetモジュールが追加オプションとして必要になります。）
- **TI社製DSP TMS320F28335**: TI社製 TMS320F28335DSP開発プラットフォームで汎用的に使用できるCコードを生成します。PE-PRO/F28335以外のTMS320F28335搭載のコントローラボードに対してコードを生成したい場合に利用します。**Elements → SimCoder for Code Generation → TI F28335 Target**メニューを選択してTMS320F28335に対応している素子を選ぶことができます。（利用するためにはSimCoderモジュールに加えてTI F28335 Targetモジュールが追加オプションとして必要になります。）
- **汎用DSP**: 汎用ハードウェアプラットフォームで使用できるCコードの雛形を生成することもできます。ユーザは生成された汎用Cコードを修正して特定のハードウェア(DSP・ $\mu$ C等)で使用することができます。**Elements → SimCoder for Code Generation → General Hardware Target**メニューを選択して汎用ハードウェアに対応している素子を選ぶことができます。（利用するためにはSimCoderモジュールに加えてGeneral HardwareTargetモジュールが追加オプションとして必要になります。）

## 1.3 コード生成用の素子

**Elements → Event Control**および**Elements → SimCoder for Code Generation**メニューのすべての素子がコード生成に使用できます。更に、PSIM標準ライブラリのいくつかの素子もコード生成時に使用することができます。

**Options → Settings**メニューの「*Show image next to elements that can be used for code generation*」オプションボックスをチェックすると、コード生成に使用できるPSIM標準ライブラリの素子のところにマークが表示されます。

また、各ハードウェアターゲットによって下記のようなマークも付きます。

- PE-Expert3専用素子: 
- PE-PRO/F28335専用素子: 
- TI F28335専用素子: 
- 汎用DSP専用素子: 

### 2.1 自動コード生成方法

通常、SimCoderを利用してコード生成する場合、次の4つのステップを行います。

- ・連続系の制御システムによるシステムのシミュレーション
- ・離散系の制御システムによるシステムのシミュレーション
- ・対象制御ハードウェアがない場合は、制御回路部をサブ回路にしてコード生成
- ・対象制御ハードウェアが決まっていれば、ハードウェア素子を追加してコード生成

しかし、最初の2つのステップは省略することもできます。例えば、PSIMで回路図を作成して、直接コードを生成することも可能であり、必ずしもシミュレーションを行う必要はありません。制御システムを離散系で作成しなければSimCoderでコード生成できないため、デジタルコントロールモジュールが必要です。

降圧チョップパを利用してSimCoderのコード生成方法を以下に具体的に説明します。

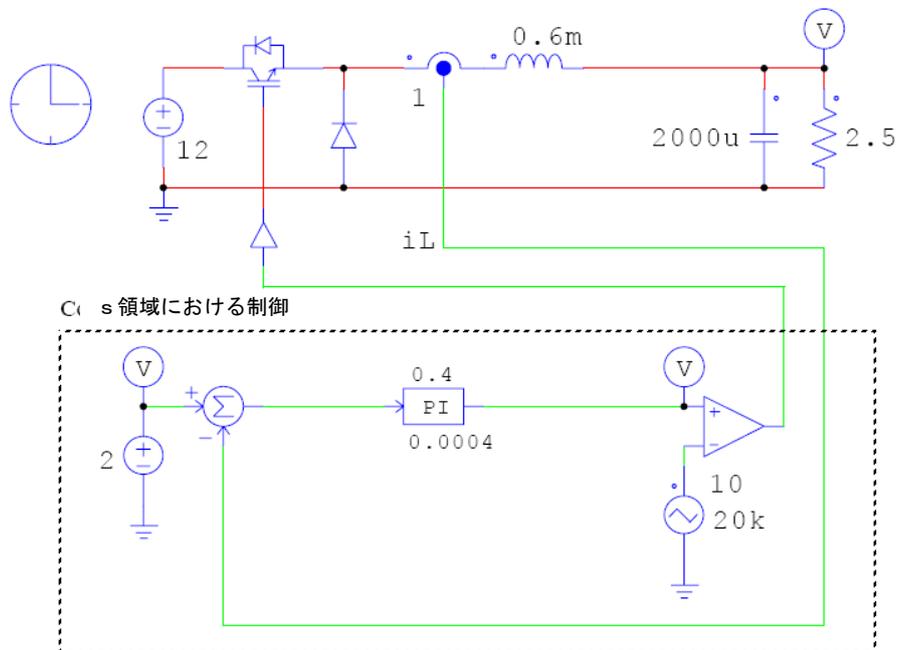


図2.1 降圧チョップパ制御回路（s領域）

## 2.2 ハードウェア素子なしのシステム

### 2.3 連続系システム

通常は、最初に連続系でシミュレーション回路を作成します。図2.1は電流制御付の降圧チョッパ回路です。この回路では、PIコントローラは連続系の  $s$  領域で設計しています。このコントローラのPIゲイン、 $k$ と時定数  $T$ はそれぞれ0.4と0.0004です。スイッチング周波数は20kHzです。

この例の目的は、点線内の制御回路についてCコードを自動的に生成することです。コードを生成するために最初に行わなければならないことは連続系 ( $s$  領域) のアナログPIコントローラを離散系 ( $z$  領域) のデジタルPIコントローラに変換することです。

### 2.4 離散系システム

アナログコントローラのパラメータをデジタルコントローラのパラメータに変換するために、デジタルコントロールモジュールの *s2z Converter* プログラムを使用します。 *s2z Converter* を使用するには、PSIMを起動して **Utilities -> s2z Converter** を選択してください。

アナログコントローラをデジタルコントローラに変換するとき、色々な変換法を使用することができます。最も一般的な方法はバイリニア変換（双一次変換 タスティン変換、台形差分法とも呼ばれる）およびバックワードオイラー法（後退オイラー法）です。この例では、バックワードオイラー法を使用します。サンプリング周波数はスイッチング周波数と同じ20kHzとして *s2z Converter* で変換すると、デジタルPIコントローラのパラメータは以下のようになります：

比例部分のゲイン、 $k1 = 0.4$

積分部分のゲイン、 $k2 = 1000$

デジタルコントローラを用いた回路を以下に示します。

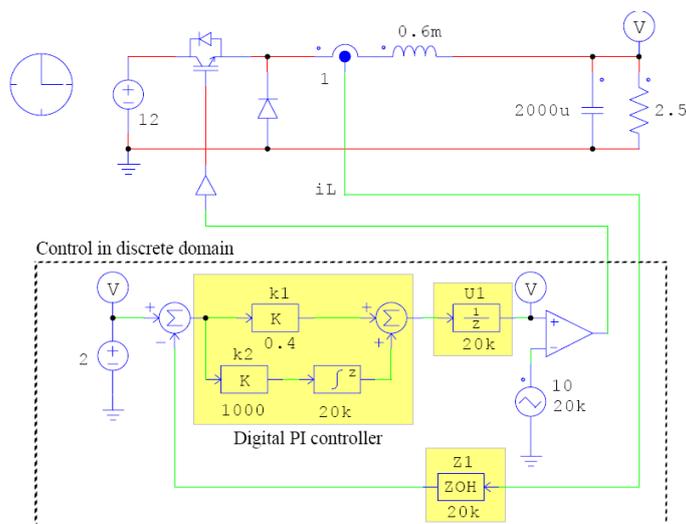


図2.2 降圧チョッパ制御回路 ( $z$  領域)

図2.1と図2.2を比較すると、z領域の回路では黄色のハイライトで示した3箇所の変更があります。アナログPIコントローラはデジタルPIコントローラに置き換えられます。デジタル積分器の「Algorithm Flag」は1、サンプリング周波数は20kHzに設定します。ゲインk1とk2は上に述べた通りです。更に、ゼロ次ホールドブロックZ1は、デジタル制御システムではフィードバック電流*i<sub>L</sub>*をA/D変換してサンプリングするために使用されます。最後に、単位時間遅れブロックU1は、実際のデジタル制御の1サンプリング遅れをモデル化するため使用されています。

バックワードオイラー法を利用した積分器の入出力関係は以下のように表現できます。

$$y(n) = y(n-1) + T_s * u(n)$$

ここで、*y(n)*と*u(n)*は現在の時間の出力と入力であり、*y(n-1)*は1サンプリング前の出力です。上の式に基づいて、離散系の積分器を加算器と単位時間遅れブロックで以下の図2.3のように置き換えることができます。

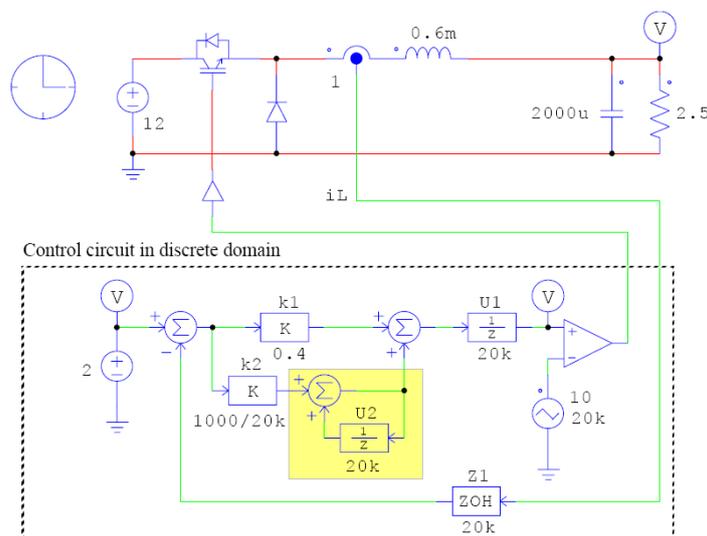


図2.3 バックワードオイラー法に基づいた降圧チョッパ制御回路（z領域）

比例ゲインk2は、Tsをサンプリング周波数20kHzで割った値とします。この回路の利点は、積分器の積分動作の開始/停止を簡単に制御することができることです。（詳しくは図2.6で説明します。）

## 2.5 ハードウェア素子付きシステム

SimCoderは対象制御ハードウェア用のコードを生成します。コード生成の前に、適切なハードウェア素子を使用した回路を作成します。また、ハードウェア素子の各設定値は、実際のハードウェアの範囲を考慮して、必要に応じて値をスケーリングする必要があります。

ハードウェア素子付きシステムのシミュレーション回路の作成は、以下のように行います。

- A/D変換器、デジタル入出力等を追加する
- コンパレータを使用したPWM発生回路をハードウェアPWM発生器に置き換える
- 必要に応じてイベントシーケンス制御を追加する

図2.4はPE-Expert3用のハードウェア素子を使用した回路であり、ハードウェア素子は黄色でハイライトされています。

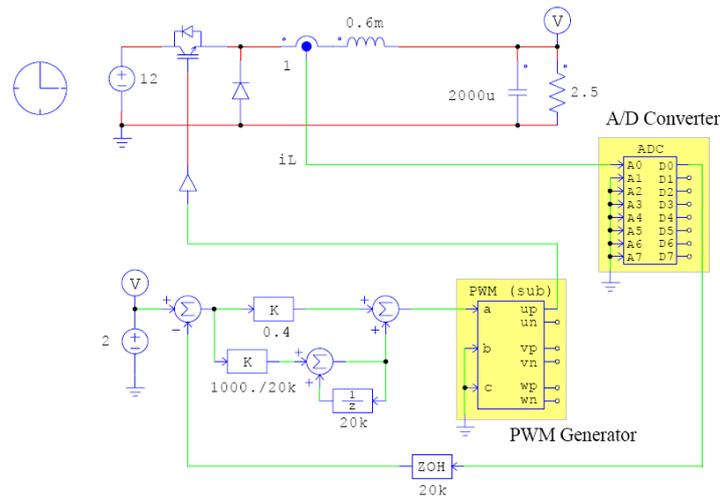


図2.4 ハードウェア素子を使用した降圧チョッパ制御回路

図2.3に対して、図2.4では以下のように変更されています。

- 電流センサの後に、A/D変換器を置いています。このA/D変換器の入力範囲に注意する必要があります。電流センサ出力がA/D変換器の入力範囲に収まらない場合は、電流センサ信号をスケールしなければなりません。この例の場合は、電流センサ出力はA/D変換器の範囲内(+5V~-5V)にあるためスケールは不要で、A/D変換器のパラメータ「AD0 Range」は5に設定します(5に設定するとA/Dのゲインは1になります)。
- 図2.3におけるコンパレータとキャリア信号は、ハードウェアPWM発生器に置き換えられています。ハードウェアPWM発生器には1サンプリング周期遅れを含んでいるので、単位時間遅れブロックが削除されています。ハードウェアPWM発生器では、パラメータ「Carrier frequency」は20kHz、「Peak Value」は10V、「Start PWM at beginning」は「Start」に設定します。
- 「Simulation Control」メニューで、「Hardware Type」は「Myway\_PE Expert3」に設定します。

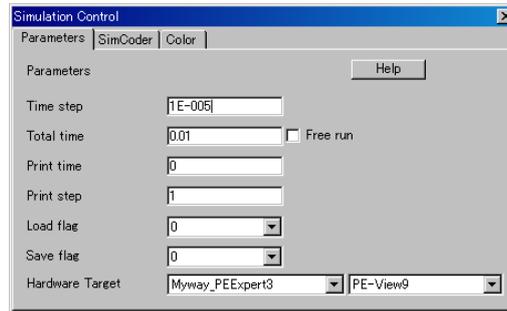


図2.5 ハードウェアタイプの設定

## 2.6 Cコード生成

ハードウェア素子を制御システムに追加後、**Simulate -> Generate Code** を選択してCコードを生成することができます。次に、生成されたプロジェクトファイルをMywayプラスのPE-View環境にロードして、コードをコンパイルし、それをハードウェアにアップロードすることができます。生成されたコードの例を以下に示します。

```

/*****
// This code is created by SimCoder Version 1.0 for Myway PE-Expert3
//
// SimCoder is copyright by Powersim Inc., 2008
//
// Date: March 14, 2008 15:10:04
*****/

#include.....<math.h>
#include.....<mwio3.h>
#define....._DEBUG * Comment it out if not watching data from PE_VIEW */

interrupt void Task();

float .....fGbliref = 0.0;
float      fGblUDELAY1 = 0.0;
```

If defined, values of voltage probes can be watched in PE-View.

interrupt void Task()

The main interrupt service routine for 20 kHz

```
{
    float fVDC2, fPEV_ADC1, fPEV_ADC1_1, fPEV_ADC1_2, fPEV_ADC1_3, fZOH3, fSUM1,
    fP2;
    float fSUMP3, fUDELAY1, fP1, fSUMP1, fPEV_PWM_11Div0, fPEV_PWM_11Div1, fPE
    V_PWM_11Div5;
    float fPEV_PWM_11Div2, fPEV_PWM_11Div6, fPEV_PWM_11Div3, fPEV_PWM_11Div7;
    pev_ad_start(0, 0);

    fUDELAY1 = fGblUDELAY1;
    fVDC2 = 2;

#ifdef _DEBUG
    fGbliref = fVDC2;
#endif

    while (pev_ad_in_st(0, 0));
    pev_ad_in_grp(0, 0, &fPEV_ADC1, &fPEV_ADC1_1, &fPEV_ADC1_2, &fPEV_ADC1_3);

    fZOH3 = fPEV_ADC1;
    fSUM1 = fVDC2 - fZOH3;
    fP2 = fSUM1 * (1000./20000);
    fSUMP3 = fP2 + fUDELAY1;
    fGblUDELAY1 = fSUMP3;
    fP1 = fSUM1 * 0.4;
    fSUMP1 = fP1 + fSUMP3;
    fPEV_PWM_11Div0 = ((0+10)/2.0);
    fPEV_PWM_11Div1 = fSUMP1 - fPEV_PWM_11Div0;
    fPEV_PWM_11Div5 = fPEV_PWM_11Div1 * (2.0/(10));
    fPEV_PWM_11Div2 = 0 - fPEV_PWM_11Div0;
    fPEV_PWM_11Div6 = fPEV_PWM_11Div2 * (2.0/(10));
    fPEV_PWM_11Div3 = 0 - fPEV_PWM_11Div0;
    fPEV_PWM_11Div7 = fPEV_PWM_11Div3 * (2.0/(10));

    pev_inverter_set_uvw(0, 0, fPEV_PWM_11Div5, fPEV_PWM_11Div6, fPEV_PWM_11Div7);

#ifdef _DEBUG
    watch_data();
#endif
}
```

`void Initialize(void)`      The initialization routine

```
{
    pev_init(0);
    pev_ad_set_range(0, 0, 5, 1, 1, 1);
    pev_ad_set_range(0, 1, 1, 1, 1, 1);
    pev_inverter_init(0, 0, 20000, (4e-6)*1E9);
    pev_inverter_set_syncint(0, 0.0);
    int5_init_vector(Task);
    int5_enable_int();
    pev_inverter_enable_up_int5(0);
    pev_inverter_set_uvw(0, 0, 0, 0, 0);
    wait(100);
    pev_inverter_stop_pwm(0, 0);
}
```

`void main()`      The main program

```
{
#ifdef    _DEBUG
    watch_init();
#endif

    int_disable();
    Initialize();
    int_enable();
    while (1) {
    }
}
```

生成されたコードは以下の構造になっています。

`void main ()` : メイン関数であり、初期化ルーチンを呼んで無限ループを走らせます。

`void Initialize ()` : 初期化関数であり、ハードウェアを初期化します。

`Interrupt void Task ()` : 割り込み実行関数であり、20kHzごとにこの関数が呼び出されます。

この例では、すべての制御ブロックは20kHzのサンプリングレートで実行されます。制御システムに他のサンプリングレートがあるときは、もう1つの割り込み実行関数が作成されます。制御システムでサンプリングレートが付いてないブロックがある場合、対応するコードはメイン関数内に生成されます。

また、テストとデバッグのために、コードにはデバッグフラグ「\_DEBUG」が作成されます。これにより、PSIM回路図の電圧プローブの値をPE-Viewのリアルタイムスコープで見ることが可能になります。この例では、2V電圧源につないである電圧プローブの値をPE-Viewのスコープに表示することができます。

コードと必要なプロジェクトファイルは PSIM 回路ファイルと同じフォルダのサブフォルダーに保存されます。このプロジェクトファイルを PE-View 環境にロードして、コードをコンパイルして、それをハードウェアにアップロードすることができます。

## 2.5 イベントコントロール付きシステム

制御システムでは、イベントコントロールが必要な場合があります。すなわち、ある条件が満たされた時、システムがある状態から別の状態へ移行する場合があります。SimCoderはPSIMのサブ回路を利用してイベントコントロールを行います。イベントコントロールの詳細については第4章を参照してください。

StopモードとRunモードの2つのモードがあるイベントコントロール付きシステムを次のように作成します。

- システムのstart/stopを制御するために手動スイッチを加えます。これにより、システムにはStopモードとRunモードの2つモードができます。この手動スイッチの位置を変化させてあるモードから別のモードに移行します。
- Stopモードでは、積分器は働かないようにし、その出力を0にリセットします。イベントコントロール付きシステムの例を図2.6に示します。図において、S1およびS2はサブ回路であり、このサブ回路S1およびS2の内容を図2.7に示します。
- このシステムには、サブ回路S1によって表わされるStopモードと、サブ回路S2によって表わされるRunモードの2つのモードがあります。デフォルトの運転モードはStopモードであり、これは、サブ回路S1のポートEIN1にデフォルトイベント素子を接続することによって定義されます。
- サブ回路S1には2つの入力イベントポートEIN1とEIN2、1つの出力イベントポートEORun、1つの入力信号ポートRunSwitch、1つの出力信号ポートRunModeがあります。サブ回路S2には1つの入力イベントポートEIN、1つの出力イベントポートEOSTOP、1つの出力信号ポートRunModeがあります。
- StopモードとRunモードとを相互に移行するために条件を定義します。条件はグローバル変数RunSWを使用します(グローバル変数についてセクション5.2を参照してください)。この変数はグローバル変数素子をデジタル入力素子の出力に接続することによって定義されます。
- 押しボタンスイッチSW1のオン/オフ状態はハードウェアデジタル入力素子によって検出され、スイッチがオフの場合、デジタル入力とグローバル変数RunSWは1(HIGH)になりシステムがRunモードへ移動する。RunSWが0(LOW)の時は、Stopモードになります。
- マルチプレクサMUX1は、Stopモードの時に積分回路が積分動作しないように追加されています。つまり、Stopモードの時RunSWは0であり積分回路は動作しません。RunモードでRunSWが1の時、積分回路は積分動作を行います。

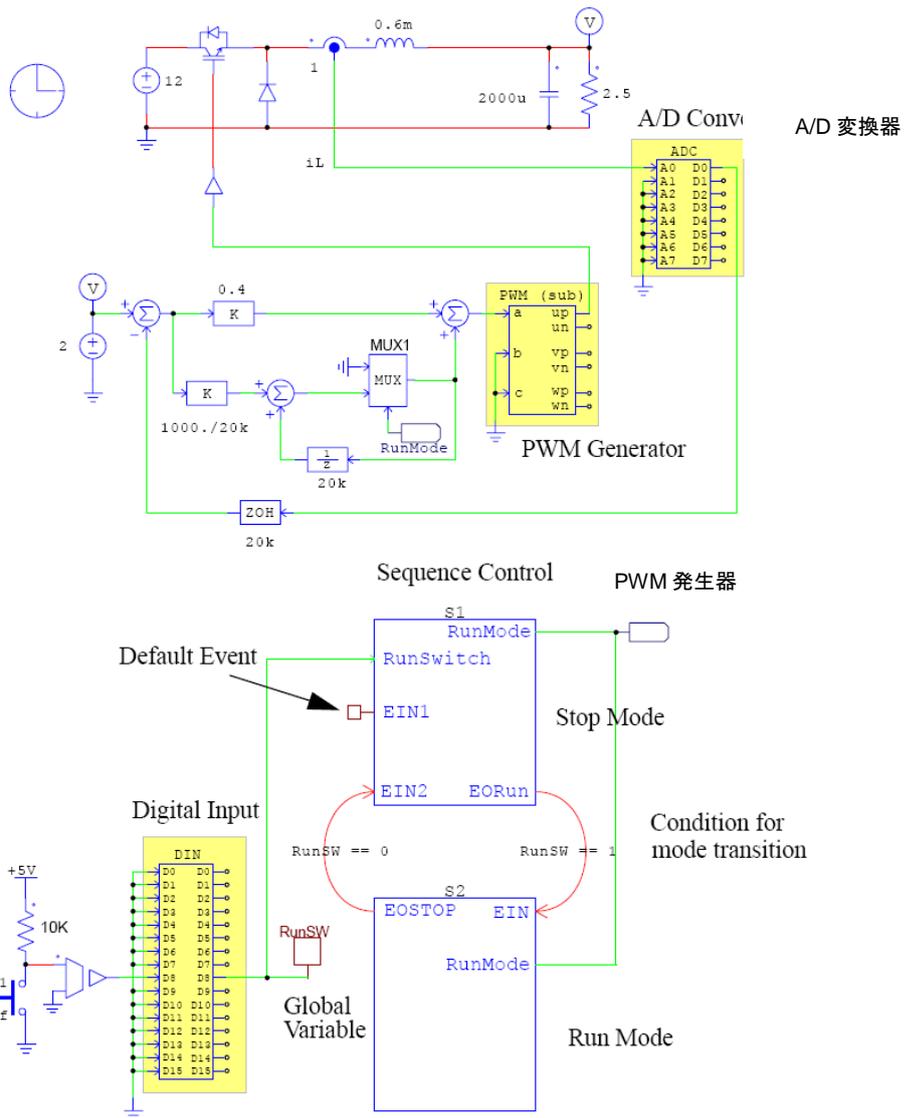


図2.6 シーケンス制御付き降圧チョップ制御回路

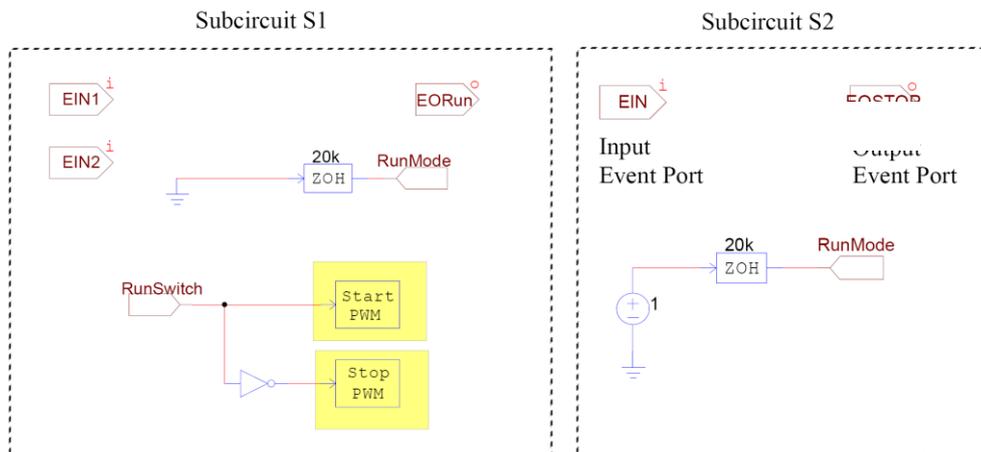


図 2.7 Run モードと Stop モードのサブ回路

システムは以下のように動作します：

- 手動スイッチのオン／オフ状態はハードウェアデジタル入力素子によって検出され、その後、この信号はポート「RunSwitch」を通過してサブ回路S1に送られます。同じ信号がグローバル変数「RunSw」として指定されます。グローバル変数は条件文の中で使用することができます。
- システムは「Stopモード」から開始します。これは、サブ回路S1のポートEIN1にデフォルトイベント素子を接続することで定義しています。条件「RunSw == 1」が成立すると、システムは「Stopモード」から「Runモード」へ移行します。これは、S1の出カイベントポートEORunとS2の入カイベントポートEINの接続によって定義されています。
- 「Runモード」にいるとき、条件「RunSw == 0」が成立すると、システムは「Runモード」から「Stopモード」へ移行します。これはS2の出カイベントポートEOSTOPとS1の入カイベントポートEIN2の接続によって定義されています。
- 「Stopモード」では、RunModeを「0」に設定しています。また、RunSwitchが「0」ならばハードウェアPWM発生器は停止していますが、RunSwitchが「1」に変更されると、PWMを開始し同時に「Stopモード」から「Runモード」へ移行します。
- 「Runモード」では、積分器の動作を有効にするためにRunModelは「1」に設定されています。

