SimCoder™

User Manual

Powersim Inc.

Mywayプラス株式会社

SimCoder User Manual

Version 12.0

Release 1

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

URL: https://www.myway.co.jp/

目次

1 SimCoder について

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-Expert4及びTI社製高速浮動小数点型DSPTMS320F2833x、F2803xなどの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。

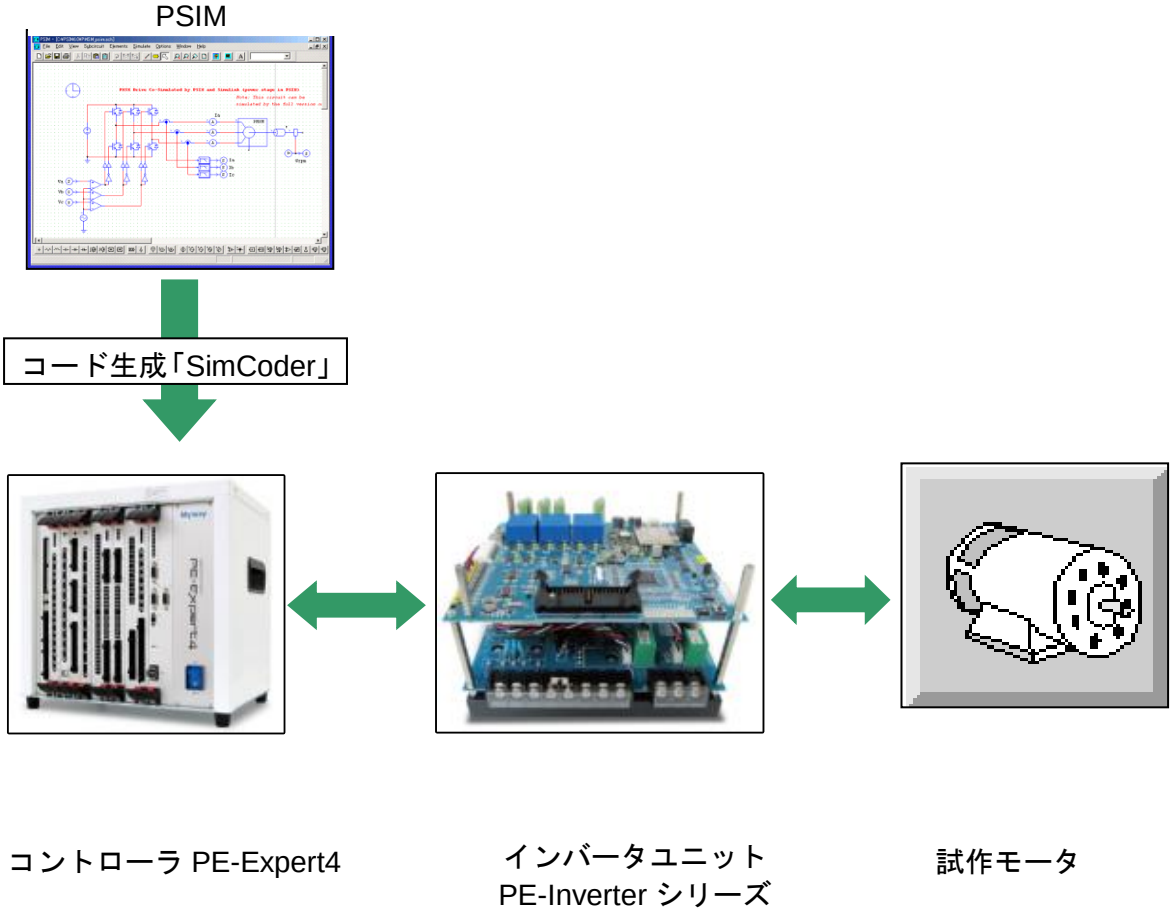


図 1.1　SimCoder と PE-Expert4 を利用した実験システム

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　SimCoder のシミュレーション制御設定(ハードウェア等)