## 3 サブシステムのコード生成

---

### 3.1 サブシステム

SimCoderでは、サブシステムを含むシステム、またはサブシステム単独でコード生成することができます。しかし、いくつかの制限があるため以下に説明します。

- サブ回路には双方向の入出力信号ポートは使用できません。入力には入力信号ポート、出力には出力信号ポートを使用します。
- ハードウェア入出力素子（A/D変換器やPWM発生器等）およびハードウェア割り込み素子はサブシステムでは使用できません。これらは、トップレベルのメイン回路でのみ使用可能です。
- サブシステムの入力にサンプリングレートがあり、サブシステム内の回路からこのレートを得ることができないとき、入力のサンプリングレートを定義するためにゼロ次ホールドブロックを接続しなければなりません。ゼロ次ホールドブロックが使用されないと、この入力およびこの入力に接続するその後のブロックはサンプリングレートなしで扱われます。

例えば、サブシステムの外でサンプリングレートが定義され、サブシステム内にはサンプリングレートを示すような離散的な素子がない場合、同じサンプリングレートのゼロ次ホールドブロックを入力に接続しなければなりません。

サブシステムの入力にゼロ次ホールドブロックが接続されていない時、SimCoderはサブシステムが接続されているブロックからサンプリングレートを引き出します。しかし、あいまいさを避けるために、入力にゼロ次ホールドブロックを接続してサンプリングレートを定義することをお勧めします。

サブシステムのコードを生成する方法を説明するために、セクション2.5のシステムの電流フィードバックとPIコントローラ部のコードを生成します。図3.1のように、コード生成する部分をサブ回路S3として回路図を作成します。サブ回路S3には2つの入力*iL*、*RunMode*、そして1つの出力*Vm*があります。入力*iL*および*RunMode*のサンプリングレートは20kHzであり、図3.2のようにサブ回路S3の入力はゼロ次ホールドブロックと接続されています。

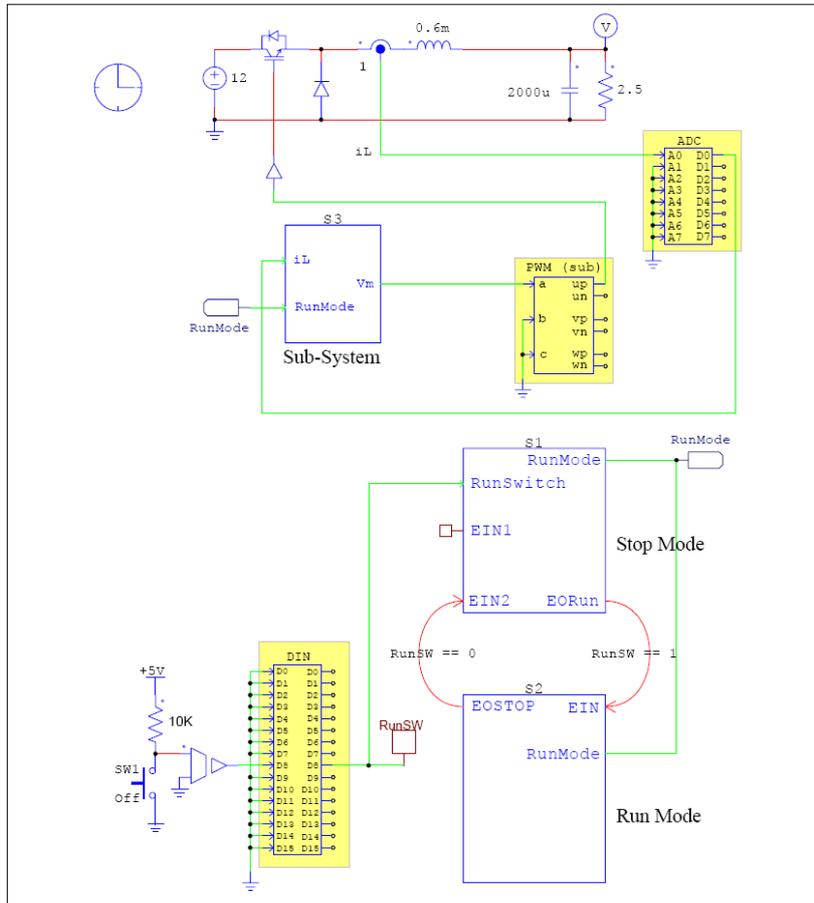


図3.1 降圧チョップパのサブシステムのコード生成

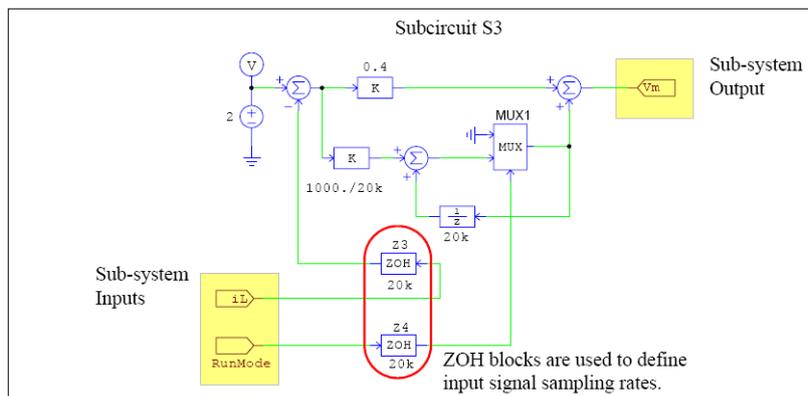


図3.2 サブシステムの制御ブロック

## 3.2 コード生成

サブシステムのコードを生成するためには、サブ回路を右クリックし、Attributesを選びSubcircuit VariablesタブでGenerate Codeをクリックします。

このサブシステムには、1つのサンプリングレートしかありませんので、生成コードには、1つの関数しかありません。変数 $fIn0$ と $fIn1$ はサブ回路の2つの入力 $iL$ とRunModeに対応しており、また、変数 $fOut0$ はサブ回路の出力 $Vm$ に対応しています。全体システムの生成コードと異なって、サブシステムのコードにはmain関数および初期化ルーチンがありません。

```
float      fGblS3_iref = 0.0;
float      fGblS3_UDELAY1 = 0.0;

void TaskS3(float fIn0, float fIn1, float *fOut0)
{
    float fS3_VDC2, fS3_Z3, fS3_SUM1, fS3_P1, fS3_P2, fS3_SUMP3, fS3_Z4, fS3_MUX1;
    float fS3_UDELAY1;

    fS3_UDELAY1 = fGblS3_UDELAY1;
    fS3_VDC2 = 2;
    fS3_Z3 = fIn0;
    fS3_SUM1 = fS3_VDC2 - fS3_Z3;
    fS3_P1 = fS3_SUM1 * 0.4;
    fS3_P2 = fS3_SUM1 * (1000./20000);
    fS3_SUMP3 = fS3_P2 + fS3_UDELAY1;
    fS3_Z4 = fIn1;
    fS3_MUX1 = (fS3_Z4 == 0) ? 0 : fS3_SUMP3;
    *fOut0 = fS3_P1 + fS3_MUX1;

#ifdef   _DEBUG
    fGblS3_iref = fS3_VDC2;
#endif

    fGblS3_UDELAY1 = fS3_MUX1;
}
```

### 4.1 基本概念

イベントは、システムがある状態から別の状態へ移行することを記述するために使用します。以下の図4.1はシステムの状態とそれらの間の状態遷移を示しています。

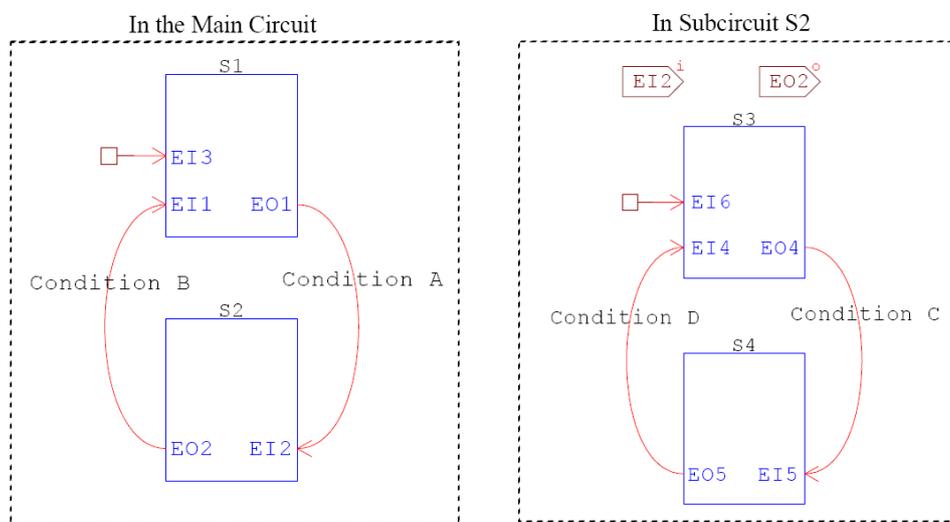


図4.1 状態遷移図

メイン回路では、2つの状態(S1およびS2)があり、状態はサブ回路で実現されています。S1には2つの入力イベントポートEI1とEI3、1つの出力イベントポートEO1があります。S2には1つの入力ポートEI2および1つの出力イベントポートEO2があります。最初は、S1がデフォルト状態になっています。これは入力イベントポートEI3にデフォルトイベント素子を接続することで定義されます。

S1の出力イベントポートEO1はS2の入力イベントポートEI2に接続されています。条件Aが成立すると、システムはS1からS2に移行します。同様に、S2の出力イベントポートEO2はS1の入力イベントポートEI1に接続されています。この場合、条件Bが成立すると、システムはS2からS1に移行します。

右側のシステムはサブ回路S2の内容を示しています。サブ回路S2には2つの状態、S3とS4があります。システムはサブ回路S2へ移動すると、デフォルトでS3になります。条件Cが成立すると、S3からS4に移行します。条件Dが成立すると、S4に移行します。このようにシステムが持つことができる状態の数に制限はありません。

## 4.2 イベントコントロール素子

イベントと状態遷移を制御するためには次の素子を使用します。

- 入力イベントポート
- 出力イベントポート
- デフォルトイベント素子
- ハードウェア割り込み素子（セクション5.3.2を参照）
- グローバル変数

サブ回路に入力イベントポートを追加すると、このサブ回路への移行を許可するポートが作成されます。同様に、サブ回路に出力イベントポートを追加すると、このサブ回路から他のサブ回路への移行を許可するポートが作成されます。入力イベントポート(EI1)と出力イベントポート(EO1)を追加した場合のサブ回路は図4.2のようになります。



図4.2 入出力イベントポート付きサブ回路

図のようにイベントポートは正方形、信号ポートは円形のノードシンボルで表示されます。

Editメニューの「Event Connection」機能を選択して、入力イベントポートには出力イベントポートかハードウェア割り込み素子を接続することができます。出力イベントポートには入力イベントポートを接続することができます。入出力イベントポートやハードウェア割り込み素子は他のタイプのノードに接続することはできません。

出力イベントポートに関しては、移行条件を定義しなければなりません。例としてサブ回路S3の出力イベントポートEO1のプロパティウインドウを図4.3に示します。

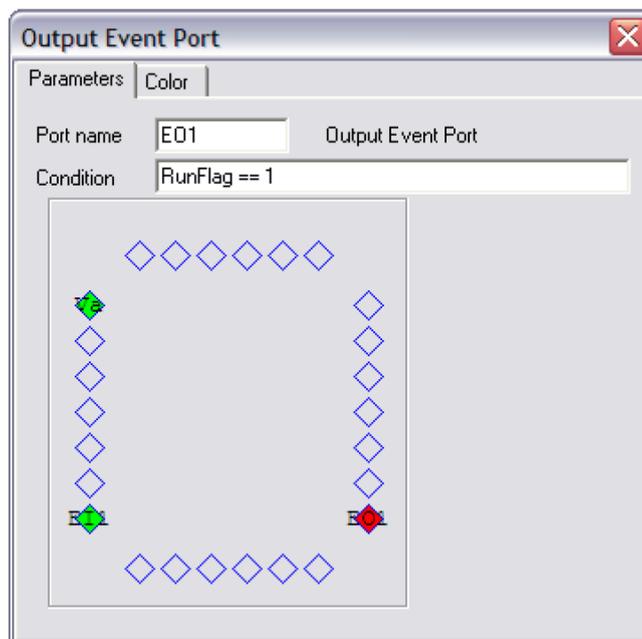


図4.3 出カイベントポート

移行条件「RunFlag==1」が成立すれば、他のサブ回路へ移行します。条件式は以下の例のようにCコードで表記します。

```
(RunFlag == 1) && (FlagA >= 250.) || (FlagB < Vconst)
```

条件式にはグローバル変数、数値、パラメータファイルで定義する定数、またはメイン回路からサブ回路に送られた変数を使用することができます。グローバル変数を作成するには、グローバル変数素子をノードに接続してください。

### 4.3 イベント付きサブ回路の制限

入力または出カイベントポートを含むサブ回路はイベント付きサブ回路であり、イベント付きサブ回路の中のすべてがイベントの特性を引き継ぎます。すなわち、サブ回路Bの中にサブ回路Aがあって、サブ回路Bがイベント付きである場合、サブ回路Aが入力/出カイベントポートもたない場合でも、サブ回路Aはイベント付きサブ回路として考えなければなりません。

PSIMIには、3種類のサブ回路があります。

- **標準サブ回路**：標準サブ回路はイベントポートを含んでおらず、従来のサブ回路と同じです。
- **イベント付きサブ回路**：イベント付きサブ回路には入出カイベントポートがあり、ハードウェア割り込み素子は接続されていません。
- **ハードウェア割り込み付きサブ回路**：ハードウェア割り込み付きサブ回路は、出力

イベントポートを含まず入力イベントポートだけを含んでおり、入力イベントポートにはハードウェア割り込み素子が接続されます。このタイプのサブ回路はハードウェア割り込みを扱うためだけに使用されます。

イベント付きサブ回路およびハードウェア割り込み付きサブ回路にはコード生成が可能となるように次のような制限があります。

- イベント付きサブ回路およびハードウェア割り込み付きサブ回路は、コード生成に使用可能な素子のみを含めることができます。つまり、このタイプのサブ回路はコード生成に使用できない抵抗やrmsブロックを含むことができません。
- ハードウェア割り込み付きサブ回路は、複数の入力イベントポートを持つことができますが出力イベントポートを持つことができません。更に、入力イベントポートにはハードウェア割り込み素子のみ接続可能です。入出力信号ポートにはハードウェア素子のみ接続することができ、他の機能ブロックに接続することができません。図4.4にハードウェア割り込みのあるサブ回路の接続方法を示します。

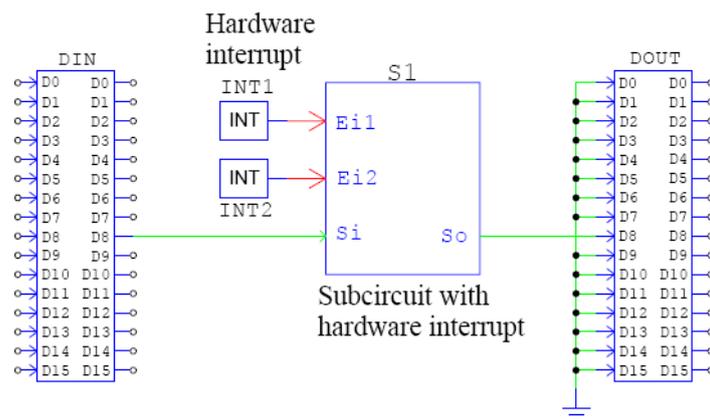


図4.4 ハードウェア割り込み付きサブ回路

ここで、S1はハードウェア割り込み付きサブ回路です。このサブ回路には2つハードウェア割り込み素子INT1とINT2が接続されています。この回路の入力信号ポートSiはハードウェアデジタル入力として出力信号ポートSoはハードウェアデジタル出力に接続されています。

- ハードウェア割り込み付きサブ回路がサンプリングレートの定義された離散系のブロックを含んでいると、そのサンプリングレートは無視されハードウェア割り込みが発生したときだけサブ回路が呼ばれます。例えば、サブ回路が離散積分器を含んでいると、離散積分器のサンプリングレートは無視されます。離散積分器の計算における前の時間は前回の割り込みが発生した時間になります。
- 2つのサブ回路の出力信号を接続する場合、直接接続しなければなりません。他のブロックの経路で接続することができません。これについては以下の図で説明します。

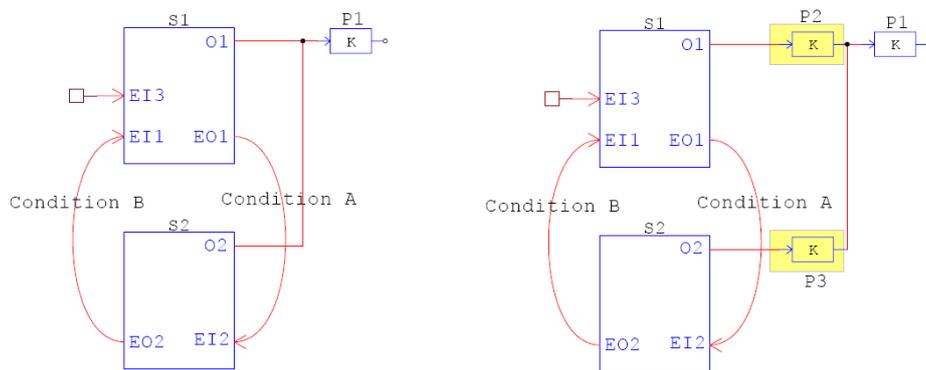


図4.5 2つのサブ回路の接続方法

左側の回路では、サブ回路S1とS2の両方がそれぞれ出力信号ポートO1およびO2を持っています。サブ回路S1およびS2の出力信号ポートはともに比例制御器ブロックP1の入力に接続されています。この接続で、P1ブロックの入力はポートO1またはO2のどちらかから信号が来ることになります。この接続は許可されます。しかし、右側の回路では、ポートO1はブロックP2に接続され、ポートO2はブロックP3に接続され、そして、次にP2とP3の出力がP1の入力に接続されています。このような接続は許可されません。この場合、ブロックP2はサブ回路S1の中に、そして、ブロックP3はサブ回路S2の中に移動する必要があります。

SimCoderは、MywayプラスのPE-Expert3ハードウェア開発プラットフォームをサポートしています。更に、SimCoderを利用して、汎用ハードウェアプラットフォームのためにCコードを生成することができます。ユーザは、生成された汎用的なCコードを修正してそれぞれのハードウェア（DSP・ $\mu$ C）で使用することができます。

### 5.1 標準 PSIM ライブラリの素子

PSIM標準ライブラリの素子の中では、以下の素子がコード生成に使用できます。

#### Elements→Control メニュー:

Proportional block（比例制御器）

Comparator（比較器）

Limiter（リミッタ）

Upper Limiter（上限）

Lower Limiter（下限）

Range Limiter（レンジリミッタ）

Summer (+/-)（加算器）

Summer (+/)（加算器）

Summer (3-input)（加算器-3入力）

#### Elements→Control→Computational Blocks メニュー:

Multiplier（乗算器）

Divider（除算器）

Square-root（平方根ブロック）

Sine

Sine (in rad.)

Arcsine

Cosine

Cosine (in rad.)

Arccosine

Tangent

Arctangent

Exponential ( $a^x$ )（指数ブロック）

Power ( $x^a$ ) (累乗)  
LOG (base e) (対数ブロック)  
LOG10 (base 10) (対数ブロック)  
Absolute Value (絶対値ブロック)  
Sign Block (符号関数ブロック)

**Elements→Control→Other Function Blocks メニュー:**

Multiplexer (2-input) (マルチプレクサ2入力)  
Multiplexer (4-input) (マルチプレクサ4入力)  
Multiplexer (8-input) (マルチプレクサ8入力)

**Elements→Control→Logic Elements メニュー:**

AND Gate (ANDゲート)  
AND Gate (3-input) (ANDゲート-3入力)  
OR Gate (ORゲート)  
OR Gate (3-input) (ORゲート-3入力)  
XOR Gate (XORゲート)  
NOT Gate (NOTゲート)  
NAND Gate (NANDゲート)  
NOR Gate (NORゲート)

**Elements→Control→Digital Control Module メニュー:**

Zero-Order Hold (ゼロ次ホールド)  
Unit Delay (単位時間遅れ)  
Integrator (積分器)  
Differentiator (微分器)  
External Resettable Integrator (外部リセット付き積分器)  
Internal Resettable Integrator (内部リセット付き積分器)  
FIR Filter (FIRフィルタ)  
FIR Filter (file) (FIRフィルタ(1))  
Digital Filter (デジタルフィルタ)  
Digital Filter (file) (デジタルフィルタ(1))  
z-domain Transfer Function (z 領域伝達関数)

**Elements→Other メニュー:**

Parameter File (パラメータファイル)

## Elements→Other→Function Blocks メニュー:

abc-dqo Transformation (abc-dqo変換)  
dqo-abc Transformation (dqo-abc変換)  
abc-alpha/beta Transformation (ABC- $\alpha/\beta$ 変換)  
alpha/beta-abc Transformation ( $\alpha/\beta$ -ABC変換)  
ab-alpha/beta Transformation (AB- $\alpha/\beta$ 変換)  
ac-alpha/beta Transformation (AC- $\alpha/\beta$ 変換)  
alpha/beta-dq Transformation ( $\alpha/\beta$ -dq変換)  
dq-alpha/beta Transformation (dq- $\alpha/\beta$ 変換)  
x/y-r/angle Transformation (x/y-r/angle変換)  
r/angle-x/y Transformation (r/angle-x/y変換)  
Lookup Table (早見表)  
2-D Lookup Table (integer) (早見表)  
2-D Lookup Table (interpolation) (早見表)  
Math Function (数式関数)  
Math Function (2-input) (数式関数-2入力)  
Math Function (3-input) (数式関数-3入力)  
Math Function (5-input) (数式関数-5入力)  
Math Function (10-input) (数式関数-10入力)  
Simplified C Block (単純化Cブロック)

## Elements→Sources メニュー:

Constant (定数)  
Ground (グラウンド)  
Ground (1) (グラウンド)  
Ground (2) (グラウンド)

## Elements→Sources→Voltage メニュー:

DC (直流源)  
DC (battery) (バッテリー)  
Sawtooth (のこぎり波)  
Grounded DC (接地電圧源)  
Grounded DC (1) (接地電圧源)  
Math Function (数式関数)

## 5.1.1 パラメータファイルブロックを用いたグローバルパラメータ設定

パラメータファイルブロックはSimCoder内でも同様に使用することができます。また、SimCoder内ではグローバルパラメータとして使用することもできます。

コードの可読性や修正の簡易性を高めるためにコード内に直接の値を書き込むのではなく、定数名を用いることがあります。例えば、制御器のゲインを1.23とした場合、コード内に1.23と直接書き込むのではなく、 $K_p=1.23$ と一度定義したのち、コード内では $K_p$ を用います。

この場合、パラメータはコード内の複数の場所で用いられ、グローバルな定数になります。パラメータをグローバルパラメータとして使いたい場合は、パラメータファイルブロックの中で、パラメータ名の前に「(global)」という文字を入れてください。

下記にその例を示しています。比例制御器のゲインの値をパラメータ $K_p$ として、パラメータファイル内では $K_p$ を「(global) $K_p=0.4$ 」と記述しています。

下記に生成されるコードを示しています。パラメータ $K_p$ はコードの最初に0.4と定義され、以降のコード内では $K_p$ が使われています。

## 5.1.2 のこぎり波生成

のこぎり波はDSP内で、システムのタイマーとしてや、また他の周期波形（正弦波など）を生成するために使われます。のこぎり波電圧源（**Elements** → **Sources** → **Voltage**の中より）はDSPのカウンタを使って実現されます。

例えば、PE-Expert3では、PEVボードの32ビットフリーランカウンタ（20ns毎カウント）を使ってのこぎり波を生成します。

汎用ハードウェアでは、対象とするDSP内に32ビットフリーランカウンタ（20ns毎カウント）が存在するとしてのこぎり波を生成します。

## 5.2 イベントコントロール素子

イベントコントロールを実現するために使用される素子を以下に説明します。

### 5.2.1 入力イベント

入力イベント素子のシンボルを以下に示します。



シンボルの右上隅の文字「i」は「入力」を表しています。入力イベント素子はサブ回路インターフェースポートの1つであり、サブ回路の中でのみ使用可能です。この素子をダブルク

クリックすると以下に示すようにポート名と位置を定義できます。

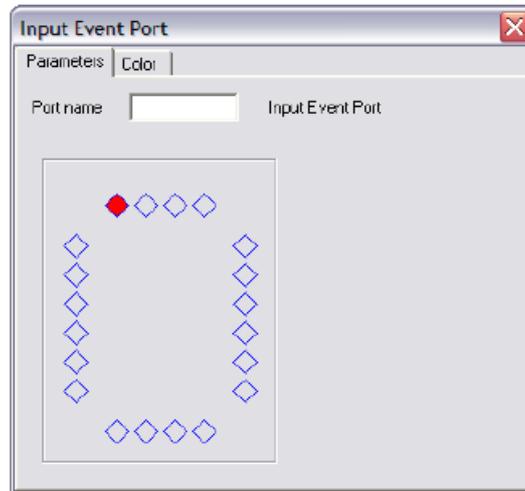


図5.1 入力イベントポート

入力イベントポートがあるサブ回路を呼び出すメイン回路において、この入力イベントポートにイベント接続ワイヤを接続すると、イベント接続の条件が成立した場合にこのポートを通してこのサブ回路へシステムの状態が移行します。

## 5.2.2 出力イベント

出力イベント素子のシンボルを以下に示します。



シンボルの右上隅の文字「o」は「出力」を表しています。出力イベント素子はサブ回路インターフェースポートの1つであり、サブ回路の中でのみ使用可能です。この素子をダブルクリックすると以下に示すようにポート名と位置と移行条件を定義できます。

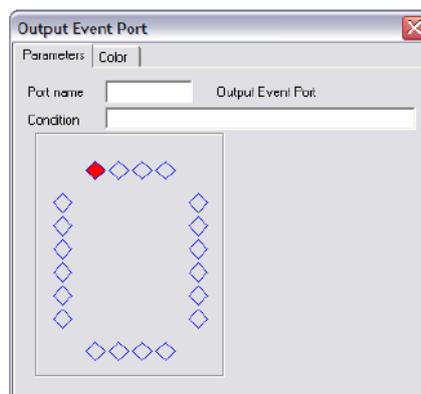


図5.2 出力イベントポート

出カイベントポートで定義された移行条件が成立すると、システムはこのサブ回路から他のサブ回路へ移行します。条件式は以下の例のようにCコードで表記します。

```
A == 1
A >= B
(A > B) && (C > D)
```

ここで、A、B、C、Dはグローバル変数か定数です。

## 5.2.3 デフォルトイベント

デフォルトイベント素子のシンボルを以下に示します。



デフォルトイベント素子は、どのサブ回路がデフォルト状態であるかを定義するのに使用され、サブ回路の入カイベントポートに接続されます。

## 5.2.4 イベント接続

イベント接続ワイヤは入カイベントポートと出カイベントポートとを接続するツールです。イベント接続ワイヤはイベントを接続するときのみ使用することができます。イベント接続ワイヤをダブルクリックして、出カイベントポートの条件式を修正することができます。

イベント接続ワイヤでは、開始点と終点以外にも2つのポイントがあり、これらの2つのポイントを移動することによって接続ワイヤの形を変えることができます。これらの2つのポイントを変更するには、イベント接続ワイヤをハイライトし、右クリックして、「Modify handle 1」あるいは「Modify handle 2」を選択してください。

## 5.2.5 イベントブロック初回実行フラグ

プログラムの組み方によっては、イベント用のサブ回路に入った初回だけ区別して命令を実行することがあります。イベントブロック初回実行フラグ（Flag for Event Block First Entry）を使うと、イベント用サブ回路に入ったのが初めてなのか、そうでないのかを区別することができます。

シンボル:



仕様:

パラメータ	機能
Event Subcircuit Block Name	グローバル変数名

パラメータの「Event Subcircuit Block Name」にサブ回路名を設定します。（イベントサブ

回路ブロックS1を初回実行したかどうかの区別をしたい場合は、パラメータ「Event Subcircuit Block Name」をS1にしてください。）サブ回路内を初回実行している間、本ブロックの出力ノードの値が1になります。その他の場合は0になります。

## 5.3 グローバル変数

パラメータ	機能
Name	グローバル変数名
Initial value	グローバル変数の初期値

信号をグローバル変数として定義するには、グローバル変数素子をノードに接続してください。コード生成のための制御回路の信号のみグローバル変数として定義することができます。

グローバル変数はグローバルにアクセスすることができ、サブ回路を含む回路内のすべてのグローバル変数の初期値を変化させることができます。

グローバル変数は、信号シンクか信号ソースとして使用することができます。信号シンクとして使用するときはノードから信号値を読みこみ、信号ソースとして使用するときはノードの値を設定します。

条件式の変数はグローバル変数として定義しなければなりません。イベント条件式でグローバル変数を使用する例を以下に示します。

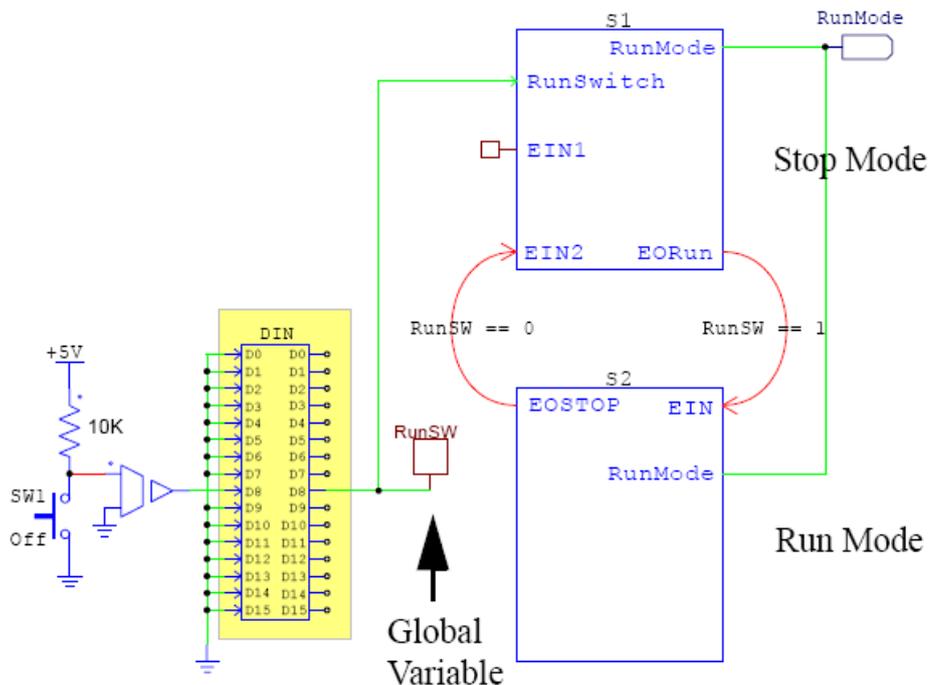


図5.3 グローバル変数の定義方法

この例では、グローバル変数 *RunSW* はデジタル入力素子の出力ピンD8に接続されています。このグローバル変数はStopモードとRunモードとの移行条件に使用されます。

グローバル変数は信号ソースとしても使用することができ、別のブロックに値を渡すことができます。

グローバル変数はあるノードから別のノードに値を渡すためのラベルとしては使用できません。

グローバル変数の使用には以下の制限があります：

- 同じ信号伝達経路でのみ同じ名前のグローバル変数を複数使用できます。
- 同じ信号伝達経路ではない場合、同じ名前のグローバル変数は同時に動作しないサブ回路内でのみ複数使用できます。

グローバル変数の接続方法を図5.4に説明します。

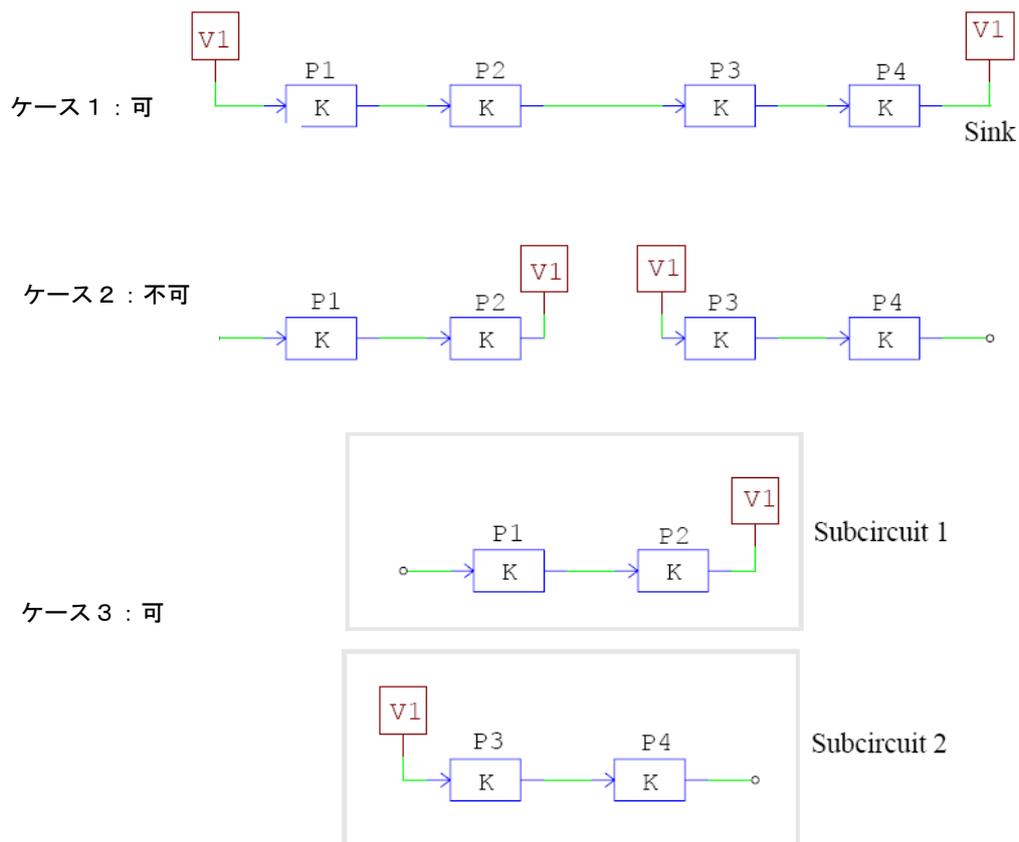


図5.4 グローバル変数の接続方法

ケース1では、グローバル変数V1は、ソースとして最初に使用されブロックP1の入力に値を割り当

てます。一連の計算の後、ブロックP4の出力は同じグローバル変数V1に割り当てられます。両方のグローバル変数が同じ信号伝達経路にあるので、この使用方は正しいです。

ケース2では、ブロックP2の出力からの値をブロックP3の入力へ渡すために、グローバル変数V1はラベルとして使用していますが、これは許可されずエラーとなります。値をあるノードから別のノードへ渡すためには、ラベル(  )を使用するかワイヤでノードを接続してください。

ケース3では、グローバル変数V1はサブ回路1およびサブ回路2で使用されます。システムでは、サブ回路1とサブ回路2は同時に実行しないので、このようなグローバル変数の使用は可能です。

## 5.4 ハードウェア割り込み

DSPIは、デジタル入力、エンコーダ、キャプチャ、PWM生成器などからハードウェア割り込みを発生させることができます。ハードウェア割り込みブロックは素子から発生した割り込みと対応させたい割り込み関数であるサブ回路を繋げる役割を持っています。

ハードウェア割り込み要素はサブ回路に置くことはできず、トップレベルのメイン回路にしか置くことはできません。

シンボル:

Interrupt



仕様:

パラメータ	機能
Device Name	割り込みを発生させる素子名の設定。
Channel No	割り込みを発生させる素子のチャンネル番号の設定。
Edge Detection Type	割り込み信号の発生エッジタイミングの設定。 デジタル入力およびキャプチャ割り込みのときに有効。 <ul style="list-style-type: none"> <li>- No edge detection : 割り込みは発生しません。</li> <li>- Rising edge : 入力信号の立ち上がりで割り込みが発生します。</li> <li>- Falling edge : 入力信号の立ち下がりで割り込みが発生します。</li> <li>- Rising/falling edges: 入力信号の立ち上がりおよび立ち下がりの両方で割り込みが発生します。。</li> </ul>

割り込み素子の使用方法を以下の図5.7に示します。

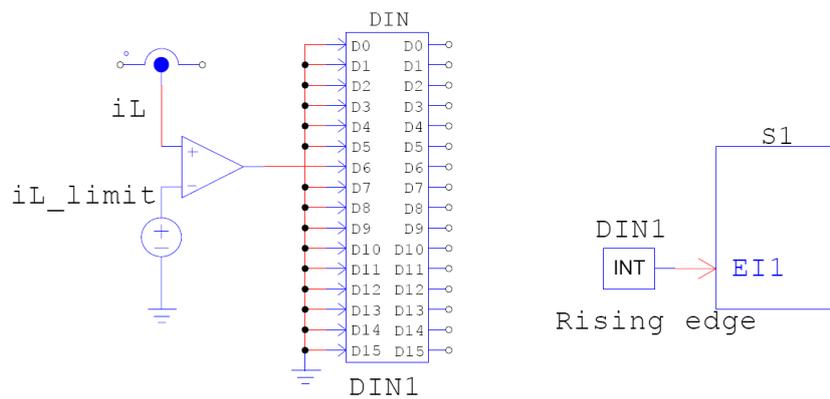


図5.7 割り込み素子の使用方法

この回路では、検出電流 $iL$ とリミット値 $iL\_limit$ を比較して、検出電流 $iL$ がこのリミット値を超えると、パルス信号が発生してそれをデジタル入力素子DIN1の入力に送ります。この回路では、割り込み素子の「Device Name」は「DIN1」として、「Edge Detection Type」は「Rising edge」として定義します。これにより、パルス信号が割り込みを発生させ、サブ回路S1のイベント入力ポートEI1から割り込み関数S1へ移行します。

## 6 TI F28335 Hardware Target

---

オプションモジュールTI F28335 Hardware TargetとSimCoderの組み合わせによって、PSIMの制御回路図からTexas Instruments社製浮動小数点型DSP TMS320F28335を搭載した基板で汎用的に使用できるコードを生成することができます。

TI F28335ハードウェアターゲットはF28335全てのパッケージに対応しています。下図はLQFPのTMS320F28335DSPのピンアサインです。TI F28335 Hardware Targetが対応している主な関数は図内で色を付けています。

TI F28335 Hardware Targetには下記の機能があります。

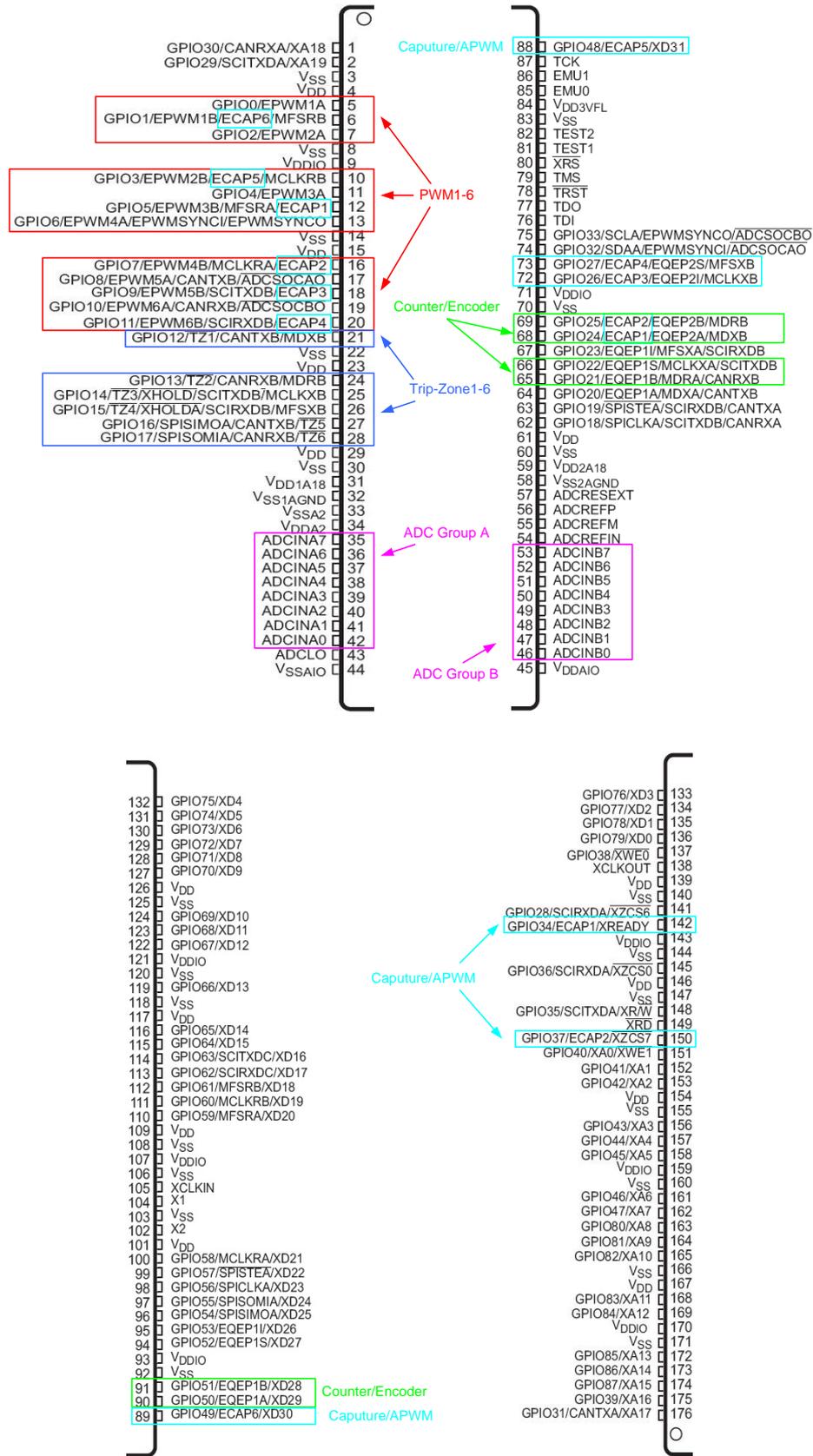
- 3-phase、2-phase、1-phase、Single PWM生成機能
- PWM生成のStart、Stop機能
- トリップゾーン、トリップゾーンステート機能
- A/D変換機能
- デジタル入力機能
- デジタル出力機能
- Up/Downカウンタ機能
- エンコーダ機能
- キャプチャ機能
- DSP クロック機能
- ハードウェア基板設定機能

複数のサンプリング周波数でコードを生成する回路の場合、SimCoderは優先的にPWM割り込みの周波数に使用します。他のサンプリング周波数では、タイマ1割り込み、タイマ2割り込みの順に使われます。3つ以上のサンプリング周期がある場合は対応する割り込み関数をメイン関数内で実行する形にします。

TI社製TMS320F28335ではPWM生成部からハードウェア割り込みを発生させることができますので、PWM生成ブロックに繋がっていて、PWM生成ブロックを同じ周波数を持つ全ての素子を検索してグループ化します。各素子ブロックは自動的に配置され、出力コード内の割り込み関数として実行されます。

また、ハードウェア割り込みはデジタル入力、エンコーダ、キャプチャ、そしてトリップゾーンにもあります。各ハードウェア割り込みは割り込みブロックから生成され、(5.4章参照)割り込みブロックは割り込み関数と繋げる必要があります。(割り込み関数はサブ回路で作ります。)二つ以上の割り込みを使うときはそれぞれの機能に割り込みブロックと割り込み関数を設定します。

本章ではTI F28335 Hardware Targetライブラリ内の各素子の使い方について解説します。



## 6.1 PWM 生成器

TMS320F28335 には、PWM1(GPIO0、GPIO1)、PWM2(GPIO2、GPIO3)、PWM3(GPIO4、GPIO5)、PWM4(GPIO6、GPIO7)、PWM5(GPIO8、GPIO9)、PWM6(GPIO10、GPIO11)のように合計で 6 組の PWM 出力があります。各組は互いに相補的になっており、特別な設定の場合を除いて、PWM1A が正の出力ならば PWM1B は逆の負の出力になります。

SimCoder では 6 つの PWM を下記のように使うことができます。

1. 3-phase PWM 生成器：2 種類あります。PWM1、2、3 と PWM4、5、6 という組み合わせです。
2. 2-phase PWM 生成器：6 種類あります。PWM1~6 を相補的な動作ではなく、設定によって幾つかの特別なモードで動作させることができます。
3. 1-phase PWM 生成器：PWM1~6 を相補的に動作させます。

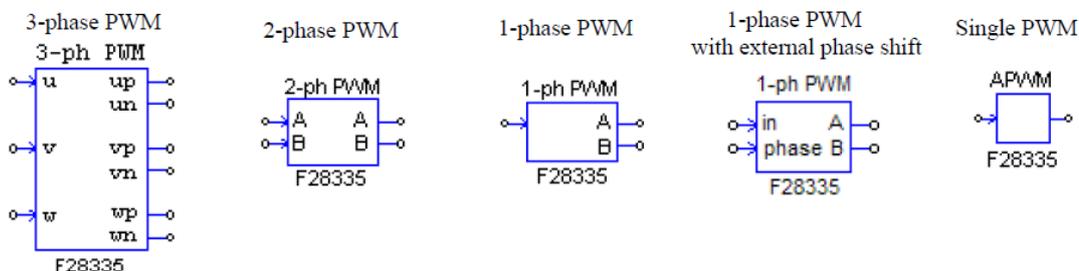
これらの PWM 生成器は A/D 変換のトリガを掛けることや、トリップゾーン信号を用いることもできます。

上記で説明した PWM 生成器の他にもう一点、6 つの single PWM 生成器があります。これは機能的には他のブロックに比べて制限されているものとなっており、A/D 変換トリガやトリップゾーンが使えず、更にピンがキャプチャと共用になっているため、キャプチャ機能と同時に使うことができません。

SimCoder の PWM 生成器は内部にスイッチング一回分の遅れを持っており、PWM の出力は入力が入ってから 1 サイクル分遅れて出力されます。これは実際の DSP の持つ遅れをシミュレーションに反映するためです。

PWM 生成器のシンボルとパラメータを以下に示します。

### シンボル:



三相 PWM 生成器の図中では、u、v、w は三相を示しています。(a、b、c 相とも呼ばれます。) p は正の出力を示し、n は負の出力を示しています。例えば、三相 PWM123 では、「up」は PWM1A、「un」は PWM1B となります。

## 3-phase PWM 生成器 :

### 仕様:

パラメータ	機能
PWM Source	PWM生成器の出力ピン設定。「3-phase PWM 123」(PWM1~3)か「3-phase PWM 456」(PWM4~6)のどちらかを選択します。
Dead Time	PWM生成器のデッドタイム (Td) の設定。単位は[sec]
Sampling Frequency	PWM生成器のサンプリング周波数の設定。単位は[Hz] PWM信号のデューティはここで設定した周波数で更新されます。
PWM Freq. Scaling Factor	PWM周波数とサンプリング周波数の倍率の設定。 1、2、3の設定ができます。 PWM周波数(制御スイッチで生成されるPWM出力信号の周波数)がサンプリング周波数の何倍かを設定します。 例) サンプリング周波数が50kHzで本設定値が2の場合はPWM周波数が100kHzになります。100kHzのスイッチング2回につき1回値の更新を行います。
Carrier Wave Type	キャリア波形タイプの設定。 <ul style="list-style-type: none"> <li>- 「Triangular(start low)」: キャリアを三角波、PWM出力の初期値はLow。</li> <li>- 「Triangular(start high)」: キャリアを三角波、PWM出力の初期値はHIGH。</li> <li>- 「Sawtooth(start low)」: キャリアをのこぎり波、PWM出力の初期値はLow。</li> <li>- 「Sawtooth(start high)」: キャリアをのこぎり波、PWM出力の初期値はHIGH。</li> </ul>
Trigger ADC	A/D変換へのトリガの設定。 <ul style="list-style-type: none"> <li>- 「Do not trigger ADC」: A/D変換トリガ機能を無効にします。</li> <li>- 「Trigger ADC Group A」: A/D変換器のGroup Aへのトリガを有効にします。</li> <li>- 「Trigger ADC Group B」: A/D変換器のGroup Bへのトリガを有効にします。</li> <li>- 「Trigger ADC Group C」: A/D変換器のGroup Cへのトリガを有効にします。</li> <li>- 「Trigger ADC Group A&amp;B」: A/D変換器のGroup AとGroup Bの両方へのトリガを有効にします。</li> </ul>
ADC Trigger Position	A/D変換トリガを出力する位置の設定。値は0~1まで設定可能です。 (例) 0の場合、PWM周期の始まりでA/D変換トリガを掛けます。 0.5の場合、PWM周期の180°の位置でA/D変換トリガを掛けます。
Use Trip-Zone I	PWM生成器と繋がる6つの各トリップゾーンの設定。 <ul style="list-style-type: none"> <li>- 「Disable Trip-Zone i」: 対応するトリップゾーン信号を無効にします。</li> <li>- 「One shot」: One shotモードでトリップゾーン信号を使用します。トリップゾーン信号が検出されると、PWM出力は手動で開始させる必要があります。</li> <li>- 「Cycle by cycle」: cycle-by-cycleモードでトリップゾーン信号を使用します。そのときの周期内でトリップゾーン信号が有効になり、次の周期でPWMは自動的に再出力されるようになります。</li> </ul>
Trip Action	トリップゾーン信号に対するPWM生成器の動作の設定。 <ul style="list-style-type: none"> <li>- 「High Impedance」: PWM出力部はハイインピーダンスになります。</li> <li>- 「PWM A high &amp; PWM B low」: PWMの正の出力はハイレベル、負の出力はローレベルになります。</li> <li>- 「PWM A low &amp; PWM B high」: PWMの正の出力はローレベル、負の出力はハイレベルになります。</li> <li>- 「No action」: 何も動作を設定しません。</li> </ul>
Peak-to-Peak Value	キャリア波のピーク間電圧Vppの設定。
Offset Value	キャリア波のオフセット電圧Voffsetの設定。
Initial Input Value u, v, w	u、v、w三相入力カノードの初期値の設定。
Start PWM at Beginning	開始からPWM信号出力をするかどうかの設定。 <ul style="list-style-type: none"> <li>- 「Start」: プログラム開始時にPWM出力を許可します。</li> <li>- 「Do not start」: 「Start PWM」関数を実行するまでPWM出力をしません。</li> </ul>