SimCoder の設定は次の図のようにシミュレーション制御のウィンドウの SimCoder タブで、正しくコード生成できるよう各項目設定を行います。

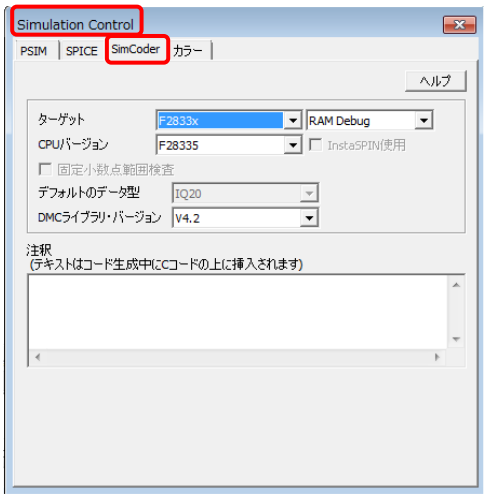


図 1.2　シミュレーション制御ウィンドウ　SimCoder タブ画面

ハードウェアターゲット

- **PE-Expert4**：Mywayプラス製のPE-Expert4 DSP開発プラットホームで使用できるCコードを生成します。**素子 → SimCoderコード自動生成 → PE-Expert4 Target** メニューを選択してPE-Expert4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-Expert4 **ターゲット**モジュールが追加オプションとして必要になります。）

- **F2833x**：TI社製 浮動点TMSF2833xシリーズのDSPを使用したハードウェアです。**素子→ SimCoderコード自動生成 → F2833x ターゲット**メニューを選択してTMS320F2833x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 F2833x **ターゲット**モジュールが追加オプションとして必要になります。）

- **F2803x**：TI社製 浮動点TMSF2803xシリーズのDSPを使用したハードウェアです。**素子→ SimCoderコード自動生成 → F2803x ターゲット**メニューを選択してTMS320F2803x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 F2803x **ターゲット**モジュールが追加オプションとして必要になります。）

- **ターゲット無し**：シミュレーションのみ。ハードウェアターゲット設定なし。

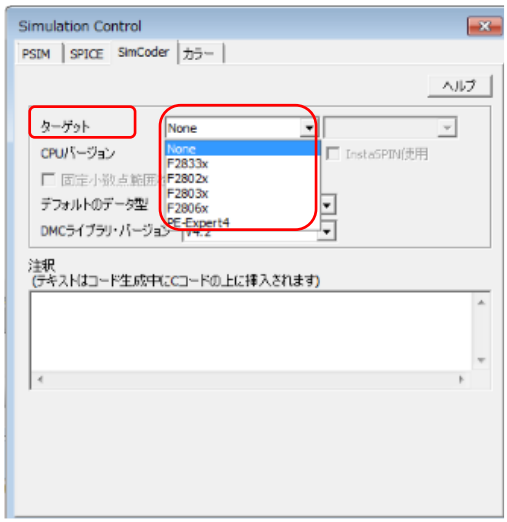


図 1.3　ターゲット選択画面

プロジェクトの構成

- *PE-Expert4*ターゲットに対しては*PE-ViewX*
- *F2833x,F2803x*ターゲットに対しては*RAM Debug,RAM Release,Flash Release*または*Flash RAM Release*を設定できます。

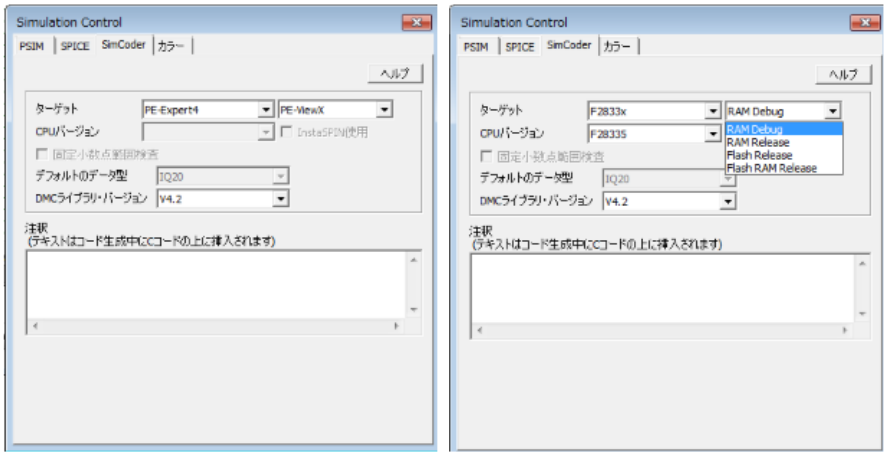


図 1.4　プロジェクト構成設定画面

CPUバージョン

- *F2833x*ターゲットに対するCPUバージョンは：F28335,28334,28332です。
- *F2803x*ターゲットに対するCPUバージョンは：F28035,28034,28033,28032,28031
- 28030です。