## 1-phase PWM 生成器（外部位相シフト機能付きを含む）：

### 仕様：

パラメータ	機能
PWM Source	PWM生成器の出力ピン設定。位相シフトを使用しない場合は「PWM1」から「PWM6」まで、位相シフトを使用する場合は「PWM2」から「PWM6」を選択します。
Output Mode	PWM生成器の出力モードの設定。 <ul style="list-style-type: none"> <li>- 「Use PWM A&amp;B」：PWM出力のAとBを両方使います。また互いに相補的になります。</li> <li>- 「Use PWM A」：PWM出力Aのみ使用します。</li> <li>- 「Use PWM B」：PWM出力Bのみ使用します。</li> </ul>
Dead Time	PWM生成器のデッドタイム (Td) の設定。単位は[sec]
Sampling Frequency	PWM生成器のサンプリング周波数の設定。単位は[Hz] PWM信号のデューティはここで設定した周波数で更新されます。
PWM Freq. Scaling Factor	PWM周波数とサンプリング周波数の倍率の設定。 1、2、3の設定ができます。 PWM周波数（制御スイッチで生成されるPWM出力信号の周波数）がサンプリング周波数の何倍かを設定します。 例）サンプリング周波数が50kHzで本設定値が2の場合はPWM周波数が100kHzになります。100kHzのスイッチング2回につき1回値の更新を行います。
Carrier Wave Type	キャリア波形タイプとPWM出力信号の初期状態の設定。 <ul style="list-style-type: none"> <li>- 「Triangular(start low)」：キャリアを三角波、PWM出力の初期値はLow。</li> <li>- 「Triangular(start high)」：キャリアを三角波、PWM出力の初期値はHIGH。</li> <li>- 「Sawtooth(start low)」：キャリアをのこぎり波、PWM出力の初期値はLow。</li> <li>- 「Sawtooth(start high)」：キャリアをのこぎり波、PWM出力の初期値はHIGH。</li> </ul>
Trigger ADC	A/D変換へのトリガの設定。 <ul style="list-style-type: none"> <li>- 「Do not trigger ADC」：A/D変換トリガ機能を無効にします。</li> <li>- 「Trigger ADC Group A」：A/D変換器のGroup Aへのトリガを有効にします。</li> <li>- 「Trigger ADC Group B」：A/D変換器のGroup Bへのトリガを有効にします。</li> <li>- 「Trigger ADC Group C」：A/D変換器のGroup Cへのトリガを有効にします。</li> <li>- 「Trigger ADC Group A&amp;B」：A/D変換器のGroup AとGroup Bの両方へのトリガを有効にします。</li> </ul>
ADC Trigger Position	A/D変換トリガを出力する位置の設定。値は0~1まで設定可能です。 (例) 0の場合、PWM周期の始まりでA/D変換トリガを掛けます。 0.5の場合、PWM周期の180°の位置でA/D変換トリガを掛けます。
Use Trip-Zone I	PWM生成器と繋がる6つの各トリップゾーンの設定。 <ul style="list-style-type: none"> <li>- 「Disable Trip-Zone i」：対応するトリップゾーン信号を無効にします。</li> <li>- 「One shot」：One shotモードでトリップゾーン信号を使用します。トリップゾーン信号が検出されると、PWM出力は手動で開始させる必要があります。</li> <li>- 「Cycle by cycle」：cycle-by-cycleモードでトリップゾーン信号を使用します。そのときの周期内でトリップゾーン信号が有効になり、次の周期でPWMは自動的に再出力されるようになります。</li> </ul>
Trip Action	トリップゾーン信号に対するPWM生成器の動作の設定。 <ul style="list-style-type: none"> <li>- 「High Impedance」：PWM出力部はハイインピーダンスになります。</li> <li>- 「PWM A high &amp; PWM B low」：PWMの正の出力はハイレベル、負の出力はローレベルになります。</li> <li>- 「PWM A low &amp; PWM B high」：PWMの正の出力はローレベル、負の出力はハイレベルになります。</li> <li>- 「No action」：何も動作を設定しません。</li> </ul>
Peak-to-Peak Value	キャリア波のピーク間電圧Vppの設定。
Offset Value	キャリア波のオフセット電圧Voffsetの設定。
Phase Shift	PWM生成器の出力に対してシフトする位相の値[deg]

	(外部位相シフト入力なしの1-phase PWM生成器)
Initial Input Value	入力ノードの初期値の設定。
Start PWM at Beginning	開始からPWM信号出力をするかどうかの設定。 - 「Start」 : プログラム開始時にPWM出力を許可します。 - 「Do not start」 : 「Start PWM」関数を実行するまでPWM出力をしません。

外部入力位相シフト機能付き 1-phase PWM 生成器は 2 入力の素子となります。1 つは PWM 入力 ("in" ノード)、もう 1 つは位相シフト[deg]の入力 ("phase"ノード) です。

1-phase PWM 生成器は、別の PWM 信号に対して位相をシフトさせた PWM 信号を生成することができます。「PWM1、2、3」と「PWM1、4、5、6」の 2 つの組み合わせがあります。

基準の PWM と位相をシフトさせる PWM は、同じ組み合わせのものでなければなりません。

つまり、PWM1 を基準として、PWM2 と 3、または PWM4、5、6 が PWM1 に対して位相シフトさせることができます。または PWM2 を基準として、PWM3 は PWM2 に対して位相シフトさせることができます。同様に、PWM4 (または 5) を基準として、PWM5 (または 6) を PWM4 (または 5) に対して位相シフトさせることができます。

しかし、PWM4、5、6 の基準として PWM2 または 3 を使用することは許可されていません。

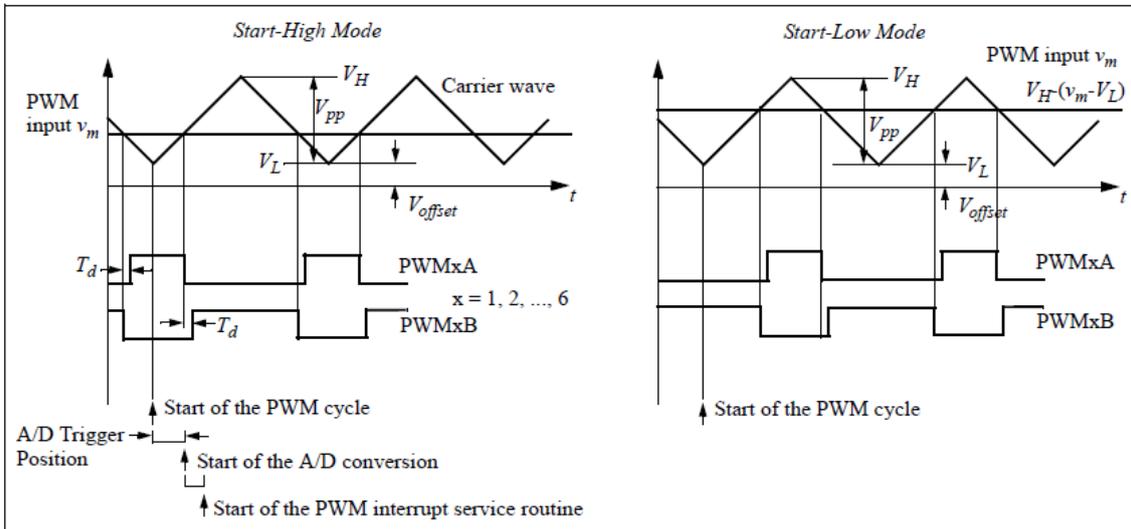
また、基準となる PWM とシフトされる PWM は、組み合わせ内で連続している必要があります。

つまり、PWM1 を基準として PWM3、または PWM5 または PWM6 を位相シフトすることはできません。

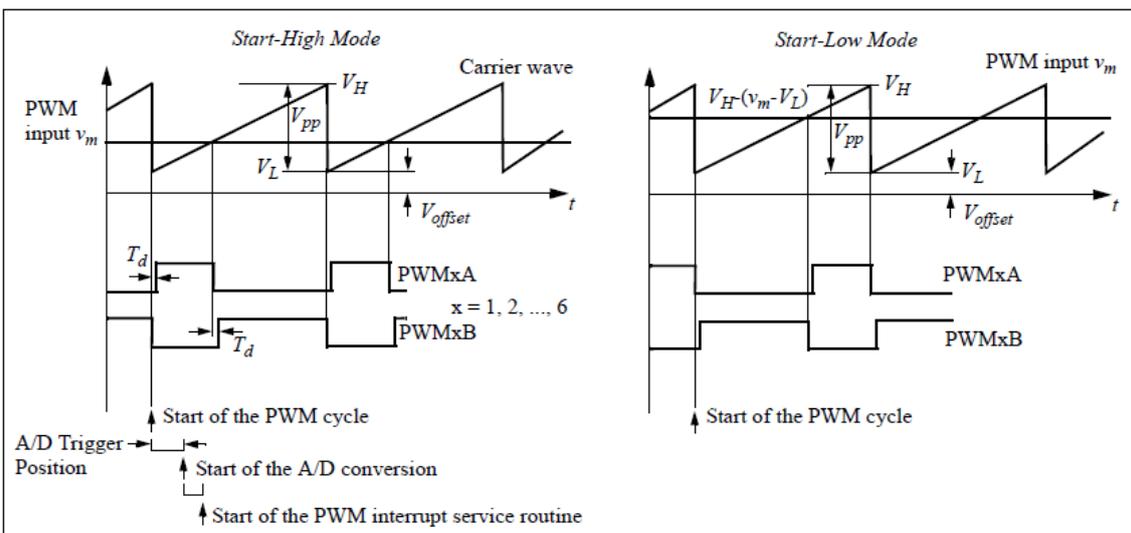
位相シフトの値は、度[deg]になります。値が $-30^{\circ}$  の場合、出力は基準の PWM 生成器の出力に対してスイッチングサイクルの  $30^{\circ}$  右に (遅れ) にシフトされます。これは、 $30^{\circ}$  右に PWM の搬送波をシフトすることと同じです。位相の値が  $30^{\circ}$  の場合、出力は  $30^{\circ}$  左 (進み) にシフトされません。

搬送波には 2 種類あります。三角波 (等しい傾きの間隔の立ち上がりと立ち下がりを持つ) とノコギリ波です。加えて、後述のような "Start-Low" と "Start-High" の 2 つの動作モードがあります。

三角搬送波の PWM 生成器の入力波形と出力波形は以下のとおりです。



ノコギリ搬送波の PWM 生成器の入力波形と出力波形は以下のとおりです。



上記の図は、デッドタイムの定義と PWM 生成器が A/D コンバータを動作させる際の時系列を示しています。PWM サイクルの開始時に A/D コンバータのトリガを設定した場合、A/D トリガポジションによって定義されるある一定の遅延の後、A/D 変換を開始します。A/D 変換が完了した後、PWM 割り込みルーチンが開始されます。

PWM 生成器が A/D コンバータを動作させない場合は、PWM サイクルの開始時に PWM 割り込みルーチンが開始されます。

上記の図は、「Start-High」と「Start-Low」モードの動作方法を示しています。PWM入力を「Vm」、搬送波の最小値を「VL」、最大値を「VH」とします。「Start-High」モードでは、PWMの正出力PWMAはスイッチングサイクルの開始時にHIGHとなり、入力「Vm」が搬送波よりも大きい間はHIGHを持続します。例えば、搬送波が0~1、つまりVL=0、VH=1の時、Vmが0.2の場合、PWM出力PWMAは搬送波が0.2より小さい間はHIGHを持続します。

一方、「Start-Low」モードでは、PWM正出力PWMAはスイッチングサイクルの開始時にLOWとなり、搬送波が「VH-(Vm-VL)」の値より大きい際にHIGHとなります。例えば、搬送波が0~1、つまりVL=0、VH=1の時、Vmが0.2の場合、PWM出力PWMAは搬送波が0.8より大きい間はHIGHとなります。

注：「Start-Low」モードでは、PWM信号を生成するために搬送波と比較する前にPWM入力「Vm」は「VH-(Vm-VL)」に変換されます。変換によって、「Start-Low」と「Start-High」の両方のモードで同じデューティ比の表現となります。例えば、VL=0、VH=1のノコギリ波、またはVL=-VHの三角波の場合、PWMA出力のデューティ比Dは「Start-Low」と「Start-High」の両方で「D=Vm/VH」となります。

## 2-phase PWM 生成器：

仕様：

パラメータ	機能
PWM Source	PWM生成器の出力ピン設定。 PWM1~6までを選択します。
Mode type	PWM生成器の動作モードの設定。 6つのモードから一つを選択します。各モードの動作の詳細は欄外の図を参照してください。
Sampling Frequency	PWM生成器のサンプリング周波数の設定。単位は[Hz] PWM信号のデューティはここで設定した周波数でアップデートされます。
PWM Freq. Scaling Factor	PWM周波数とサンプリング周波数のスケーリングファクタの設定。 1、2、3の設定ができます。 PWM周波数（制御スイッチで生成されるPWM出力信号の周波数）がサンプリング周波数の何倍かを設定します。 例）サンプリング周波数が50kHzで本設定値が2の場合はPWM周波数が100kHzになります。100kHzのスイッチング2回につき1回値のアップデートを行います。
Trigger ADC	A/D変換へのトリガの設定。 <ul style="list-style-type: none"> <li>- 「Do not trigger ADC」：A/D変換トリガ機能を無効にします。</li> <li>- 「Trigger ADC Group A」：A/D変換器のGroup Aへのトリガを有効にします。</li> <li>- 「Trigger ADC Group B」：A/D変換器のGroup Bへのトリガを有効にします。</li> <li>- 「Trigger ADC Group C」：A/D変換器のGroup Cへのトリガを有効にします。</li> <li>- 「Trigger ADC Group A&amp;B」：A/D変換器のGroup AとGroup Bの両方へのトリガを有効にします。</li> </ul>
ADC Trigger Position	A/D変換トリガをかける位置の設定。 値は0~1まで設定可能です。

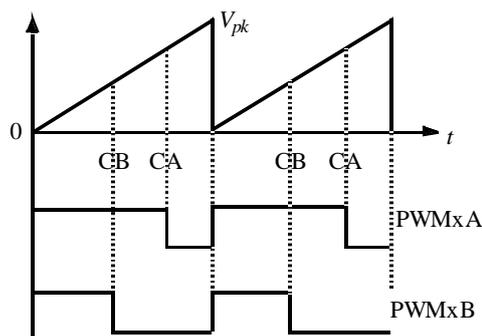
	(例) 0の場合、PWM周期の始めでA/D変換トリガを掛けます。 1の場合、PWM周期の最後でA/D変換トリガを掛けます。
Use Trip-Zone I	PWM生成器と繋がる6つの各トリップゾーンの設定。 2- 「Disable Trip-Zone I」 : 対応するトリップゾーン信号を無効にします。Z 2- 「One shot」 : One shotモードでトリップゾーン信号を使用します。トリガが掛かった後はPWMは手動で出力を開始するようになります。 - 「Cycle by cycle」 : cycle-by-cycleモードでトリップゾーン信号を使用します。そのときの周期内でトリップゾーン信号が有効になり、次の周期でPWMは自動的に再出力されるようになります。
Trip Action	トリップゾーン信号に対するPWM生成器の動作の設定。 - 「High Impedance」 : PWM出力部はハイインピーダンスになります。 - 「PWM A high & PWM B low」 : PWMの正の出力はハイレベル、負の出力はローレベルになります。 - 「PWM A low & PWM B high」 : PWMの正の出力はローレベル、負の出力はハイレベルになります。 - 「No action」 : 何も動作を設定しません。
Peak Value	キャリア波のピーク電圧 $V_{pk}$ の設定。
Initial Input Value A, B	A、B入力ノードの初期値の設定。
Start PWM at Beginning	開始からPWM信号出力をすかどうかの設定。 - 「Start」 : 開始からPWM出力を許可します。 - 「Do not start」 : 「Start PWM」関数を実行するまでPWM出力をしません。

2-phase PWM 生成器の出力形式は動作モード設定によって変わります。(以降の図参照) モードによってキャリアは三角波とのこぎり波のどちらかになります。また、キャリア値は0から $V_{pk}$ までの間の値を取り、DC オフセットの設定はできません。

### 動作モード1:

図のCAとCBは2-phase PWM 生成器の二つの入力A、Bになり、各出力のターンオフタイミングを決めます。

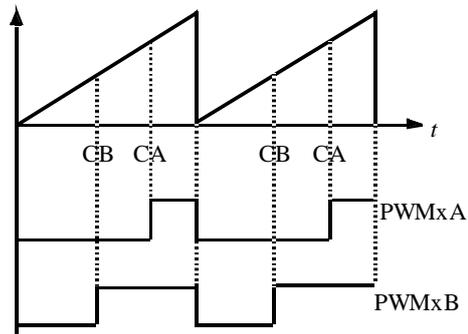
Mode 1:



### 動作モード2:

動作モード1と違い、書く出力のターンオンタイミングを決めるモードです。

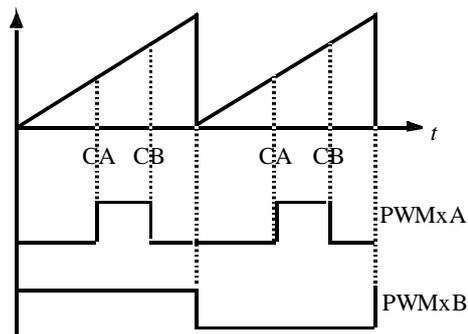
Mode 2:



### 動作モード 3 :

入力 A は PWM A 出力のターンオンのタイミングを決め、入力 B は PWM B 出力のターンオフのタイミングを決めます。PWM B 出力は PWM 周期の間オンになり、次の周期の間はオフになる、を繰り返します。

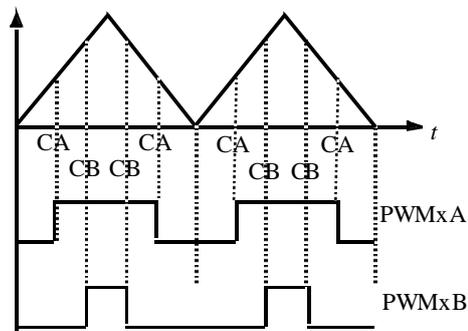
Mode 3:



### 動作モード 4 :

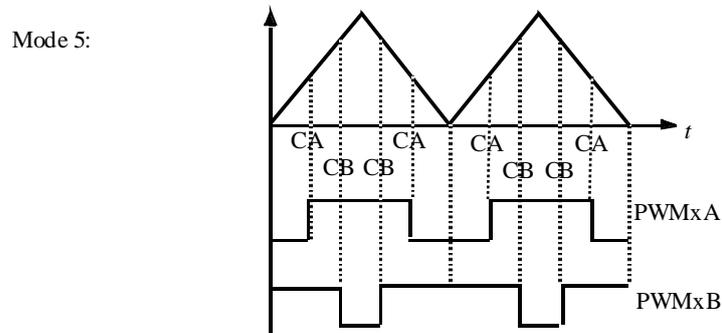
キャリアは三角波になります。各入力是对应する出力のターンオンとターンオフのタイミングを決めます。

Mode 4:



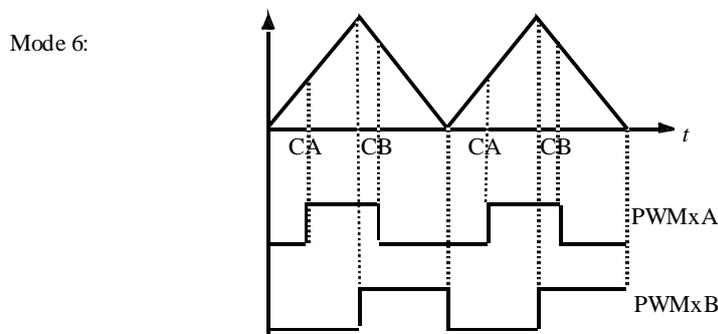
## 動作モード 5 :

動作モード 4 と違い、PWM A 出力と PWM B 出力のターンオフとターンオンのタイミングが反対に動作します。



## 動作モード 6 :

入力 A は PWMA のターンオンのタイミングを決め、入力 B は PWMA のターンオフのタイミングを決めます。PWM B は PWM 周期の最初の半周期でオンになり、次の半周期でオフになる動作をします。



## Single PWM 生成器(APWM) :

仕様:

パラメータ	機能
PWM Source	PWM生成器の出力ピン設定。 Single PWM生成器はキャプチャと同じピンを共有しており、下記のように各APWMは決められたピンから出力させることが可能です。 <ul style="list-style-type: none"> <li>- APWM1(GPIO5, GPIO24, GPIO34)</li> <li>- APWM2(GPIO7, GPIO25, GPIO37)</li> <li>- APWM3(GPIO9, GPIO26)</li> <li>- APWM4(GPIO11, GPIO27)</li> <li>- APWM5(GPIO3, GPIO48)</li> <li>- APWM6(GPIO1, GPIO49)</li> </ul>
PWM Frequency	PWM生成器の周波数の設定。単位は[Hz]

Peak-to-Peak Value	キャリア波のピーク間電圧の設定。
Offset Value	キャリア信号のオフセット電圧の設定。
Initial Input Value	入力の初期値の設定。
Start PWM at Beginning	開始からPWM信号出力をするかどうかの設定。 - 「Start」：開始からPWM出力を許可します。 - 「Do not start」：「Start PWM」関数を実行するまでPWM出力をしません。

Single PWM 生成器は機能的には他のブロックに比べて制限されており、A/D 変換トリガやトリップゾーン信号が利用できません。

## 6.2 Start PWM と Stop PWM

Start PWM と Stop PWM ブロックは、PWM 生成器の動作を開始する、または停止する機能があります。ブロック図とパラメータは下記に示します。

シンボル:



仕様:

パラメータ	機能
PWM Source	PWM生成器の出力の選択。 PWM1~6、3-phase PWM123と3-phase PWM456、そしてCapture 1~6が選択できます。

## 6.3 トリップゾーンとトリップゾーンステート

TMS320F28335 は 6 つのトリップゾーンを持っており、トリップゾーン 1~6 がそれぞれ GPIO12~GPIO17 に対応しています。トリップゾーンは外部からの不良や問題発生信号によって動作します。各 PWM 出力は対応した動作をするようにプログラムされます。

一つの PWM 生成器は 1 つから 6 つ全てまで一度に複数のトリップゾーンを使うことができます。割り込みブロックと共に使えばトリップゾーン信号は割り込み生成用に使えます。

入力信号がロー (0) になったとき、トリップゾーン信号は Trip 時動作を行います。

シンボル:



## 仕様（トリップゾーン）：

パラメータ	機能
Port GPIO12 (トリップゾーン1)	GPIO12ポートをトリップゾーン1に設定します。
Port GPIO13 (トリップゾーン2)	GPIO13ポートをトリップゾーン2に設定します。
Port GPIO14 (トリップゾーン3)	GPIO14ポートをトリップゾーン3に設定します。
Port GPIO15 (トリップゾーン4)	GPIO15ポートをトリップゾーン4に設定します。
Port GPIO16 (トリップゾーン5)	GPIO16ポートをトリップゾーン5に設定します。
Port GPIO17 (トリップゾーン6)	GPIO17ポートをトリップゾーン6に設定します。

## 仕様（トリップゾーンステート）：

パラメータ	機能
PWM Source	PWM生成器の出力の選択。 - PWM1~6、3-phase PWM123と3-phase PWM456、そしてCapture 1~6が選択できます。

トリップゾーン割り込みは PWM 生成器のパラメータとして One-Shot モードまたは Cycle-by-Cycle モードを設定したときに使うことができます。

Cycle-by-Cycle モードでは、割り込みはそのときの PWM 周期内のみで有効になります。また、One-Shot モードでは入力信号が(0)になったときのみ割り込みによる Trip 時動作が有効になります。割り込みにより出力が一時停止した後、PWM 生成器は再起動を開始するので原則として PWM 出力は続けられます。