固定小数点範囲検査

- これはF2803x ターゲットの場合のみ使用できます。チェックボックスにチェックを入れるとSimCoderはシミュレーションデータをチェックしデータ範囲のリストを作成します。このリストは範囲境界に近いものもしくは超えているデータはハイライトされています。

デフォルトのデータ型

- ハードウェアターゲットが浮動点型の場合に自動的に選択されます。ハードウェアターゲットが固定点型かNoneの場合、プルダウンメニューから選択できます。
- ターゲット無しの場合、次の1つとなります。：Float,　Integer,IQ1,IQ2,…..,IQ30

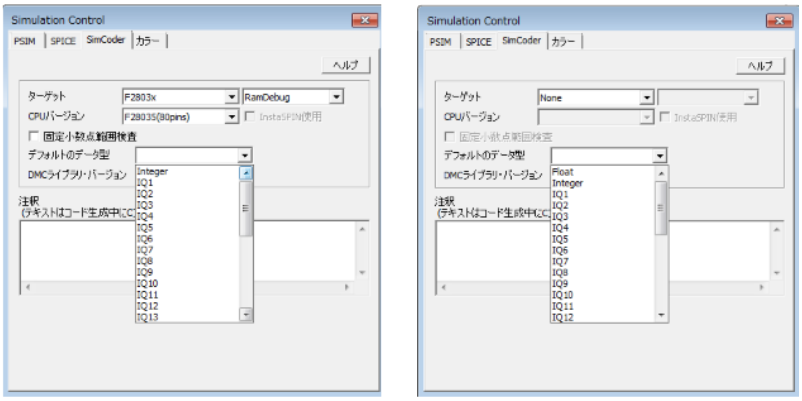


図 1.5 デフォルトデータ設定画面

DMCライブラリバージョン

- Texas InstrumentのDMC(Digital Motor Control)ライブラリがモーター制御ユーザーのためにC2000 DSPとして開発されたC言語の関数(またはマクロ)として構成されています。
- このリソースの活用するためには、SimCoderはコード生成のためのPSIM素子ライブラリへDMCライブラリ関数を組み込みます。
- これらのマクロはTIからリリースされた違ったバージョンがあります。SimCoderはバージョンV4.0,V4.1,V4.2を設定できます。一度バージョンを選択すると他のバージョンのマクロは無視されますのでご注意ください。

コメント

- コメント領域がSimCoderタブの下の方にあり、コメントとして追加できます。ここに入力したテキストはCコードの先頭にコメントとして追記されます。

1.3 コード生成用の素子

素子 →イベント制御　および **素子 → SimCoderコード自動生成**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、ハードウェアターゲットの各々の素子は**素子**>>SimCoderメニューにあります。PSIM標準ライブラリにある多数の素子もコード生成時に使用することができます。

オプション ➔設定メニューのタブAdvancedのHardware code generation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。

- コード生成用素子： CG
- TI F2833x,F2803x,F2806x,F2802x専用素子： TI

2　コード生成方法

2.1 概要

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。

2.2 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k=0.4とT=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。

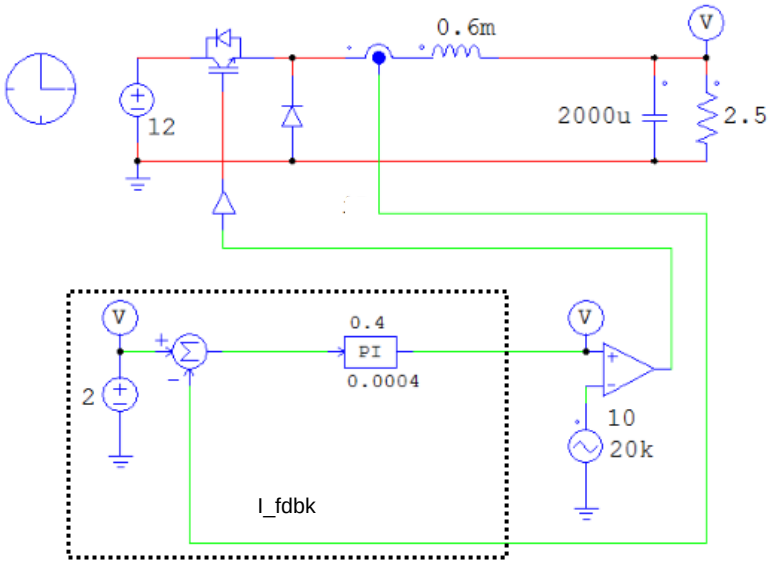


図2.1　降圧チョッパ制御回路（s領域）

2.3 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**ユーティリティ** -> **s2z コンバータ(z)**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次に示します。

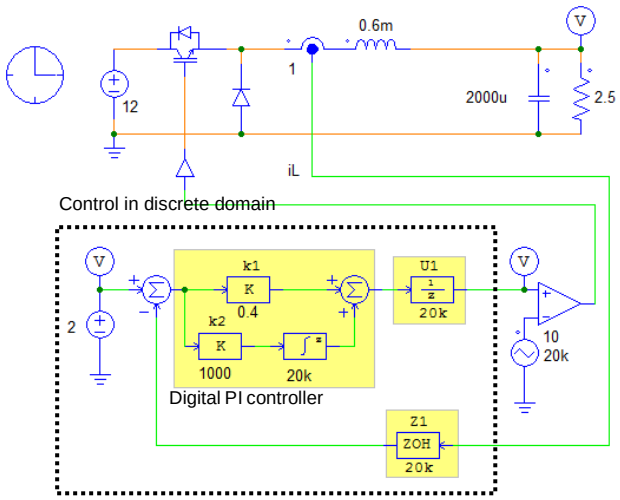


図2.2　降圧チョッパ制御回路（z 領域）

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*k*1と*k*2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iL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。

変換されたデジタル制御は安定した制御ループと所望の性能となっているはずです。もしそうなっていない場合はアナログコントロールシステムへ戻りアナログコントローラの再設計を繰り返さなければなりません。

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*T*sはサンプリング周期です。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。

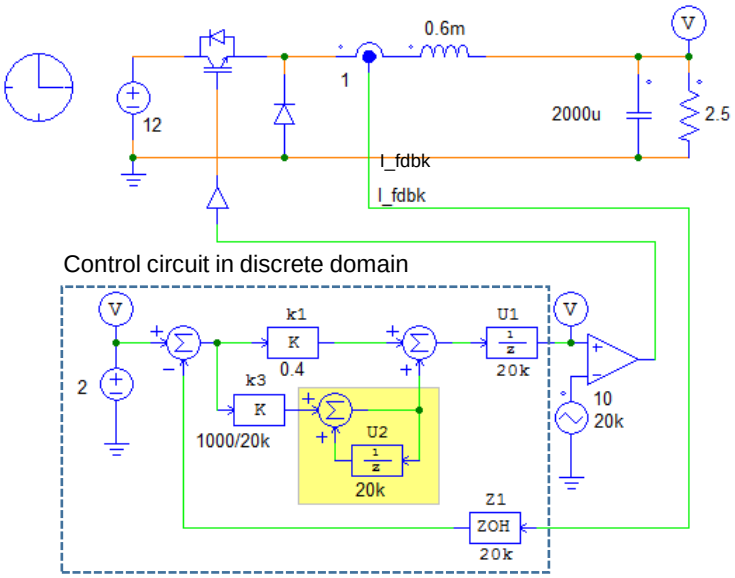


図2.3　バックワードオイラー法に基づいた降圧チョッパ制御回路（z 領域）

積分ゲインk3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。）

2.4 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F2833x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。