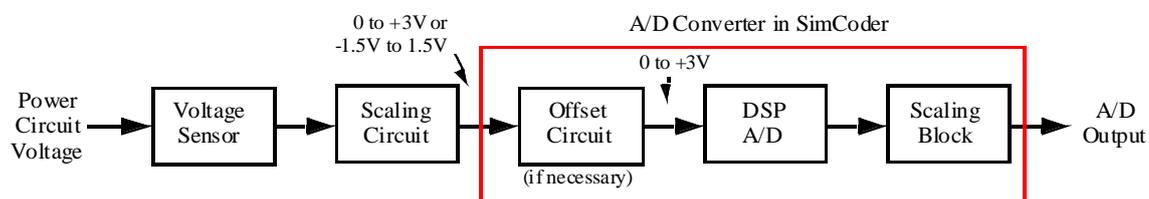
PWM 生成器から割り込みが発生させられたとき、トリップゾーン State ブロックはそれが One-Shot モード、Cycle-by-Cycle モードどちらの Trip 信号なのかを出力します。(1 のとき、One-Shot モード。0 のとき、Cycle-by-Cycle モード。)

トリップゾーンと接続して割り込みブロックを使う場合は、割り込みブロックの Device Name のパラメータはトリップゾーンブロック名ではなく、PWM 生成器の名前にしてください。例えば、PWM 生成器の名前が「PWM\_G1」、トリップゾーン1のブロック名が「TZ1」ならば、割り込みブロックの名前は「TZ1」ではなく「PWM\_G1」にしてください。(この場合は割り込みブロックの Channel Number のパラメータは使用しません。)

## 6.4 A/D 変換器

TMS320F28335 には 12 ビット 16 チャンネルの A/D 変換器があります。A/D 変換器は Group A と Group B の二つに分かれており、DSP への回路的な入力範囲は 0~+3V までです。

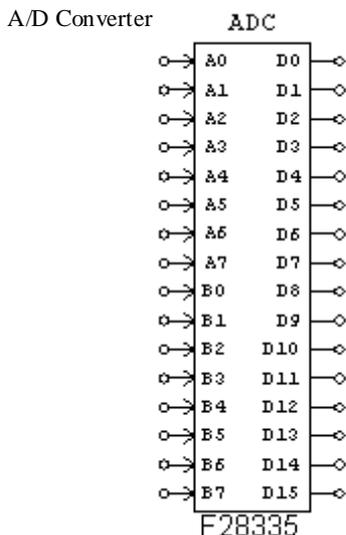
一般的に A/D 変換器から DSP へは、主回路の値（電圧、電流、速度など）が取り込まれます。主回路電圧を取り込む例を挙げると、ある程度の値になった主回路電圧は、最初に電圧センサで制御信号に変換されます。その後、回路信号を DSP 入力可能な範囲（0~+3V）に変換するため、スケーリング回路があり、必要によっては DC オフセット回路も使われます。DSP 内で信号はデジタル値へ変換され、スケーリングブロックで元の入力信号範囲に復元されます。（下図参照）



図より分かるように、SimCoder の A/D 変換器は、厳密には実際の DSP 搭載 A/D 変換器と等価ではなく、オフセット機能、A/D 変換機能、スケーリング機能の組み合わせの構成を持っています。

※以降は「A/D 変換器」という言葉を使うときは、DSP 搭載の A/D 変換器のことではなく、SimCoder の A/D 変換器ブロックのことを指すこととします。

シンボル:



## 仕様:

パラメータ	機能
ADC Mode	A/D変換器の動作モードを設定します。 <ul style="list-style-type: none"> <li>- Continuous : 連続変換を行います。変換値の読み値は、最後に変換が行われた値になります。</li> <li>- Start/Stop(8-Channel) : 片側のグループ (8チャンネル) のみを指令に従って一回だけ変換します。</li> <li>- Start/Stop(16-Channel) : 全ての入力チャンネルを指令に従って一回だけ変換します。</li> </ul>
Ch Ai or Bi Mode	各A/D変換チャンネルの入力モードの設定。 チャンネルにはAiとBiがあり、それぞれは0~7の値を取ります。 <ul style="list-style-type: none"> <li>- AC : -1.5Vから+1.5Vの入力が可能なac入力になります。</li> <li>- DC : 0Vから+3Vの入力が可能なdc入力になります。</li> </ul>
Ch Ai or Bi Gain	各A/D変換チャンネルのゲインの設定。 チャンネルにはAiとBiがあり、それぞれは0~7の値を取ります。

A/D 変換は Continuous モードに設定すると自律的に実行され、その他の設定の場合は PWM 生成器からのトリガで実行されます。

出力値は下記の数式で与えられます。

$$V_o = k \times V_i$$

$V_i$  は A/D 変換器の入力値です。

入力値は入力可能範囲内の値を設定してください。入力可能範囲外の値を入力した場合、出力は制限値でクランプされ、ワーニングメッセージを出力します。A/D 変換器の入力ポートの信号は、DC 入力モードの場合は+3V が最大値としてスケーリングされ、AC 入力モードの場合は 1.5V がピークとしてスケーリングされます。

以下に DC 入力設定時の構成例の図と AC 入力の場合の例を示しています。

### DC モードの例 :

A/D 変換器は DC モードに設定されており、主回路電圧値が DC で 0V から 150V までの値を取るとして、実際に入力値 100V となった場合の例を取り下記に説明を行います。

センサのゲインを 0.01 だとすると実際に入力されてくる値は下記になります。

$$V_{I_{max_s}} = 150 \times 0.01 = 1.5V$$

$$V_{I_s} = 100 \times 0.01 = 1V$$

最大値を見たとき、センサからは 1.5V が来ますが、DSP 入力可能最大値は 3V なのでゲイン 2 を追加します。(回路としては  $0.01 \times 2 = 0.02$  のゲインを持つことになります。) 調整後の入力値は下記のようにになります。

$$V_{I_{max_{s_c}}} = 1.5 \times 2 = 3V$$

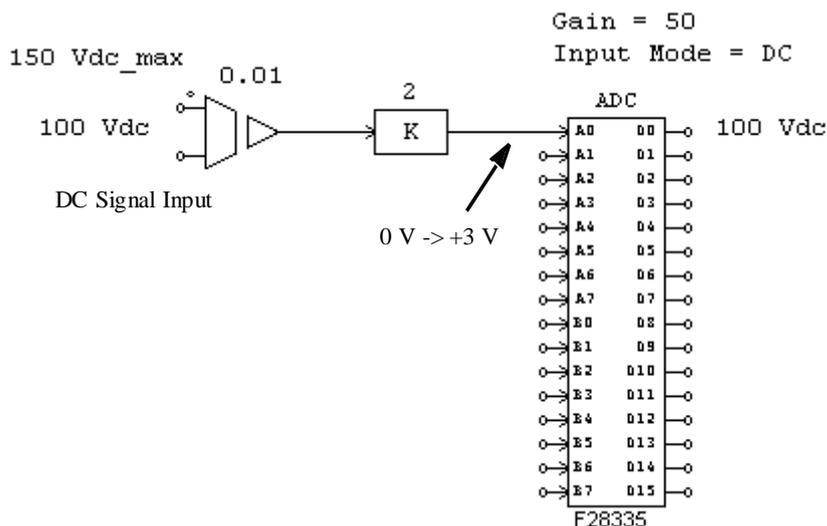
$$V_{I_{s_c}} = 1 \times 2 = 2V$$

A/D 変換後のスケールリングブロックの出力は、実際に主回路から来た電圧と合わせるとすると、A/D 変換器のゲインは 50 (電圧センサと調整用のゲインの逆数) になります。結果として A/D 変換器からの出力は下記になります。

$$V_{o_{max}} = 50 \times 3 = 150V$$

$$V_o = 50 \times 2 = 100V$$

回路図は以下になります。



※上図では比例制御器ブロックのゲインを 1、A/D 変換器のゲインを 100 にしてもシミュレーション結果は同じ結果になりますが、コード生成としては適切に行われぬ可能性がある点をご注意ください。コードは+3V が入力の最大値としてスケールリングしますが、この場合、1.5V が最大値になります。入力値の最大値が+3V になるようにスケールリングするようにしてください。

### AC モードの例 :

A/D 変換器は AC モードに設定されており、主回路電圧値が DC で  $\pm 75V$  までの値を取るとして、実際に入力値  $\pm 50V$  となった場合の例を取り下記に説明を行います。

センサのゲインを 0.01 だとすると実際に入力されてくる値は下記になります。

$$V_{i\_max\_s} = +/-0.75V$$

$$V_{i\_s} = +/-0.5V$$

ピーク値を見たとき、センサからは  $\pm 0.75V$  が来ますが、DSP 入力可能ピーク値は  $\pm 1.5V$  なのでゲイン 2 を追加します。調整後の入力値は下記のようにになります。

$$V_{i\_max\_s\_c} = +/-1.5V$$

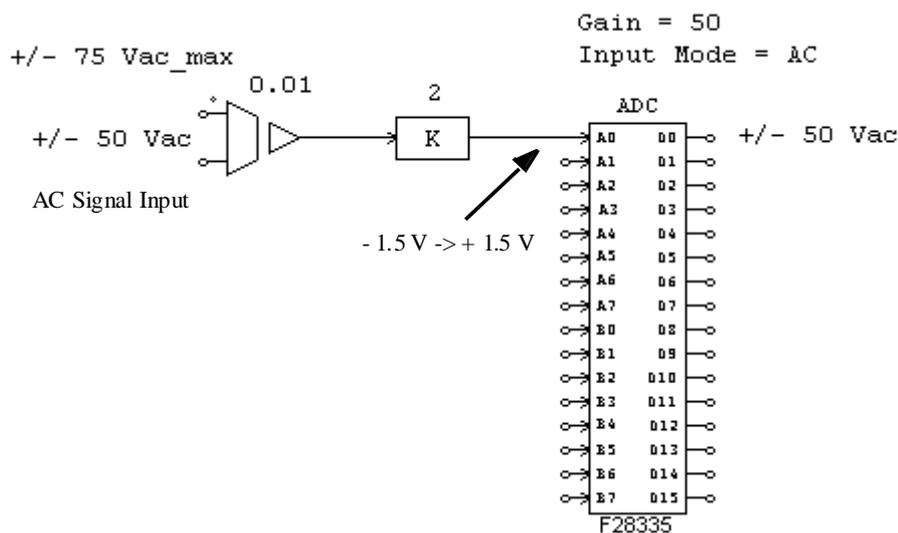
$$V_{i\_s\_c} = +/-1V$$

A/D 変換後のスケージングブロックの出力は、実際に主回路から来た電圧と合わせるとすると、A/D 変換器のゲインは 50 (電圧センサと調整用のゲインの逆数) になります。結果として A/D 変換器からの出力は下記になります。

$$V_{o\_max} = +/-75V$$

$$V_o = +/-50V$$

回路図は以下になります。



※PSIM の回路では直接 AC 信号を A/D 変換器に入力することができます。これは、ブロックの中にオフセット追加機能が含まれており、自動的に調整を行うためです。実際の DSP は  $0V \sim 3V$  までの入力となり、外部にオフセット回路を追加する必要がある点にご注意ください。

また、正確なコード生成を行うために A/D 変換器入力ポートのピーク値の最大値は  $1.5V$  になるようにしてください。

※また、下記の制約があることにもご注意ください。

A/D 変換器は一つの PWM 生成器トリガのみを受け付けます。もし複数の PWM 生成器がある場

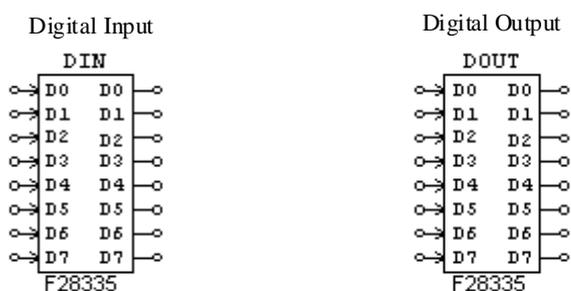
合は、一つだけの A/D 変換器へのトリガを有効にして他のブロックのトリガ機能は無効にしてください。

PWM 生成器から A/D 変換器へのトリガが無効の場合、変換グループへの入力信号は PWM 生成器の周波数ではなく、別の回路のサンプリング周波数を使います。このような場合は、A/D 変換器を Continuous モードに設定して使うことをお勧めします。

## 6.5 デジタル入力とデジタル出力

TMS320F28335 は 88 本の汎用入出力ポート（GPIO）をデジタル入力、またはデジタル出力に設定可能です。SimCoder では 8 チャンネルのデジタル入出力持つブロックを使うことができます。

シンボル:



仕様（デジタル入力）:

パラメータ	機能
Port Position for Input i	ブロックのポートをGPIOピンへの設定。 iはブロックの入力で0~7です。GPIO0~87まで選択できます。
Use as External Interrupt	外部割込み入力の設定。

仕様（デジタル出力）:

パラメータ	機能
Port Position for Output i	ブロックのポートをGPIOピンへの設定。 iはブロックの出力で0~7です。GPIO0~87まで選択できます。

※GPIO ポートを入力ポートとした場合、そのポートは他のペリフェラルのポートとしては使えない点にご注意ください。例えば GPIO1 はデジタル入力ポートと PWM1 出力の両方の設定ができませんが、同時に設定した場合、エラーが出力されます。

TMS320F28335 は最大で 7 つの外部割込み発生源を持っており、GPIO0~63 の中で使うことがで

きます。(GPIO0~31 にて 2 つまで外部割込みを利用でき、GPIO32~63 で 5 つまで利用できます。)

## 6.6 UP/DOWN カウンタ

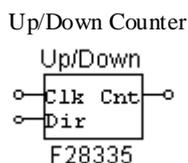
TMS320F28335 は UP/DOWN カウンタを二つ持っており、カウンタ 1 はポート GPIO20,21 または GPIO50,51 で利用でき、カウンタ 2 は GPIO24,25 で利用できます。

図内の Clk はクロック信号入力です。Dir は信号に対してカウンタをどちらへ進めるかの指令入力です。Dir が 1 のときカウンタは増加し、0 のときは減少します。

UP/DOWN カウンタブロックの出力はカウンタ値となります。

※GPIO20~21 と GPIO50~51 は同じ内部ブロックを使っているため、これらを同時に利用することはできません。

シンボル:



仕様:

パラメータ	機能
Counter Source	カウンタとポートの設定。 <ul style="list-style-type: none"> <li>- Counter1 (GPIO20, 21) : Counter1をGPIO20,21で使用します。</li> <li>- Counter1 (GPIO50, 51) : Counter1をGPIO50,51で使用します。</li> <li>- Counter2 (GPIO24, 25) : Counter2をGPIO24,25で使用します。</li> </ul>

※入力端子はポート番号の若い方から Clk 入力、Dir 入力の順に割り振られます。例えば Counter1 を GPIO20,21 に設定した場合、Clk 入力が GPIO20 に、Dir が GPIO21 に割り振られます。

※GPIO ポートを UP/DOWN カウンタに設定した場合、そのポートはエンコーダなどのポートとしては使えない点にご注意ください。例えば、UP/DOWN カウンタ 1 とエンコーダ 1 を同時に設定した場合、エラーが出力されます。

※ハードウェアボードコンフィギュレーションブロックと同時に使う場合は、所定の GPIO ポートの Encoder にチェックを入れてください。

## 6.7 エンコーダとエンコーダステート

TMS320F28335 はエンコーダ入力を二つ持っており、エンコーダ 1 はポート GPIO20,21 または

GPIO50,51 で利用でき、エンコーダ 2 は GPIO24,25 で利用できます。

エンコーダステートブロックはどの入力信号（基準信号とストローブ信号）が割り込みを発生させる設定なのかを見ることに使います。

エンコーダステートブロックの出力が 0 のときは、Z 信号（基準信号）から割り込みが発生する設定になっていることを示し、1 のときはストローブ信号から割り込みが発生する設定になっていることを示します。

※GPIO20~21 と GPIO50~51 は同じ内部ブロックを使っているため、これらを同時に利用することはできません。

## シンボル:



## 仕様（エンコーダ）:

パラメータ	機能
Counter Source	エンコーダとポートの設定。 <ul style="list-style-type: none"> <li>- Encoder1 (GPIO20, 21) : Encoder1をGPIO20,21で使用します。</li> <li>- Encoder1 (GPIO50, 51) : Encoder1をGPIO50,51で使用します。</li> <li>- Encoder2 (GPIO24, 25) : Encoder2をGPIO24,25で使用します。</li> </ul>
Use Z Signal	Z信号（基準信号）の有効無効設定。
Use Strobe Signal	ストローブ信号の有効無効設定。
Counting Direction	カウンタの方向の設定。 <ul style="list-style-type: none"> <li>- Forward : カウンタは増加方向に進みます。</li> <li>- Reverse : カウンタは減少方向に進みます。</li> </ul>
Encoder Resolution	接続するエンコーダの分解能の設定。 4096に設定した場合、カウンタ値が4095になった次のタイミングで0にクリアされます。0に設定した場合、クリアは行われません。

## 仕様（エンコーダステート）:

パラメータ	機能
Encoder Source	状態を観測するエンコーダ番号の設定。 <ul style="list-style-type: none"> <li>- Encoder1 (GPIO20, 21) : GPIO20,21のEncoder1に対して利用します。</li> <li>- Encoder1 (GPIO50, 51) : GPIO50,51のEncoder1に対して利用します。</li> <li>- Encoder2 (GPIO24, 25) : GPIO24,25のEncoder2に対して利用します。</li> </ul>

## 6.8 キャプチャとキャプチャステート

TMS320F28335 はキャプチャ入力を 6 つ持っており、割り込みを発生させることができます。割り込みブロックで割り込みトリガモードの設定ができます。

キャプチャステートブロックは 1 か 0 を出力し、1 のときは立ち上がりエッジを示し、0 のときは立下りエッジを示します。

シンボル:



仕様（キャプチャ）:

パラメータ	機能
Capture Source	キャプチャとポートの設定。 6つのキャプチャと14の設定可能なGPIOポートがあります。 <ul style="list-style-type: none"> <li>- Capture1 (GPIO5, GPIO 24, GPIO 34)</li> <li>- Capture2 (GPIO7, GPIO 25, GPIO 37)</li> <li>- Capture3 (GPIO9, GPIO 26)</li> <li>- Capture4 (GPIO11, GPIO 27)</li> <li>- Capture5 (GPIO3, GPIO 48)</li> <li>- Capture6 (GPIO1, GPIO 49)</li> </ul>
Event Filter	イベントフィルタのプリスケール設定。 入力信号は選択したプリスケールで分けられます。
Timer Mode	キャプチャカウンタタイマモード設定。 Absolute timeまたはTime differeneのどちらかを設定できます。

仕様（キャプチャステート）:

パラメータ	機能
Capture Source	状態を観測するキャプチャ番号の設定。 Capture1から6まで選択できます。

## 6.9 シリアル通信インターフェース（SCI）

F28335 DSP は、シリアルコミュニケーションインターフェース（SCI）のための関数が用意されています。SCI を介して、DSP 内部のデータは、外部 RS-232 ケーブルを使用してコンピュータに転送することができます。PSIM は、DSP とコンピュータ両方のデータの送受信およびコンピュータ上にデータを表示するための必要なすべての関数を提供します。これは DSP のコードをリアルタイムでモニタ、デバッグ、および調整するために非常に便利な方法を提供します。

SCI およびモニタ機能に関する詳細説明については、" Tutorial - Using SCI for Real-Time Monitoring in TI F28335 Target.pdf"を参照してください。

以下に説明するように、3つのSCIの機能ブロックがSimCoderで用意されています：

SCI設定、SCI入力、およびSCI出力

## 6.9.1 SCI設定

SCI設定ブロックは、SCIポート、通信速度、パリティチェックのタイプ、およびデータのバッファサイズを定義します。

シンボル：



仕様：

パラメータ	機能
SCI port	SCIポートの設定。SCIに使用できる7セットのGPIOポートがあります。 <ul style="list-style-type: none"><li>- SCIA (GPIO28, GPIO29)</li><li>- SCIA (GPIO35, GPIO36)</li><li>- SCIB (GPIO9, GPIO11)</li><li>- SCIB (GPIO14, GPIO15)</li><li>- SCIB (GPIO18, GPIO19)</li><li>- SCIB (GPIO22, GPIO23)</li><li>- SCIC (GPIO62, GPIO63)</li></ul>
Speed(bps)	SCI通信速度[bps]。プリセット速度は、200000、115200、57600、38400、19200、9600 [bps]が用意されています。または手動で他の速度を指定することができます。
Parity Check	通信エラーチェック用のパリティチェックの設定。 無効、奇数、偶数のいずれかを選択できます。
Output Buffer Size	SCI用にDSPに割り当てられたデータバッファのサイズ。バッファはRAM領域に位置しており、各バッファの要素は3つの16ビットワード（データポイントごとに、6バイト、または48ビットである）から成る1つのデータポイントを格納しています。

バッファサイズを正しく選択しなければならないことに注意してください。

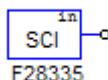
より多くの変数を長い期間にわたって監視することができるように、より多くのデータポイントを収集するためには大きなバッファが好ましい。一方、内部DSPのメモリは限られており、バッファは通常のDSPの動作を妨げるほど大きすぎるとはいけません。バッファサイズを選択する方法の詳細については、"Tutorial - Using SCI for Real-Time Monitoring in TI F28335 Target.pdf"を参照してください。

## 6.9.2 SCI入力

SCI入力ブロックは、ハングアップしうるDSPコードで変数を定義するために使用されます。SCI入力変数の名前は、DSPのオシロスコープ（ユーティリティのメニューの下）に表示され、その値はSCIを介して実行時に変更することができます。

SCI入力ブロックは、例えば、指令値の変更、コントローラパラメータの微調整のための便利な方法を提供します。

シンボル：



仕様:

パラメータ	機能
Initial Value	SCI入力変数の初期値。

回路図では、SCI 入力は定数として動作。コードが DSP 上で実行され、実行時にハングアップしうる間、値はシミュレーションの初期値に固定されます。

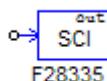
### 6.9.3 SCI 出力

SCI 出力ブロックは、表示用の変数を定義するために使用されます。

SCI 出力ブロックがノードに接続されている場合、SCI 出力ブロックの名前は、DSP のオシロスコープ（ユーティリティのメニューの下）に表示され、この変数のデータは、実行時に SCI を介してコンピュータに DSP から送信することができ、波形は DSP オシロスコープで表示することができます。

SCI の出力ブロックは、DSP 波形を観測するための便利な方法を提供します。

シンボル:



仕様:

パラメータ	機能
Data Point Step	データの収集頻度を定義します。[Data Point Step]が1の場合、すべてのデータポイントが収集され送信されます。例えば、[Data Point Step]が10の場合、10ポイントのうち1ポイントのみが収集され送信されます。

[Data Point Step]が小さすぎると、データポイントが非常に多くなる可能性があり、すべてを送信できない場合があることに注意してください。この場合、いくつかのデータポイントがデータの送信時に破棄されます。

また、[Data Point Step]のパラメータは DSP オシロスコープは連続モードの時のみ使用されます。スナップショットノードにある場合、このパラメータは無視され、すべてのポイントを収集して送信されます。シミュレーションでは、SCI 出力は電圧プローブとして動作します。

## 6.10 シリアルペリフェラルインターフェース(SPI)

F28335 DSP はシリアルペリフェラルインターフェース（SPI）のための関数が用意されています。TI F8335 ターゲットライブラリの SPI ブロックによって、容易かつ便利に外部の SPI デバイス（外付け A/D および D/A コンバータなど）と通信する機能を実装することができます。SPI デバイスのために手動でコードを書くことは、しばしば時間がかかり、大変な作業です。SPI をサポートする機能によって PSIM は大幅に簡素化し、コーディングとハードウェアの実装プロ

セスをスピードアップします。

SPI ブロックを使用する方法についての詳細な説明については、ドキュメントを参照してください" Tutorial - Using SCI for Real-Time Monitoring in TI F28335.pdf".

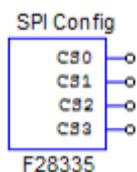
SimCoder では以下の 4 つの SCI の機能ブロックが提供されています：

SPI 設定、SPI デバイス、SPI 入力、SPI 出力

## 6.10.1 SPI 設定

SPI 設定ブロックは、SPI ポート、チップ選択ピン、および SPI のバッファサイズを定義します。SPI が使用されている回路図に必要で、ブロックはメイン回路図内に配置される必要があります。

シンボル:



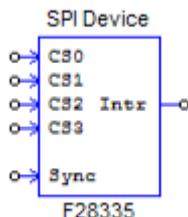
仕様:

パラメータ	機能
SPI port	SPIポートの設定。SPIポートはGPIO16-19またはGPIO54-57があります。
Chip Select Pin 0,1,2,3	チップセレクトピンのGPIOポート。PSIMは、チップセレクトピンPin0～Pin3で定義されている4つのGPIOピンを選択する必要がある16 SPIデバイスに対応しています。これらのGPIOポートとSPIスレーブ送信許可ピンSPISTEは、チップセレクト信号を生成するために使用されます。
SPI Buffer Size	SPIコマンドのバッファサイズ。バッファの各メモリセルは、SPIコマンドのインデックスが保存されます。通常は、すべてのSPIの入力/出力要素の中でSPIコマンド（例えば、Start Conversion Command, Receiving Data Command, Sending Data Command, and Sync. Command）の数に1を足したバッファサイズを指定します。

## 6.10.2 SPI デバイス

SPI デバイスのブロックは、対応する SPI ハードウェアデバイスの情報を定義します。回路図の SPI デバイスのブロック数は、SPI ハードウェアデバイスの数と同じでなければなりません。

シンボル:



仕様:

パラメータ	機能
Chip Select Pins	SPIデバイスに対応するチップセレクトピンの状態。チップセレクトピンがこの状態にあるときは、このSPIデバイスが選択されます。
Communication Speed (MHz)	SPIの通信速度 (MHz)
Clock Type	SPIハードウェアデバイスによって決定されるSPIクロックの種類。 <ul style="list-style-type: none"> <li>-立ち上がりエッジ (遅れなし) : クロックは通常はLOWで、データはクロックの立ち上がりエッジでラッチされます。</li> <li>-立ち上がりエッジ (遅れあり) : クロックは通常はLOWでデータは遅延付きのクロックの立ち上がりエッジでラッチされます。</li> <li>-立ち下がりエッジ (遅れなし) : クロックは通常はHIでデータはクロックの立ち下がりエッジでラッチされます。</li> <li>-立ち下がりエッジ (遅れあり) : クロックは通常はHIでデータは遅延付きのクロックの立ち下がりエッジでラッチされます。</li> </ul>
Command Word Length	SPI通信コマンドのワード長、または有効ビットの長さ。1~16ビットにすることができます。
Sync. Active Mode	SPIデバイスの同期信号のトリガモード。立ち上がりエッジまたは立ち下がりエッジのいずれかです。
SPI Initial Command	SPIデバイスの初期コマンドです。
Hardware Interrupt Mode	SPIデバイスが生成する割り込み信号の種類を指定します。SPIデバイスの割り込み出力ノードがデジタル出力要素の入力に接続されている場合にのみ有効です。次のいずれかになります。 <ul style="list-style-type: none"> <li>- ハードウェア割り込みなし</li> <li>- 立ち上がりエッジ</li> <li>- 立ち下がりエッジ</li> </ul>
Interrupt Timing	変換完了時のSPIデバイスの割り込み生成方法を指定します。次のいずれかになります。 <ul style="list-style-type: none"> <li>- 割り込みなし 割り込みは生成されません。この場合、DSPはSPI入力デバイスにコマンドを送信します。デバイスは変換を開始して同じコマンドで結果を返します</li> <li>- 連続多重割り込み : 多重割り込みは、各変換の後に連続で生成されます。 1つのA/D変換ユニットと複数の入力チャンネルを持つSPIデバイス用です。この場合、DSPが最初に変換のコマンドを送信し、SPIデバイスの変換が開始されます。変換が完了すると、SPIデバイスが割り込みを生成します。割り込みサービスルーチンでは、DSPが変換結果をフェッチするためにコマンドを送信し、同じSPI入力デバイスの別のチャンネルの新しい変換を開始します。</li> <li>- ファンタイム割り込み : 変換終了後に1回だけ割り込みが生成されます。1つのリクエストで複数のチャンネルの変換を実行可能なSPIデバイス用です。この場合、DSPはSPI入力デバイスにコマンドを送信し、SPIデバイスは複数の入力チャンネルの変換を完了します。すべての変換が完了すると、SPIデバイスが割り込みを生成します。</li> </ul>
Command Gaps (ns)	2つのSPIコマンドの間隔[nsec]です。
Conversion Sequence	変換シーケンスを決定するためのコマンドで区切られたSPIの入力要素の名前を定義します。このパラメータは、SPIデバイスが連続で複数の割り込みを生成する場合にのみ有効であることに注意してください。

回路図では、チップセレクトロジックが実装される方法を定義せずに、すべての SPI デバイスのチップセレクトピンは SPI 設定ブロックのチップセレクト端子に接続されています。

しかし、実際のハードウェアでは、対応するチップセレクトロジックに合わせて実装する必要があります。

SPI のコマンドは、カンマで区切られた 16 ビットの一連の数字で構成されています。 16 ビットの数値で

は、下位ビットだけがコマンドによって使用される有効ビットです。例えば、コマンドワード長が 8 の場合、ビット 0~7 はコマンドであり、ビット 8~15 は使用されません。

SPI デバイスは、入力デバイスまたは出力デバイスのいずれかです。

例えば、外部 A/D コンバータは、入力デバイスです。通常、DSP はデバイスに 1 つまたは複数の A/D 変換のコマンドを送信し、変換を開始するために同期信号を設定します。同期信号は、同じデバイスの次回のコマンドでリセットされます。

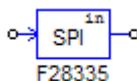
同期信号を使用した SPI の入力デバイスは、通常、割り込みサービスルーチンを入力して DSP をトリガーする割り込みピンを必要とします。

一方、外部 D/A コンバータは出力デバイスです。通常、DSP はデバイスに 1 つまたは複数の D/A 変換のコマンドを送信し、変換を開始するために同期信号を設定します。同期信号は、同じデバイスの次回のコマンドでリセットされます。

### 6.10.3 SPI 入力

SPI 入力デバイスは、複数の入力チャンネルを持つことができます。SPI 入力ブロックは、SPI 通信用の入力チャンネル、および 1 つの入力チャンネルに対応する 1 つの SPI 入力ブロックのプロパティを定義するために使用されます。

シンボル:



仕様:

パラメータ	機能
Device Name	SPI入力デバイスの名前です。
Start Conversion Command	コンマで区切られた16進の数字で、変換を開始するコマンドです。 (例 ; 0x23, 0x43, 0x00)
Receiving Data Command	コンマで区切られた16進の数字で、受信を開始するコマンドです。 (例 ; 0x23, 0x43, 0x00)
Data Bit Position	データビットが受信データの文字列のどこにいるか定義します。形式は次のとおりです。 ElementName = {Xn[MSB.. LSB]} ここで、 - “ElementName”はSPIの入力デバイスの名前です。もし現在のSPI入力デバイスの場合、代わりにyを使用してください。 - {}は、括弧内の項目が複数回繰り返されることを意味します。 - XnはSPIの入力装置から受信したn番目の単語であり、nは0から始まります。 - MSB.. LSBは、ワードの有効ビットの位置を定義します。
Input Range	入力範囲を定義するパラメータVmaxを指定します。このパラメータは、SPIデバイスがA/Dコンバータの場合にのみ有効です。デバイスの変換モードがDCの場合、入力範囲は0~Vmax。デバイスの変換モードがACの場合、入力範囲は-Vmax/2~Vmax/2。
Scale Factor	出力スケールファクタのKscale。スケールファクタが0の場合、SPIデバイスは、A/Dコンバータではなく、結果はDSPがSPI通信で受信したものとまったく同じになります。そうでない場合は、SPIデバイスはA/Dコンバータであり、結果はこの係数とA/D

	変換モードに基づいてスケールされます。
ADC Mode	デバイスのA/D変換モード。変換モードはDCまたはACのどちらでもかまいません。このパラメータはデバイスがA/Dコンバータの場合にのみ有効であることに注意してください。
Initial Value	入力の初期値です。

データのビット位置の式は、SPI 入力デバイスのデータ長を定義します。例えば、

$y = x1[3..0]x2[7..0]$ は、データ長が 12 であることを意味し、結果は 2 ワード目の下位 4 ビットと 3 番目のワードの下位 8 ビットです。受信データの文字列は 0x12、0x78、0xAF の場合、結果は 0x8AF です。

スケールファクタが 0 でない場合、出力は以下に基づいて拡大縮小されます。

DC 変換モードの時：

-シミュレーションの場合  $Output = Input \cdot K_{scale}$

-実際のハードの場合  $Output = Input \cdot K_{scale}$

AC 変換モードの時：

-シミュレーションの場合  $Output = Input \cdot K_{scale}$

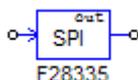
-実際のハードの場合  $Output = Input \cdot K_{scale}$

パラメータ Data\_length はデータのビット位置の計算式で計算されます。  $Output = Input \cdot K_{scale}$

## 6.10.4 SPI 出力

SPI 出力デバイスは、複数の出力チャンネルを持つものもあります。SPI 出力ブロックは、SPI 通信用の出力チャンネル、1 つの出力チャンネルに対応する 1 つの SPI 出力ブロックのロパティを定義するために使用されます。

シンボル:



仕様:

パラメータ	機能
Device Name	SPI出力デバイスの名前です。
Scale Factor	出力スケールファクタのKscale。スケールファクタが0の場合、SPIデバイスは、D/Aコンバータではなく、結果はDSPがSPI通信で受信したものとまったく同じになります。そうでない場合は、SPIデバイスはD/Aコンバータであり、結果はこの係数とD/A変換モードに基づいてスケールされます。
Output Range	出力範囲を定義するパラメータVmaxを指定します。このパラメータは、SPIデバイスがD/Aコンバータの場合にのみ有効です。デバイスの変換モードがDCの場合、入力範囲は0~Vmax、デバイスの変換モードがACの場合、入力範囲は-Vmax/2~Vmax/2。
DAC Mode	デバイスのD/A変換モード。変換モードはDCまたはACのどちらでもかまいません。このパラメータはデバイスがD/Aコンバータの場合にのみ有効であることに注意してください。
Sending Data Command	コマンドで区切られた16進の数字で、出力データを送信するコマンドです。(例； 0x23、0x43、0x00)
Data Bit Position	データビットが送信データの文字列のどこにいるか定義します。形式は次のとおりです。 ElementName = {Xn[MSB.. LSB]} ここで、

	<ul style="list-style-type: none"> <li>- “ElementName”はSPI出力デバイスの名前です。もし現在のSPI出力デバイスの場合は、代わりにyを使用してください。</li> <li>- {}は、括弧内の項目が複数回繰り返されることを意味します。</li> <li>- XnはSPI出力デバイスに送信したn番目の単語であり、nは0から始まります。</li> <li>- MSB.. LSBは、ワードの有効ビットの位置を定義します。</li> </ul>
Receiving Data Command	<p>コンマで区切られた16進の数字で、SPI出力デバイスの出力チャンネルと同期させるコマンドです。(例：0x23、0x43、0x00)</p> <p>このコマンドは、SPI出力デバイスが同期信号を持っていない場合に使用されます。</p>

データのビット位置の式は、SPI 出力デバイスのデータ長を定義します。例えば、

$y = x1[3..0]x2[7..0]$ は、データ長が 12 であることを意味し、結果は 2 ワード目の下位 4 ビットと 3 番目のワードの下位 8 ビットです。送信データの文字列が 0x12、0x78、0xAF の場合、データは 0x8AF です。

スケールファクタが 0 でない場合、出力は以下に基づいて拡大縮小されます。

DC 変換モードの時：

-シミュレーションの場合  $Output = Input \cdot K_{scale}$

-実際のハードの場合  $Output = Input \cdot K_{scale}$

AC 変換モードの時：

-シミュレーションの場合  $Output = Input \cdot K_{scale}$

-実際のハードの場合  $Output = Input \cdot K_{scale}$

パラメータ Data\_length はデータのビット位置の計算式で計算されます。

## 6.11 DSP 設定

DSP クロックブロックは外部クロック周波数と TMS320F28335 の周波数の設定ができます。

シンボル:



仕様:

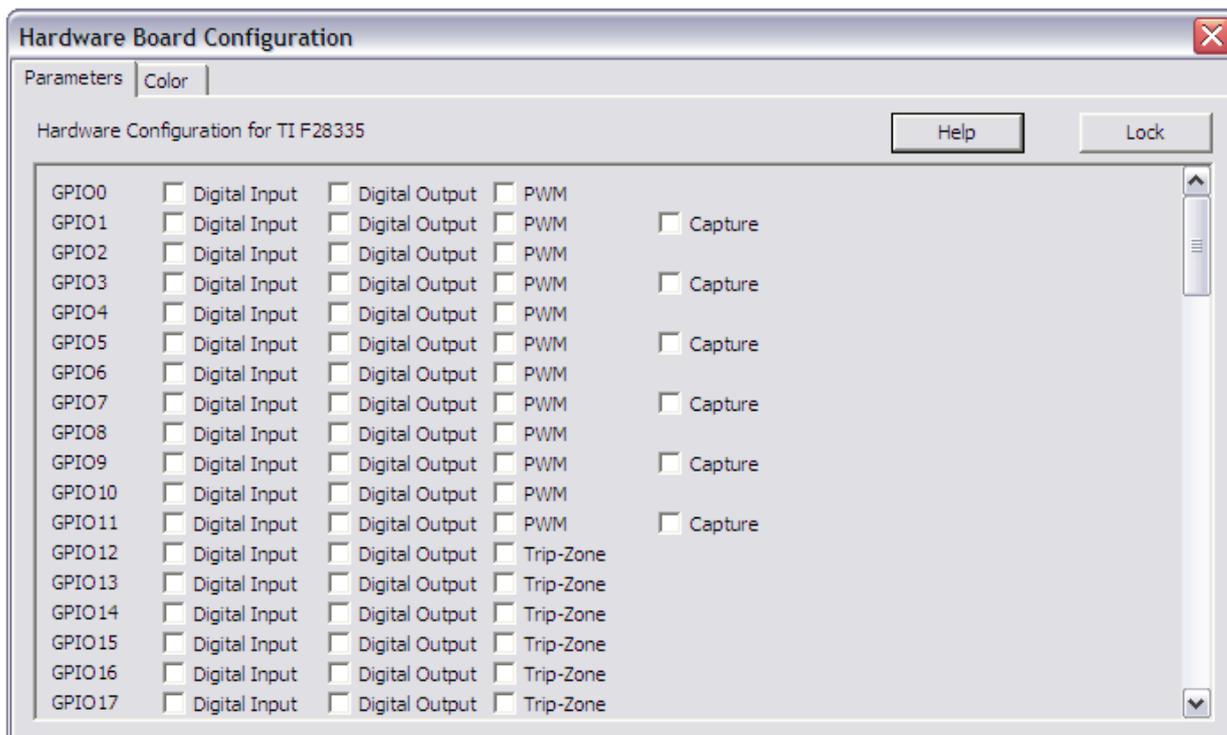
パラメータ	機能
External Clock (MHz)	DSPボードの外部クロック周波数の設定（単位はMHz）。 入力する周波数は整数値である必要があり、30MHzが最大値です。
DSP Speed (MHz)	DSPの周波数の設定（単位はMHz）。 入力する周波数は整数値である必要があり、外部クロック周波数の整数倍（1~12倍）である必要があります。また、150MHzが最大値です。

## 6.12 ハードウェアボードコンフィギュレーション

TMS320F28335 は 88 の GPIO ポート（GPIO0~GPIO87）を持っており、各ポートごとに異なった機能が割り当てられます。特定の DSP ボードでは、全てのポートが外部と繋がっている訳ではなく、またポートの機能が固定されていることもあります。ハードウェアボードコンフィギュレーション

シヨンプロックは、このような特定の DSP ボードに合わせた設定を SimCoder に持たせることができます。

シンボル:



仕様:

ハードウェアコンフィギュレーションブロックの設定画面では、各 GPIO ポートに対して設定可能な機能とその隣にチェックボックスが付いています。有効にしたい機能にチェックを入れると、チェックの入っていない他の機能は SimCoder では無効にすることができます。

例えば、GPIO1 は Digital Input、Digital Output、PWM、そして Capture の設定をすることができます。ここで PWM のところだけにチェックを入れて、他は空欄にします。すると GPIO1 を Digital Input など他の機能として使用しようとした場合、エラーメッセージが出力されます。

## 6.13 プロジェクト設定とメモリ配置

TI F28335 Hardware Target でコード生成をしようとしたとき、SimCoder は開発環境 TI Code Composer Studio(CCS)のプロジェクトファイルも作成できますので、コンパイル、リンキング、DSP へのアップロードが容易に行えます。

現在、Code Composer Studio のバージョン 3.3 に対応しています。PSIM の回路ファイル名が「test.psimsch」である場合、コード生成後、「test(C code)」というフォルダが回路ファイルと同じ

場所に作成されます。作成されたフォルダには下記のファイルが生成されています。

- test.c	: 生成された C コード
- PS_bios.h	: SimCoder F28335 のヘッダファイル
- passwords.asm	: DSP コードパスワードファイル
- test.pjt	: CCS プロジェクトファイル
- DSP2833x_Headers_nonBIOS.cmd	: ペリフェラルレジスタリンクコマンドファイル
- F28335_FLASH_Lnk.cmd	: フラッシュメモリリンクコマンドファイル
- F28335_FLASH_RAM_Lnk.cmd	: フラッシュ RAM リンクコマンドファイル
- F28335_RAM_Lnk.cmd	: RAM メモリリンクコマンドファイル

更に下記のファイルが必要となります。

- PS_bios.lib	: PSIM フォルダ内の SimCoder F28335 ライブラリ
- C28x_FPU_FastRTS_beta1.lib	: PSIM のlib フォルダ内の TI 高速浮動小数点ライブラリ

これら二つのファイルはコード生成時に自動的にプロジェクトフォルダにコピーされます。

※.c ファイルと.pjt ファイルはコード生成実行時に毎回生成及び書き換えが行われます。もしこれらのファイルを自分で別途変更を加えたい場合は、まずこれらのファイルを別のフォルダにコピーしてから行ってください。次回コード生成実行時に、強制的に上書きが行われ、手動での変更が失われる可能性があります。

## プロジェクト設定 :

CCS プロジェクトファイルには 4 つの設定があります。

- RAM Debug	: デバッグモードでコンパイル RAM メモリへ書き込みの設定
- RAM Release	: リリースモードでコンパイル RAM メモリへ書き込みの設定
- Flash Release	: リリースモードでコンパイルフラッシュメモリへ書き込みの設定
- Flash RAM Release	: デバッグモードでコンパイル RAM メモリへ書き込みの設定

RAM Debug または RAM Release 設定が選択されたとき、CCS はリンクコマンドファイル F28335\_RAM\_Lnk.cmd でプログラムとデータ空間の配置を行います。

Flash Release 設定が選択されたとき、CCS はリンクコマンドファイル F28335\_FLASH\_Lnk.cmd でプログラムとデータ空間の配置を行います。

Flash RAM Release 設定が選択されたとき、CCS はリンクコマンドファイル F28335\_FLASH\_RAM\_Lnk.cmd でプログラムとデータ空間の配置を行います。メモリ配置は RAM Release と同じです。

リリースモードでのコンパイルはデバッグモードに比べると早く済みます。また RAM Release または Flash RAM Release が最も早いです。RAM Debug は遅く、Flash Release が最も遅いです。

が、開発段階では一般的にデバッグのし易さから RAM Debug から始めて、RAM Release へそして Flash Release や RAM Release へ移行していきます。

## メモリ配置：

生成されたリンクファイルではメモリ配置は下記のように定義されます。

RAM Debug、RAM Release、Flash RAM Release の設定の場合：

<b>RAM Memory</b> 0x0000 - 0x07FF interrupt vectors stack 0x8000 - 0xFFFF program and data space
---

Flash Release の設定の場合：

<b>Flash Memory</b> 0x300000 - 0x33FFFF program password etc.
--

<b>RAM Memory</b> 0x0000 - 0x07FF interrupt vectors stack 0x8000 - 0xFFFF data space
---

RAM Debug や RAM Release では、SimCoder はプログラム空間やデータ空間を RAM メモリの 0x8000 から 0xFFFF まで定義します。

※もしプログラム空間とデータ空間を合わせたものが 0x8000 ワードを超える場合は Flash Release 設定を使う必要があります。

## 7 PE-PRO/F28335 Hardware Target

---

オプションモジュールPE-PRO/F28335 Hardware TargetとSimCoderの組み合わせによって、PSIMの制御回路図から浮動小数点型DSP TMS320F28335を搭載したパワエレ用制御ボードPE-PRO/F28335 (Mywayプラス製) で使用できるCコードを生成することができます。

生成されるコードにはパワエレ専用ライブラリPEOS/F28335の関数が使われており、ソフトウェアのデバッグやアップロードには統合開発環境PE-View9が必要になります。PE-Viewではソフトウェアを動作させた状態でリアルタイムに変数の変化を波形で観測、または変数値の読み書きが行えるため、ソフトウェアのデバッグ、パラメータ調整が極めて簡単にできるというメリットがあります。

PE-PRO/F28335 Hardware Target ライブラリには下記の機能ブロックがあります。

- 3-phase PWM 生成器と空間ベクトル PWM 生成器
- PWM 生成器の Start/Stop 機能ブロック
- A/D 変換器
- デジタル入力、エンコーダ、トリップゾーンの各機能
- デジタル出力、キャプチャ PWM の各機能

複数のサンプリング周波数でコードを生成する回路の場合、SimCoder は優先的に PWM 割り込みを使います。他のサンプリング周波数では、タイマ 1 割り込み、タイマ 2 割り込みの順に使われます。3 つ以上のサンプリング周期がある場合は対応する割り込み関数をメイン関数内で実行する形にします。

PE-PRO/F28335 では PWM 生成部からハードウェア割り込みを発生させることができますので、PWM 生成ブロックに繋がっていて、PWM 生成ブロックを同じ周波数を持つ全ての素子を検索してグループ化します。各素子ブロックは自動的に配置され、出力コード内の割り込み関数として実行されます。

また、ハードウェア割り込みはデジタル入力、エンコーダ、キャプチャ、そしてトリップゾーン (PEOS/F28335 ではゲートブロック機能として表現されています。) にもあります。各ハードウェア割り込みは割り込みブロックから生成され、(5.4 章参照) 割り込みブロックは割り込み関数と繋げる必要があります。(割り込み関数はサブ回路で作ります。) 二つ以上の割り込みを使うときはそれぞれの機能に割り込みブロックと割り込み関数を設定します。

本章では PE-PRO/F28335 Hardware Target ライブラリ内の各素子の使い方について解説します。

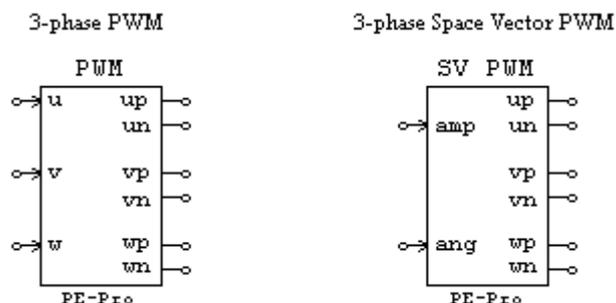
### 7.1 PWM 生成器

PE-PRO/F28335 には大きく 2 つのタイプの PWM 生成器があり、三相 PWM 生成器 (PWM123 と PWM456) と単純 PWM 生成器 (APWM5 と APWM6) です。三相 PWM 生成器については本節

で解説を行います、単純 PWM 生成器は 7.6 節で解説を行います。

PE-PRO/F28335 には、2 つのタイプの三相 PWM 生成器があります。一つは一般的な三角波比較 PWM 生成器、もう一つは空間ベクトル PWM 生成器です。各図とパラメータを下記に示します。

## シンボル:



図中の u、v、w は三相を示しています。(代わりに a、b、c とも呼ばれます。) p は正の出力を示し、n は負の出力を示しています。3-phase Space Vector PWM 生成器の場合は、amp は振幅入力を示し、ang は角度入力を示します。

## 仕様:

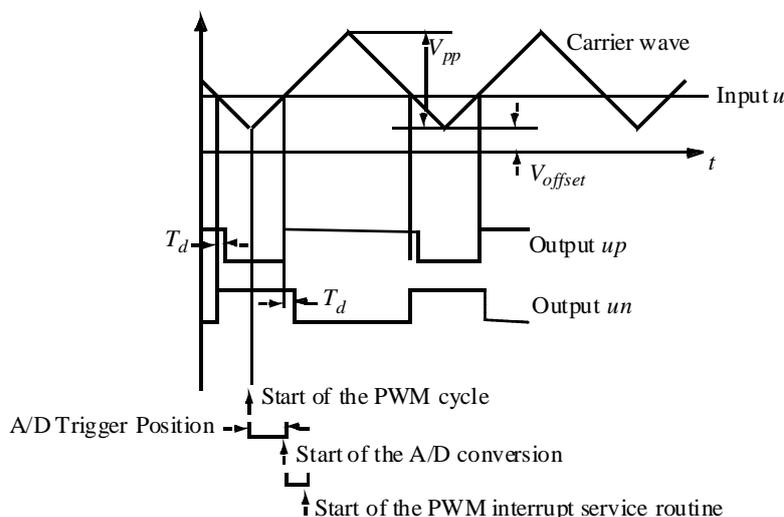
パラメータ	機能
PWM Source	PWM生成器の出力ポート設定。 - 3-phase PWM 123 : PWM1、PWM2、PWM3 - 3-phase PWM 456 : PWM4、PWM5、PWM6  3-phase PWM生成器の場合は「3-phase PWM 123」(PWM1~3)か「3-phase PWM 456」(PWM4~6)の二つを選択します。1-phase PWM生成器の場合は「PWM1」から「PWM6」までを選択します。
Dead Time	PWM生成器のデッドタイム (Td) の設定。単位は[sec]
PWM Frequency	PWM生成器の周波数の設定。単位は[Hz] PE-PRO/F28335の仕様上、1145Hzより高い値に設定してください。
Trigger ADC	A/D変換へのトリガの設定。 - 「Do not trigger ADC」 : A/D変換トリガ機能を無効にします。 - 「Trigger ADC Group A&B」 : A/D変換器のGroup AとGroup Bの両方へのトリガを有効にします。(チャンネルは1から16)
ADC Trigger Position	A/D変換トリガをかける位置の設定。 値は0~1まで設定可能です。 (例) 0の場合、PWM周期の始めでA/D変換トリガを掛けます。 1の場合、PWM周期の最後でA/D変換トリガを掛けます。
Use Trip-Zone	PWM生成器のトリップゾーンの設定。 - 「Disable」 : 対応するトリップゾーン信号を無効にします。 - 「Enable」 : PWM123はトリップゾーン1を使用し、PWM456はトリップゾーン2をそれぞれone-shotモードで使用します。一度トリガが掛かると、PWM出力を開始するには指令を与える必要があります。
Peak-to-Peak	キャリア波のピーク間電圧Vppの設定。