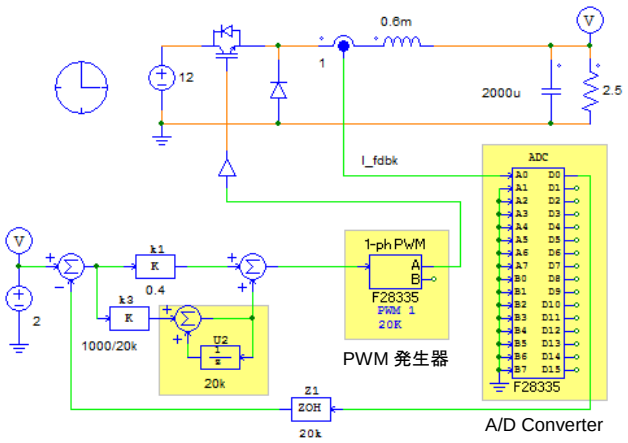


図2.4　ハードウェア素子を使用した降圧チョッパ制御回路

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。

A/D変換器を電流センサと制御サブ回路の間に置きます。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変換器の入力範囲外である場合、センサ信号をスケーリングしなければなりません。この例の場合は、A/D変換器は次のように設定します。

－ADCモード：*start-stop(8-channel)*　これはADCが片側グループのみの指令に従ってトリガーされた時に一回だけ変換します。

－Ch A0　モード：DC　設定範囲は0から3Vです。

－ChA0　ゲイン：1.0

コンパレータとキャリア波はPWM生成器に置き換えられます。PWM生成器の設定は次のようになります。

－PWMソース：PWM1　これはF28335プロセッサのPWMモジュールを指定しています。

－出力モード：Use PWM A　PWM出力ポートを指定しています。

－サンプリング周波数：20k

－キャリア波タイプ：のこぎり波（start　high）詳細は後の章にあります。

－トリガADC：Trigger ADC Group A

－ADCTriggerPosition:0

－ピーク間電圧：　10

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。

「シミュレーション制御」メニューで、SimCoderのタブにはいり

－ハードウェアタイプ：　F2833x　withRAM Debug

－CPU　Version：　F28335

－Default Fixed-Point Position：フロートに設定されるので適用外です。

注釈部分にコメントを入力するとCコードの先頭い挿入されます。編集、追加が可能です。

シミュレーションを実行し2.2の結果と近い結果が得られることを確認してみてください。

結果を確認できたら**シミュレート** -> **コード生成** を選択してCコードを生成することができます。

SimCoderで生成したCコードは次のようになります。

```
/*********************************************************************************
// This code is created by SimCoder Version 9.3.3 for TI F2833x Hardware Target
//
// SimCoder is copyright by Powersim Inc., 2009-2013
//
// Date: February 24, 2014 14:36:33
*********************************************************************************/
#include<math.h>
#include"PS_bios.h"
typedef float DefaultType;
#defineGetCurTime() PS_GetSysTimer()

interrupt void Task();

DefaultTypefGbliref = 0;
DefaultTypefGblU2 = 0;

interrupt void Task()
{
    DefaultTypefU2, fSUMP1, fSUMP3, fk3, fk1, fSUM1, fZ1, fTI_ADC1, fVDC2;

    PS_EnableIntr();
    fU2 = fGblU2;


  fTI_ADC1 = PS_GetDcAdc(0);
  fVDC2 = 2;
  fZ1 = fTI_ADC1;
  fSUM1 = fVDC2 - fZ1;
  fk1 = fSUM1 * 0.4;
  fk3 = fSUM1 * (1000.0/20000);
  fSUMP3 = fk3 + fU2;
  fSUMP1 = fk1 + fSUMP3;
  PS_SetPwm1RateSH(fSUMP1);
#ifdef_DEBUG
  fGbliref = fVDC2;
#endif
  fGblU2 = fSUMP3;
  PS_ExitPwm1General();
}

void Initialize(void)
{
  PS_SysInit(30, 10);
  PS_StartStopPwmClock(0);
  PS_InitTimer(0, 0xffffffff);
  PS_InitPwm(1, 0, 20000*1, (4e-6)*1e6, PWM_POSI_ONLY, 42822);// pwnNo, waveType, frequency, deadtime,
outtype
  PS_SetPwmPeakOffset(1, 10, 0, 1.0/10);
  PS_SetPwmIntrType(1, ePwmIntrAdc0, 1, 0);
  PS_SetPwmVector(1, ePwmIntrAdc0, Task);
  PS_SetPwmTzAct(1, eTZHighImpedance);
  PS_SetPwm1RateSH(0);
  PS_StartPwm(1);
```

```
    PS_ResetAdcConvSeq();
    PS_SetAdcConvSeq(eAdc0Intr, 0, 1.0);
    PS_AdcInit(1, !1);

    PS_StartStopPwmClock(1);
}

void main()
{
    Initialize();
    PS_EnableIntr();   // Enable Global interrupt INTM
    PS_EnableDbgm();
    for (;;) {
    }
}
```