Value	
Offset Value	キャリア波のオフセット電圧Voffsetの設定。
Initial Input Value u, v, w または Initial Amplitude, Angle	(3-phase PWM生成器のみ) u、v、w入力ノードの初期値の設定。 (3-phase Space Vector PWM生成器のみ) Amplitude、Angle入力ノードの初期値の設定。
Start PWM at Beginning	開始からPWM信号出力をするかどうかの設定。 - 「Start」：開始からPWM出力を許可します。 - 「Do not start」：「Start PWM」関数を実行するまでPWM出力をしません。

通常のPWM生成器の方は、PWMキャリア波形は三角波を用いています。PWM生成器の入出力波形を下の図で示しています。up、un はブロックの出力 up、un のノードに対応しており、PE-PRO/F28335 ボードから実際に出力されるものにあたります。

入力 u がキャリアより大きい値になったときに出力 up はローレベルになります。これは TI F28335 Hardware Target とは逆になっています。これは PE-PRO/F28335 ボードの仕様として、ボードから出力されるPWM信号はローアクティブでありDSPから出力される信号と反転したのになるからです。二つのモジュールで仕様が違う点にご注意ください。

3-phase Space Vector PWM ブロックは空間ベクトルPWM法を使った三相回路向けの信号を出力します。amp 入力は振幅にあたり、0 から 1 までの入力が有効になります。ang 入力は空間ベクトル位相角であり（単位は rad）、 $-2\pi \sim 4\pi$  までの値が有効です。



上記はデッドタイムの定義や A/D トリガが働く流れを示しています。（「Start PWM at Beginning」の欄を「Start」に設定、「ADC Trigger Position」の欄を「0」に設定した場合の流れを示しています。）A/D 変換が完了した後、PWM 割り込み関数が始まります。

また、PWM 生成器から A/D 変換器へトリガを掛けない設定のときは、PWM 周期の開始から PWM 割り込み関数が実行されます。

PWM 生成割り込みは以下の二つの方法で発生します。

- **周期割り込み** : PWM 周波数と割り込み周期は同じになります。以下の方法で生成されます。
  - Trigger ADC が選択されていない場合、PWM キャリアの始まりに PWM 生成器の割り込みが発生します。
  - Trigger ADC が選択されている場合、PWM 生成器は A/D 変換器に変換開始のトリガを掛けます。そして上図のように変換が完了したタイミングで A/D 変換器から割り込みが発生します。
- **トリップゾーン割り込み** : PE-PRO/F28335 には 2 つのトリップゾーンがあります。PWM123 はトリップゾーン 1 を使い、PWM456 はトリップゾーン 2 を使います。トリップゾーン 1 信号が 0 から 1 に変わったタイミングで PWM123 にトリップゾーン割り込みが発生します。同様にトリップゾーン 2 信号が 1 から 0 に変わったタイミングで PWM456 にトリップゾーン割り込みが発生します。トリップゾーン割り込みに入る前に全ての PWM 出力はハイインピーダンスになります。

## 7.2 Start PWM と Stop PWM

Start PWM と Stop PWM ブロックは、PWM 生成器の動作を開始する、または停止する機能があります。ブロック図とパラメータは下記に示します。

シンボル:



仕様:

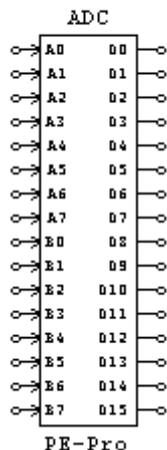
パラメータ	機能
PWM Source	PWM生成器の出力の選択。 <ul style="list-style-type: none"> <li>- 3-phase PWM123</li> <li>- 3-phase PWM456</li> <li>- APWM5</li> <li>- APWM6</li> </ul>

## 7.3 A/D 変換器

PE-PRO/F28335 には 12 ビット 16 チャンネルの A/D 変換器があります。A/D 変換器は Group A と Group B の二つに分かれています。ブロック図とパラメータは以下に示します。

シンボル:

## A/D Converter



### 仕様:

パラメータ	機能
ADC Mode	A/D変換器の動作モードを設定します。 <ul style="list-style-type: none"> <li>- Continuous : 連続変換を行います。変換値の読み値は、最後に変換が行われた値になります。</li> <li>- Start/Stop(16-Channel) : PEOS/F28335のマニュアルにおいてワンサイクルスキャンと呼ばれている機能です。PWM生成器からのトリガによって全ての入力チャンネルを指令に従って一回だけ変換します。</li> </ul>
Ch Ai or Bi Mode	各A/D変換チャンネルの入力モードの設定。 チャンネルにはAiとBiがあり、それぞれiは0~7の値を取ります。 <ul style="list-style-type: none"> <li>- AC : -1.5Vから+1.5Vの入力が可能なac入力になります。</li> <li>- DC : 0Vから+3Vの入力が可能なdc入力になります。</li> </ul>
Ch Ai or Bi Output Range	各A/D変換チャンネルの出力範囲設定。 チャンネルにはAiとBiがあり、それぞれiは0~7の値を取ります。

A/D 変換は Continuous モードに設定すると自律的に実行されます。Start/Stop モードに設定すると PWM 生成器からトリガが掛かったときに変換が実行されます。このモードで PWM 生成器からトリガを掛けない設定にした場合、PWM 割り込みの始まりで A/D 変換を行うようにします。

DC モードでは A/D 変換器の入力範囲は 0V から 3V であり、出力範囲は 0V から Vrange までになります。AC モードでは A/D 変換器の入力範囲は-1.5V から+1.5V であり、出力範囲は-Vrange から+Vrange までになります。

出力値は下記のようにスケーリングされます。

$$\text{DC モードのとき} \quad V_{oi} = V_i \times V_{range} / 3$$

$$\text{AC モードのとき} \quad V_o = V_i \times V_{range} / 1.5$$

(Vi は A/D 変換器の入力ポートの値)

例えば、DC モードで Vrange=100、Vi=3 ならば Vo=100。AC モードで Vrange=100、Vi=1.5 ならば Vo=100。

入力値は入力可能範囲内の値を設定してください。入力可能範囲外の値を入力した場合、出力は制限値でクランプされ、ワーニングメッセージを出力します。

※また、下記の制約があることにもご注意ください。

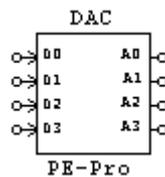
A/D 変換器同時に受け付けができる PWM トリガは 1 つだけです。

PWM 生成器から A/D 変換器へのトリガが無効の場合、変換グループへの入力信号は PWM 生成器の周波数ではなく、別の回路のサンプリング周波数を使います。このような場合は、A/D 変換器を Continuous モードに設定して使うことをお勧めします。

## 7.4 D/A 変換器

PE-PRO/F28335 には 12 ビット 4 チャンネルの D/A 変換器があります。ブロック図とパラメータを以下に示します。

シンボル:



仕様:

パラメータ	機能
Ch Di Mode	各D/A変換チャンネルの入力モードの設定。 チャンネル <i>i</i> は0~3の値を取ります。 - AC : -Vrangeから+Vrangeの入力が可能なac入力になります。 - DC : 0Vから+Vrangeの入力が可能なdc入力になります。
Ch Di Input Range	各D/A変換チャンネルの入力範囲設定。 チャンネル <i>i</i> は0~3の値を取ります。

D/A 変換器の出力範囲は 0V から+5V です。DC モードでは入力範囲は 0V から+Vrange までになります。AC モードでは D/A 変換器の入力範囲は-Vrange から+Vrange になります。

出力値は下記のようにスケーリングされます。

$$\text{DC モードのとき} \quad V_o = V_i \times 2.5 / V_{\text{range}} + 2.5$$

$$\text{AC モードのとき} \quad V_o = V_i \times 5 / V_{\text{range}}$$

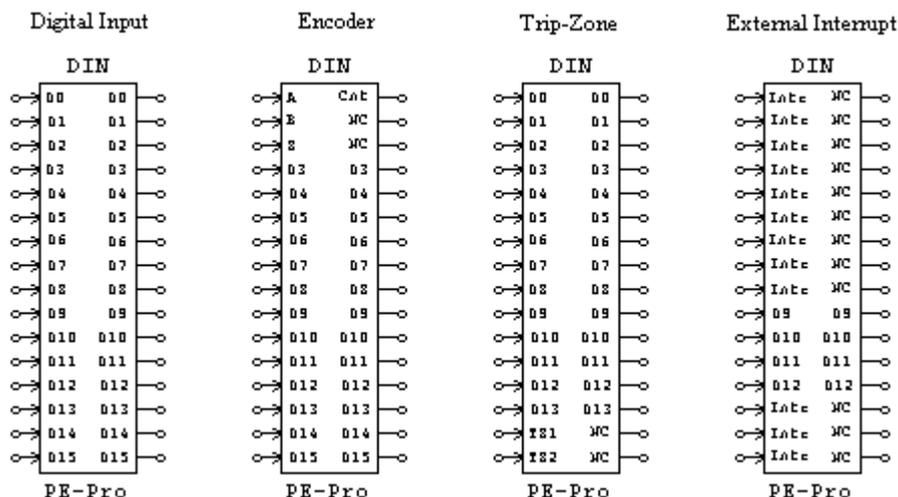
( $V_i$  は D/A 変換器の入力ポートの値)

例えば、DC モードで  $V_{\text{range}}=20$ 、 $V_i=10$  ならば  $V_o=3.75$ 。AC モードで  $V_{\text{range}}=20$ 、 $V_i=10$  ならば  $V_o=2.5$ 。

## 7.5 デジタル入力/エンコーダ/トリップゾーン

デジタル入力/エンコーダ/トリップゾーンブロックは、デジタル入力、エンコーダ、トリップゾーンそして外部割込みの機能を一つのブロックにまとめたものです。各機能に設定したときのブロック図とパラメータを以下に示します。

シンボル:



仕様:

パラメータ	機能
Ch Di Mode	各チャンネルの機能モードの設定。 チャンネル <i>i</i> は0~15の値を取ります。 <ul style="list-style-type: none"> <li>- Digital input : 全チャンネルで利用可能。</li> <li>- Encoder : チャンネル<i>i</i>=0のみで利用可能。</li> <li>- Trip-zone 1 or 2 : チャンネル<i>i</i>=14と15で利用可能。</li> <li>- External interrupt : チャンネル<i>i</i>=1~18、13~15で利用可能。</li> </ul>
Use Z Signal	Z信号（基準信号）の有効無効設定。 チャンネル <i>i</i> =1,2,3でエンコーダ機能を使う場合に利用します。
Counting Direction	カウンタの方向の設定。（チャンネル <i>i</i> =0,1,2） <ul style="list-style-type: none"> <li>- Forward : カウンタは増加方向に進みます。</li> <li>- Reverse : カウンタは減少方向に進みます。</li> </ul>
Encoder Resolution	接続するエンコーダの分解能の設定。（チャンネル <i>i</i> =0,1,2） 4096に設定した場合、カウンタ値が4095になった次のタイミングで0にクリアされます。0に設定した場合、クリアは行われません。

### エンコーダの場合 :

チャンネル Din0、1、2 はエンコーダ入力に設定することができます。Din0 は A 信号になり、Din1 は B 信号、そして Din2 は Z 信号になります。Din0 の出力のラベルが Cnt に変わり、このポートからエンコーダカウンタ値が出力されます。

### トリップゾーンの場合 :

チャンネル Din14 はトリップゾーン 1 に働き、チャンネル Din15 はトリップゾーン 2 に働きます。入力のトリップゾーン 1 信号が 0 から 1 に立ち上がったとき PWM123 にトリガが掛かり、トリップゾーン 2 信号が 0 から 1 に立ち上がったとき PWM456 にトリガが掛かります。

トリップゾーンと接続して割り込みブロックを使う場合は、割り込みブロックの Device Name のパラメータはトリップゾーンブロック名ではなく、PWM 生成器の名前にしてください。例えば、PWM 生成器の名前が「PWM\_G1」、トリップゾーン 1 のブロック名が「TZ1」ならば、割り込みブロックの名前は「TZ1」ではなく「PWM\_G1」にしてください。(この場合は割り込みブロックの Channel Number のパラメータは使用しません。)

## 外部割り込みの場合：

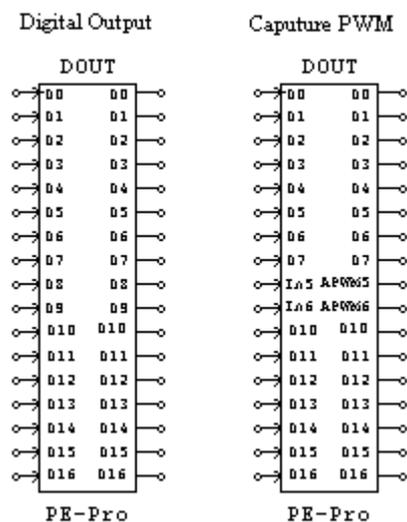
外部割り込み入力チャンネルに設定されている場合、そのチャンネルに 0 から 1 への立ち上がり信号が来たタイミングで割り込みが発生します。

PE-PRO/F28335 は最大で 7 つの外部割り込み発生源を持っており、Din3、4、5、6、14 の中で 2 つまで外部割り込みを利用でき、Din0、1、2、7、8、13 で 5 つまで利用できます。

## 7.6 デジタル出力/単純 PWM

デジタル出力/単純 PWM ブロックはデジタル出力とチャンネル 8 と 9 を利用する単純 PWM (TI 社のデータシートでは APWM) の機能を一つのブロックにまとめたものです。チャンネル 8 は APWM5(GPIO48)として、チャンネル 9 は APWM6(GPIO49)として設定できます。各機能に設定したときのブロック図とパラメータを以下に示します。

### シンボル:



## 仕様:

パラメータ	機能
Ch D8 Mode	チャンネルD8のモード設定。 Digital OutputとAPWM5のどちらかを選択します。
APWM5 Frequency	PWM生成器の周波数の設定。単位は[Hz] PE-PRO/F28335の仕様上、1Hzより高い値に設定してください。
APWM5 Peak-to-Peak Value	キャリア波のピーク間電圧Vppの設定。
APWM5 Offset Value	キャリア波のオフセット電圧Voffsetの設定。
APWM5 Initial Value	APWM5入力ノードの初期値の設定。
Start APWM5 at Beginning	開始からAPWM5号出力をするかどうかの設定。 - 「Start」：開始からAPWM5出力を許可します。 - 「Do not start」：「Start PWM」関数を実行するまでAPWM5出力をしません。
Ch D9 Mode	チャンネルD9のモード設定。 Digital OutputとAPWM6のどちらかを選択します。
APWM6 Frequency	PWM生成器の周波数の設定。単位は[Hz] PE-PRO/F28335の仕様上、1Hzより高い値に設定してください。
APWM6 Peak-to-Peak Value	キャリア波のピーク間電圧Vppの設定。
APWM6 Offset Value	キャリア波のオフセット電圧Voffsetの設定。
APWM6 Initial Value	APWM6入力ノードの初期値の設定。
Start APWM6 at Beginning	開始からAPWM6号出力をするかどうかの設定。 - 「Start」：開始からAPWM6出力を許可します。 - 「Do not start」：「Start PWM」関数を実行するまでAPWM6出力をしません。

単純 PWM 生成器 APWM5 と APWM6 は機能的に 3-phase PWM ブロックより制限されています。

例えば、単純 PWM 生成器は PWM 割り込みを発生させることはできますが、A/D 変換器への割り込みやトリップゾーン信号を発生させることはできません。

## 8 PE-Expert3 Hardware Target

---

オプションモジュールPE-Expert3 Hardware TargetとSimCoderの組み合わせによって、PSIMの制御回路図から浮動小数点型DSP TMS320C6713を搭載したパワエレ用制御システムPE-Expert3 (Mywayプラス製)で汎用的に使用できるCコードを生成することができます。

SimCoderはMywayプラスのPE-Expert3 DSP開発プラットフォームの以下のボードに対応しています。

- DSP ボード(Texas Instruments TMS320C6713 浮動小数点DSP付き)
- PEVボード

システムは以下のソースから4つの割り込みを扱うことができます：

Timer0割り込み

Timer1割り込み

PWM発生器割り込み

デジタル入力、キャプチャ、およびエンコーダからの割り込み

PWMサンプリングレートでの割り込みは、PWM発生器割り込み(INT5)を使用します。制御システムでは他のサンプリングレートがあるとき、最初にTimer0割り込み、次にTimer1割り込みを使用します。制御システムでは3つを超えるサンプリングレートがある場合、対応する割り込みルーチンはソフトウェアによってメインプログラムで扱われます。

次に、ハードウェアについて説明します。

### 8.1 PEV ボード

PEVボードには次のハードウェアと機能があります。

PWM発生器およびスタート・ストップ機能

A/D変換器

デジタル入力

デジタル出力

アップ・ダウンカウンタ

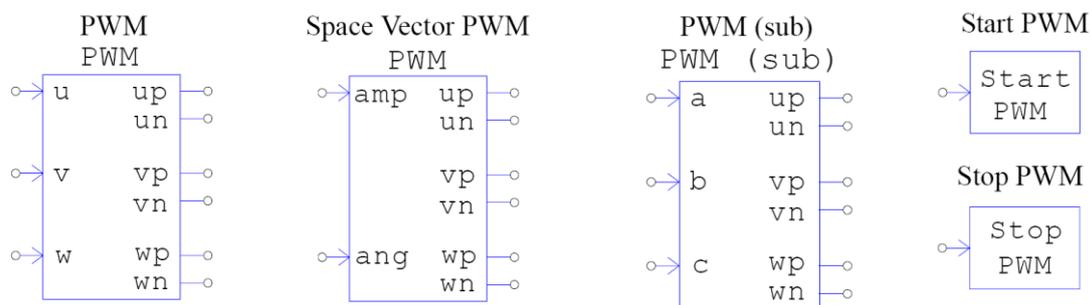
エンコーダ

キャプチャ

#### 8.1.1 PWM 発生器

PWM信号を発生するために5つの素子が準備されています。PWM(sub)発生器、PWM発生器、空間ベクトルPWM発生器、スタートPWM機能、ストップPWM機能の5つです。

## シンボル:



### PWM発生器と空間ベクトルPWM 発生器の仕様:

パラメータ	機能
Board No	PWM発生器を含んでいるPEVボードのボード番号
Channel No	PWM発生器のチャンネル番号(0か1)
Dead Time	PWM発生器のデッドタイム(s)
Carrier Frequency	PWM発生器のキャリア周波数(Hz)
Start PWM at Beginning	「Start」を設定すると最初からPWM信号を発生する。「Do not start」を設定すると「Start PWM」機能を使用してPWM信号を開始する必要があります。

### PWM(sub)発生器の仕様 :

パラメータ	機能
Board No	PWM発生器を含んでいるPEVボードのボード番号
Channel No	PWM発生器のチャンネル番号(0か1)
Dead Time	PWM発生器のデッドタイム(s)
Carrier Frequency	PWM発生器のキャリア周波数(Hz)
Peak Value	キャリア波のピークとピーク間の値
Offset Value	キャリア波のDCオフセット値
Start PWM at Beginning	「Start」を設定すると最初からPWM信号を発生する。「Do not start」を設定すると「Start PWM」機能を使用してPWM信号を開始する必要があります。

PWM発生器は、三相システムのために、正弦波PWM信号を生成します。入力「u」、「v」および「w」は、三相入力変調信号であり入力範囲は-1~1です。すなわち、入力が-1である場合、デューティサイクルは0です。また、入力が1である場合、デューティサイクルは1です。入力が0である場合、デューティサイクルは0.5です。キャリアは三角波でありデューティサイクルは0.5です。空間ベクトルPWM発生器は空間ベクトルPWM方式に基づく三相システムのためのPWM信号を発生させます。入力「amp」は空間ベクトルの振幅であり範囲は0~1です。入力「ang」は空間ベクトルの位相角であり範囲は $-2\pi$ から $4\pi$ までです。

PWM発生器の範囲は-1~1ですから、PWM発生器の入力はこの範囲を超える場合スケーリングをする必要があります。PWM(sub)の場合、このスケーリングの必要はありません。このPWM(sub)を利用すれば、コンパレータを用いたPWM回路からハードウェアPWM発生素子を利用した回路へ簡単に変更できます。

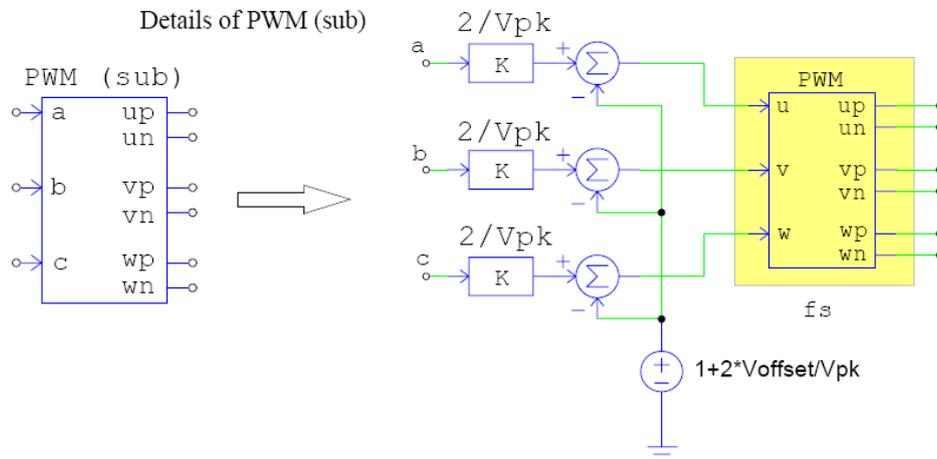


図5.5 PWM(sub)発生器の詳細

PWM(sub)の詳細を図5.5の右側に示します。この回路はスケーリング回路とハードウェアPWM素子から成ります。スケーリングすることによって、入力a、b、およびcの範囲は-1から1までとする必要はありません。キャリア電圧源と同じようにピークとピーク間の値やDCオフセット値を直接定義することができます。

PSIMのコンパレータと三角波キャリア電圧源を利用して作成したPWM(sub)素子の等価回路を図5.6に示します。

キャリア電圧源のパラメータは以下です。

V_peak_to_peak:	Vpk
Frequency:	fs
Duty Cycle:	0.5
DC Offset:	Voffset
Tstart:	0
Phase Delay:	0

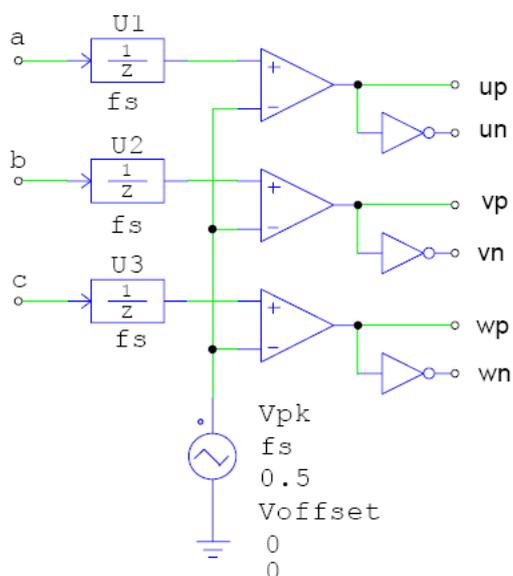


図5.6 コンパレータと三角キャリア電圧源で作成したPWM(sub)素子の等価回路

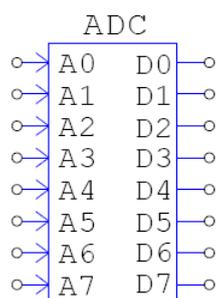
1サンプル遅れブロックU1、U2、U3はハードウェアPWM素子の固有の遅れをモデル化します。また、キャリアは三角波でそのデューティサイクルは0.5です。

PWM信号生成を開始するためには、「Start PWM」素子の入力を1Vにし、PWM信号を停止するためには「Stop PWM」素子の入力を1Vにしてください。

## 8.1.2 A/D 変換器

A/D変換器はアナログ信号をDSPが処理することができるデジタル信号へ変換します。

シンボル:



仕様:

パラメータ	機能
Board No	A/D変換器を含んでいるPEVボードのボード番号
ADi Range	iチャンネルのレンジ $V_{range}$

A/D変換器の入力レンジは-5V~+5Vです。A/D変換器の出力は以下に基づいてスケージングされます:

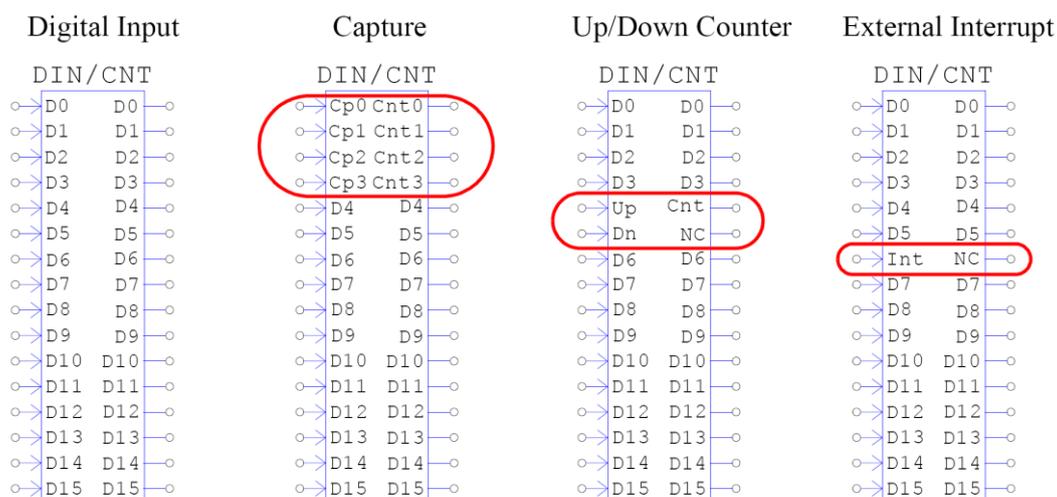
$$V_o = V_i \times V_{\text{range}} / 5$$

例えば、A/D入力 $V_i$ は2V、A/Dレンジ $V_{\text{range}}$ は20の場合、出力 $V_o$ は $2 \times 20 / 5 = 8V$ になります。

## 8.1.3 デジタル入力・キャプチャ・カウンタ

PEVボードのデジタル入力は16ピンあり、ピンD0~D3はキャプチャ素子と共有され、ピンD4およびD5はアップ・ダウンカウンタと共有されます。そしてピンD6は外部割り込みの入力として使用することができます。従って、入力/出力ピンは定義によって異なった機能に割り当てられます。また素子のシンボルは定義により変化します。各定義毎のシンボルと仕様を以下に示します。

シンボル:



仕様:

パラメータ	機能
Board No	PEVボードのボード番号
Input/Capture $i$	以下の1つになります: <ul style="list-style-type: none"> <li>■ <i>Digital Input <math>i</math></i>: 入力ピン<math>D_i</math>はデジタル入力になります。</li> <li>■ <i>Capture <math>i</math></i>: 入力ピン<math>D_i</math>はキャプチャの入力になり、出力ピン<math>D_i</math>はキャプチャ<math>i</math>の出力になります。入力・出力ピンのキャプションはCp<math>_i</math>とCnt<math>_i</math>に変化します。</li> </ul>
Counter Source $i$	カウンタソースの名前です。これは、汎用タイマのために「GP_TIMER」あるいはエンコーダの名前のどちらかになります
Input 4 and 5/Counter	以下の1つになります: <ul style="list-style-type: none"> <li>■ <i>Digital Input 4 and 5</i>: 入力ピンD4およびD5はデジタル入力になります。</li> <li>■ <i>Counter</i>: 入力ピンD4とD5はアップ・ダウンカウンタの入力になり、出力ピンD4はカウンタの出力になります。カウンタモードでは、出力ピンD5は使用されません。入力ピンのキャプションはD4の場合Up (アップカウンタ)、D5の場合Dn (ダウンカウンタ) に変化します。出力ピンのキャプションはD4の場合Cnt (カウンタ出力)、D5の場合NC (接続なし) に変化します。</li> </ul>
Counter Mode	入力D4とD5がカウンタ入力として定義されるとき、カウンタモードは「Up/Down」か「Direction/Pulse」のどちらかになります。
Input 6/External Interrupt	以下の1つになります: <ul style="list-style-type: none"> <li>■ <i>Digital Input 6</i>: 入力ピン6はデジタル入力になります。</li> <li>■ <i>External Interrupt</i>: 入力ピン6は外部割り込みの入力になります。入力ピンのキャプションはInt (Interrupt) および出力ピンのキャプションはNC (接続なし) に変化します。</li> </ul>

## キャプチャとして:

キャプチャ素子には4つの入力があります。入力がLOWからHIGHに変わる時、ソースのカウンタ値をキャプチャして出力ポートをから出力します。キャプチャ素子はエンコーダあるいは汎用タイマのカウンタ値をキャプチャすることができます。

## カウンタとして:

カウンタには「Up/Down」モードと「Direction/Pulse」モードの2つの操作モードがあります。カウンタが「Up/Down」モードに設定してある場合、「Up」入元にパルスが入力されるとカウンタはカウントアップし、「Dn」入元にパルスが入力されるとカウントダウンします。カウンタが「Direction/Pulse」モードで設定してある場合、「Pulse」入元にパルスが入力されたときに「Dir」入元が0であればカウントアップし、そして「Dir」入元が0ならばカウントダウンします。

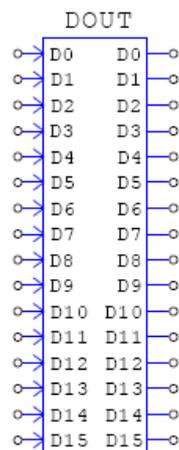
## 外部割り込みとして:

入元ピンD6が外部割り込みとして定義されている場合、入元が0から1に変化すると割り込みが発生します。

## 8.1.4 デジタル出力

デジタル出力のシンボルと仕様を以下に示します。

### シンボル:



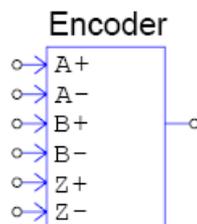
### 仕様:

パラメータ	機能
Board No	デジタル出力素子を含んでいるPEVボードのボード番号

## 8.1.5 エンコーダ

エンコーダはモータ駆動システムで位置測定に使用されます。エンコーダはオープンコレクタあるいは差動入力モードで動作させることができます。

シンボル:



仕様:

パラメータ	機能
Board No	エンコーダを含んでいるPEVボードのボード番号
Encoder Mode	エンコーダの動作モードを設定します。この設定により、オープンコレクタの入力を使用するか、差動入力を使用するかを選択します。
Counting Direction	カウンタ方向(「Forward」か「Reverse」)を設定します。「Forward」に設定するとエンコーダがカウントアップし、「Reverse」に設定するとカウントダウンします。

## 8.2 PE-Expert3 ランタイムライブラリ

PE-Expert3は高速計算のためにランタイムライブラリを提供します。SimCoderを利用してコードを作成する時、PSIM素子の対応するPE-Expert3ランタイムライブラリを使用したコードが生成されます。

PSIM素子と対応するPE-Expert3ランタイムライブラリを以下の表に示す。

PSIM素子	PE-Expert3ランタイムライブラリ
Sine or Sin (in rad.)	mwsin(float x)
Cosine or Cosine (in rad.)	mwcoss(float x)
Tangent Inverse	mwarctan2(float y, float x)
Square-root	mwsqrt(float x)
ABC-alpha/beta Transformation	uvw2ab(float u, float v, float w, float *a, float *b)
AB-alpha/beta Transformation	uv2ab(float u, float v, float *a, float *b)
AC-alpha/beta Transformation	uw2ab(float u, float w, float *a, float *b)
alpha/beta-ABC Transformation	ab2uvw(float a, float b, float *u, float *v, float *w)
alpha/beta-dq Transformation	ab2dq(float a, float b, float *d, float *q)
dq-alpha/beta Transformation	dq2ab(float d, float q, float *a, float *b)
xy-r/angle Transformation	xy2ra(float y, float x, float *a, float *b)
r/angle-xy Transformation	ra2zy(float r, float a, float *x, float *y)

## 9 General Hardware Target

オプションモジュールGeneral Hardware TargetとSimCoderの組み合わせによって、PSIMの制御回路図から汎用的に使用できるCコードの雛形を生成することができます。

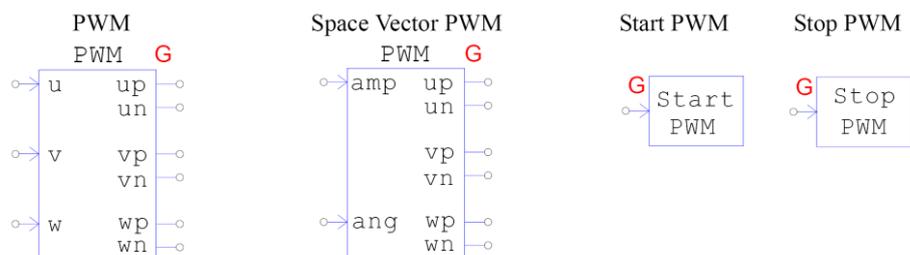
汎用ハードウェアには次のハードウェア素子があります。

- PWM発生器
- A/DおよびD/A変換器
- デジタル入出力
- エンコーダ
- レゾルバ
- キャプチャ
- 割り込み

### 9.1 PWM 発生器

PWM信号を発生するために4つの素子が準備されています。PWM発生器、空間ベクトルPWM発生器、スタートPWM機能およびストップPWM機能の4つです。

シンボル:



仕様:

パラメータ	機能
Device ID	PWM発生器を含んでいるデバイスの番号(整数)
Dead Time	PWM発生器のデッドタイム
Start PWM at Beginning	「Start」を設定すると最初からPWM信号を発生し、「Do not start」を設定すると回路の中でPWM信号発生を開始する必要があります。

PWM発生器は、三相システムのために、正弦波PWM信号を生成します。入力「u」、「v」および「w」は、三相入力調整信号向けであり入力範囲は-1~1です。すなわち、入力が-1である場合、デ

ューティサイクルは0です。また、入力が1である場合、デューティサイクルは1です。入力が0である場合、デューティサイクルは0.5です。

空間ベクトルPWM発生器は空間ベクトルPWM方式に基づく三相システムのためのPWM信号を発生させます。「amp」は空間ベクトルの振幅であり範囲は0~1です。「ang」は空間ベクトルの位相角であり範囲は $-\pi$ から $\pi$ までです。

PWM 信号を開始するためには、「Start PWM」素子の入力を1Vにし、PWM 信号を停止するためには「stop PWM」素子の入力を1Vにしてください。

コード生成において使用されるPWM発生器関連の関数を以下に示します。

### **PWM の場合:**

初期設定のときに以下の関数が呼ばれます。

**GeneralPwmInit** ({device ID}, {PWM frequency}, {dead time}):

[この関数はPWM発生器のPWM周波数とデッドタイムの初期設定をします]

**GeneralPwmIntrVector** ({interrupt function name})

[この関数はPWM発生器の割り込み関数を定義します]

**SetGeneralPwmUvw** ({device ID}, {initial value of input U}, {initial value of input V}, {initial value of input W})

[この関数はPWM発生器の入力U,V,Wの初期値を設定します]

以下の関数はPWM発生器の入力値を設定します。

**SetGeneralPwmUvw** ({device ID}, {input U value}, {input V value}, {input W value})

[この関数はPWM発生器の入力U,V,Wの値を設定します]

### **空間ベクトル PWM の場合 :**

初期設定のときに以下の関数が呼ばれます。

**GeneralPwmInit** ({device ID}, {PWM frequency}, {dead time}):

[この関数はPWM発生器のPWM周波数とデッドタイムの初期設定をします]

**GeneralPwmIntrVector** ({interrupt function name})

[この関数はPWM発生器の割り込み関数を定義します]

**SetGeneralPwmRa** ({device ID}, {initial value of the amplitude input R},{initial value of the angle input A})

[この関数は空間ベクトルPWM発生器の振幅と角度の初期値を設定します]

PWM発生器の入力値を設定する時以下の関数を呼ばれる :

**SetGeneralPwmRa** ({device ID}, {amplitude input R}, {angle input A})

[この関数は空間ベクトルPWM発生器の振幅と角度の値を設定します]

## Start PWM の場合:

この素子の入力信号が「1」になると以下の関数が呼ばれます。

**StartGeneralPwm** ({device ID}, 1):

[この関数はPWM発生器を起動します]

## Stop PWM の場合:

この素子の入力信号が「1」になると以下の関数が呼ばれます。

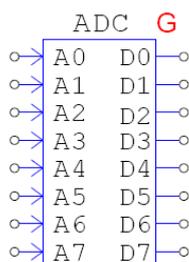
**StopGeneralPwm** ({device ID}, 0):

[この関数はPWM発生器を停止します]

## 9.2 A/D 変換器

A/D変換器はアナログ信号をDSPが処理することができるデジタル信号へ変換します。

シンボル:



仕様:

パラメータ	機能
Device ID	A/D変換器を特定するのに使用される整数番号
ADC Mode	<p>A/D変換器の動作モード。A/D返還器は5つモードがあります :</p> <p><i>ADC always ready:</i> このモードでは、A/D変換器のデータはいつでも準備ができています。</p> <p><i>1 sampling buffer:</i> 変換器は1つのバッファを持っており、一度に1つの入力チャンネルの変換を行います。</p> <p><i>2 sampling buffers:</i> 変換器は2つのバッファを持っており、一度に2つの入力チャンネルの変換を行います。チャンネルA0とA1は一つのグループとして読まれます。チャンネルA2およびA3、チャンネルA4およびA5、チャンネルA6およびA7は別々のグループになります。</p> <p><i>4 sampling buffers:</i> 変換器は4つのバッファを持っており、一度に4つの入力チャンネルの変換を行います。 チャンネルA0、A1、A2、A3 グループとして読まれます。チャンネルA4、A5、A6、A7は一つのグループとして読まれます。</p> <p><i>8 sampling buffers:</i> 変換器は8つのバッファを持っており、一度に8つの入力チャンネルの変換を行います。チャンネルA0、A1、A2、A3、A4、A5、A6、A7グループとして読まれます。</p>
Chi Gain	iチャンネルのゲインK <sub>i</sub>

A/D変換器の入力範囲は制限されていません。A/D変換器の出力は以下の式に基づいてスケーリングされます。

$$V_o = V_i * K_i$$

A/D変換器の関連関数を以下に示します。

## ADC モードが「ADC always ready」モードにある場合：

初期設定のときに以下の関数が呼ばれます。

**GeneralAdcInit0** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):

[この関数はそれぞれのA/Dチャンネルのゲインを初期化します]

A/D変換器の値を読み込むときに次の関数が呼ばれます。

**GeneralAdcRead** ({device ID}, {ch0 value}, {ch1 value}, {ch2 value}, {ch3value}, {ch4value}, {ch5 value}, {ch6 value}, {ch7 value})

[この関数はすべてのチャンネルの値を読み込みます]

## ADC モードが「1 sampling buffer」モードにある場合：

初期設定のときに以下の関数が呼ばれます。

**GeneralAdcInit1** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):

[この関数はそれぞれのA/Dチャンネルのゲインを初期化します]

A/D変換器の値を読み込むときに以下の関数が呼ばれます。

**GeneralAdcStart1** ({device ID}, {channel number})

[この関数は特定の入力チャンネルのA/D返還器の動作を起動します。チャンネル番号は0~7です]

**GeneralAdcWait1** ({device ID}, {channel number})

[この関数はA/D値が読まれる準備ができるまで待ちます]

**GeneralAdcRead1** ({device ID}, {channel output})

[この関数は特定の入力チャンネルの値を読み込みます]

## ADC モードが「2 sampling buffers」モードにある場合：

初期設定のときに以下の関数が呼ばれます。

**GeneralAdcInit2** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):

[この関数はそれぞれのA/Dチャンネルのゲインを初期化します]

A/D変換器の値を読み込むときに次の関数が呼ばれます。

**GeneralAdcStart2** ({device ID}, {group number})

[この関数はグループの入力チャンネルのA/D変換を開始します。グループ番号は0~3です]

**GeneralAdcWait2** ({device ID}, {group number})

[この関数は入力チャンネルのグループ値が読まれる準備ができるまで待ちます]

**GeneralAdcRead2** ({device ID}, {output\_a}, {output\_b})

[この関数は特定のグループの2つ入力チャンネルの値を読み込みます。例えば、グループ番号は0の場合、output\_aおよびoutput\_bはチャンネル0とチャンネル1の値になります]

### ADC モードが「4 sampling buffers」モードにある場合：

初期設定のときに以下の関数が呼ばれます。

**GeneralAdcInit4** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):

[この関数はそれぞれのA/Dチャンネルのゲインを初期化します]

A/D変換器の値を読み込むときに次の関数が呼ばれます。

**GeneralAdcStart4** ({device ID}, {group number})

[この関数はグループの入力チャンネルのA/D変換を開始します。グループ番号は0か1です]

**GeneralAdcWait4** ({device ID}, {group number})

[この関数はグループの入力チャンネルの値を読まれる準備ができるまでに待ちます。グループ番号は0か1です。]

**GeneralAdcRead4** ({device ID}, {output\_a}, {output\_b}, {output\_c}, {output\_d})

[この関数は特定のグループの4つ入力チャンネルの値を読み込みます。例えば、グループ番号は0の場合、output\_a、output\_b、output\_c、output\_dはそれぞれチャンネル0~3の値になります]

### ADC モードが「8 sampling buffers」モードにある場合：

初期設定のときに以下の関数が呼ばれます。

**GeneralAdcInit8** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):

[この関数はそれぞれのA/Dチャンネルのゲインを初期化します]

A/D変換器の値を読み込むときに次の関数が呼ばれます。

**GeneralAdcStart8** ({device ID})

[この関数はA/D変換器を起動します]

**GeneralAdcWait8** ({device ID})

[この関数はA/D値が読まれる準備ができるまで待ちます]

**GeneralAdcRead8** ({device ID}, {output\_0}, {output\_1}, {output\_2}, {output\_3}, {output\_4},

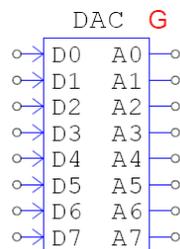
{output\_5}, {output\_6}, {output\_7})

[この関数はA/D変換器の8つ入力チャンネルの値を読み込みます。変数output\_*i* は入力チャンネル*i*の値に対応します]

## 9.3 D/A 変換器

D/A 変換器はデジタル信号をアナログへ変換します。

シンボル:



仕様:

パラメータ	機能
Device ID	D/A変換器を特定するのに使用される整数番号
Ch <i>i</i> Gain	D/A変換器チャンネル <i>i</i> のゲイン <i>K<sub>i</sub></i>

D/Aコンバータの出力は以下の式に基づいてスケーリングします。

$$V_o = V_i * K_i$$

初期設定のときに以下の関数が呼ばれます。

**GeneralDacInit** ({device ID}, {ch0 gain}, {ch1 gain}, {ch2 gain}, {ch3 gain}, {ch4 gain}, {ch5 gain}, {ch6 gain}, {ch7 gain}):

[この関数はそれぞれのD/Aチャンネルのゲインを初期化します]

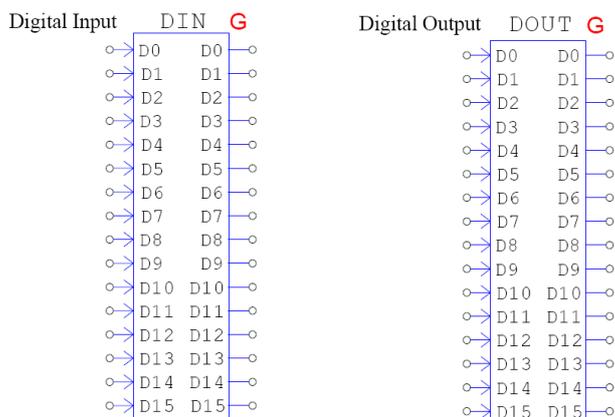
D/A変換器の値を読み込むときに次の関数が呼ばれます。

**SetGeneralDacValue** ({device ID}, {channel number}, {value})

[この関数はD/Aコンバータの出力を設定します]

## 9.4 デジタル入出力

シンボル:



## 仕様:

パラメータ	機能
Device ID	素子を特定するのに使用される整数番号

デジタル入力およびデジタル出力は以下の関数を呼び出します。

デジタル入力の場合：

**GetGeneralDinValue** ({device ID}):

[この関数はデジタル入力を読み込みます]

デジタル出力の場合：

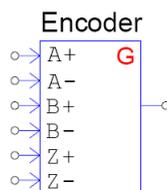
**SetGeneralDoutValue** ({device ID}):

[この関数はデジタル出力を設定します]

## 9.5 エンコーダ

エンコーダはモータ駆動システムで位置測定に使用されます。エンコーダはオープンコレクタあるいは差動入力モードで作動することができます。

シンボル:



## 仕様:

パラメータ	機能
-------	----

Device ID	エンコーダを特定するのに使用される整数番号
Use Z Signal	「True」か「False」を選択します。「True」に設定すると、エンコーダはABZモードで作動します。
Counting Direction	カウンタ方向(「Forward」か「Reverse」)を設定します。「Forward」に設定するとエンコーダがカウントアップし、「Reverse」に設定するとカウントダウンします。

初期設定のときに以下の関数が呼ばれます。

**SetEncoderMode** ({device ID}, {mode flag}):

[この関数はエンコーダの動作モードを設定します]

**ClearEncoderCount** ({device ID}):

[この関数はエンコーダのカウンタをクリアします]

**EnableEncoderIntr** ({device ID}):

[この関数はエンコーダの割り込みを許可します]

エンコーダ値を読み込むとき以下の関数が呼ばれます。

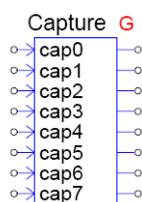
**GetGeneralEncoderValue** ({device ID})

[この関数はエンコーダ値を読み込みます]

## 9.6 キャプチャ

キャプチャ素子はエンコーダあるいは汎用タイマのカウンタ値をキャプチャすることができます。

シンボル:



仕様:

パラメータ	機能
Device ID	キャプチャを特定するのに使用される整数番号
Chi Counter Source	カウンタソースの名前。これは、汎用タイマのために「GP_TIMER」あるいはエンコーダの名前のどちらかになります。

キャプチャ素子には8つの入力があります。入力がLOWからHIGHに変わるとき、ソースのカウンタ値をキャプチャして出力ポーから出力します。

初期設定のときに以下の関数が呼ばれます。

**SetGeneralCaptureMode** ({device ID}, {channel number}, {}, {}):

[この関数はキャプチャの動作モードを設定します]

エンコーダ値を読み込むときに以下の関数が呼ばれます。

**GetGeneralCaptureCount** ({device ID})

[この関数はCapturedカウンタ値を読み込みます]

## 9.7 割り込み

次の素子が割り込みを発生することができます:

- デジタル入力
- エンコーダ
- キャプチャ

シンボル:

Interrupt



仕様:

パラメータ	機能
Device Name	割り込みを開始するデバイスの名前
Channel No	割り込みデバイスの入力チャンネル番号
Trigger Type	これはデジタル入力およびキャプチャ機能に適用する。トリガータイプの値は以下のうちの一つになります: <ul style="list-style-type: none"><li>- <i>None</i>: 割り込みは発生しない</li><li>- <i>Rising edge</i>: 入力信号の立ち上がりで割り込みが発生する</li><li>- <i>Falling edge</i>: 入力信号の立ち下がりで割り込みが発生する</li><li>- <i>Rising/falling edges</i>: 入力信号の立ち上がりおよび立ち下がりで割り込みが発生する</li></ul>

割り込み素子の使用方法を以下の図に示します。

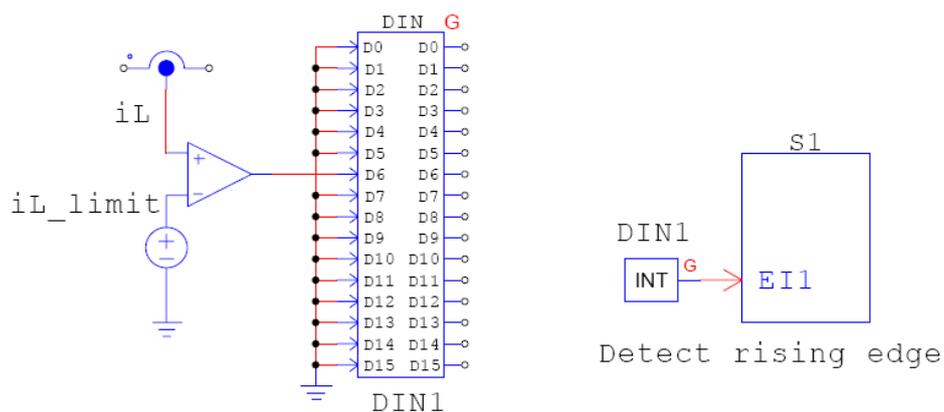


図5.8 割り込み素子の使用方法

この回路では、検出電流 $iL$ がリミット値 $iL\_limit$ と比較されます。検出電流 $iL$ がこのリミット値を超えると、パルス信号が発生してデジタル入力素子DIN1の入力に送られます。この回路では、割り込み素子の「Device Name」は「DIN1」として、「Edge Detection Type」は「Rising edge」として定義します。これにより、パルス信号が割り込みを発生させ、サブ回路S1の入カイベントポートEI1によってイベントがS1へ移行します。

## SimCoder User's Guide

---

発行: Myway プラス株式会社

〒220-0022

神奈川県横浜市西区花咲町 6-145

横浜花咲ビル

TEL.045-548-8836

FAX.045-548-8832

ホームページ: <http://www.myway.co.jp>

Eメール: [sales@myway.co.jp](mailto:sales@myway.co.jp)

---