図2.5　生成されたコードの例

生成されたコードは以下の構造になっています。

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。

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

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。

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。

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 TI の環境にロードして、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　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、**シミュレート** -> **コード生成** 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) {
        }
}
```

図2.5　生成されたコードの例

生成されたコードは以下の構造になっています。

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。

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

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。

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。

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、それハードウェアにアップロードします。

2.6 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です。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。

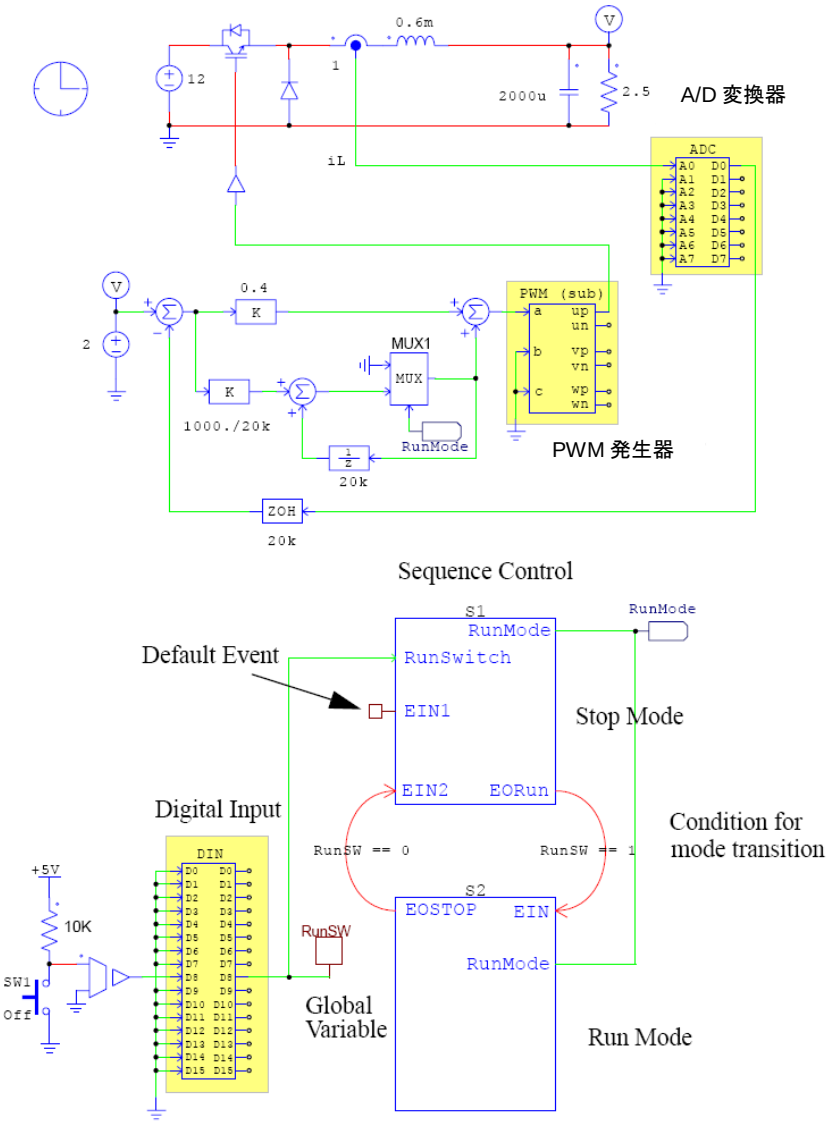


図2.6　シーケンス制御付き降圧チョッパ制御回路

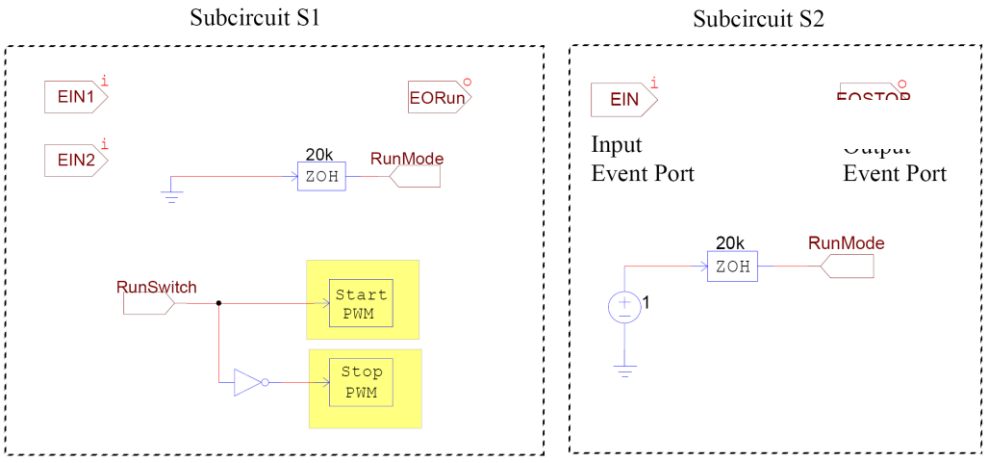


図 2.7　Run モードと Stop モードのサブ回路

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」を通ってサブ回路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の出力イベントポートEORunとS2の入力イベントポート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の出力イベントポートEOSTOPとS1の入力イベントポート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RunMode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。

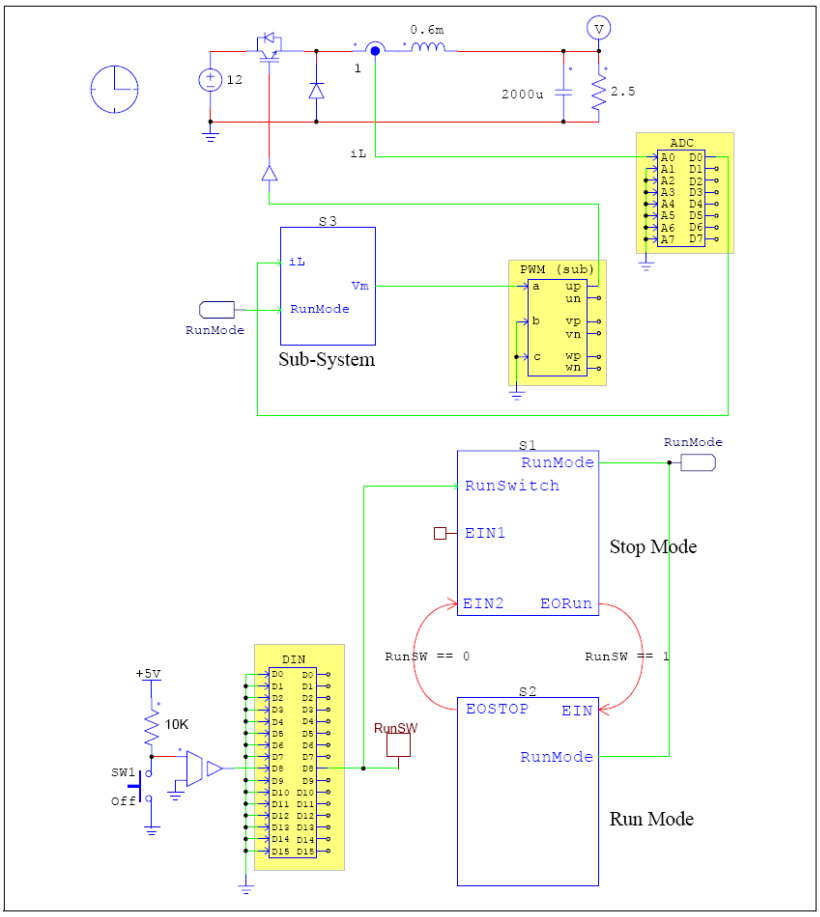


図3.1　降圧チョッパのサブシステムのコード生成

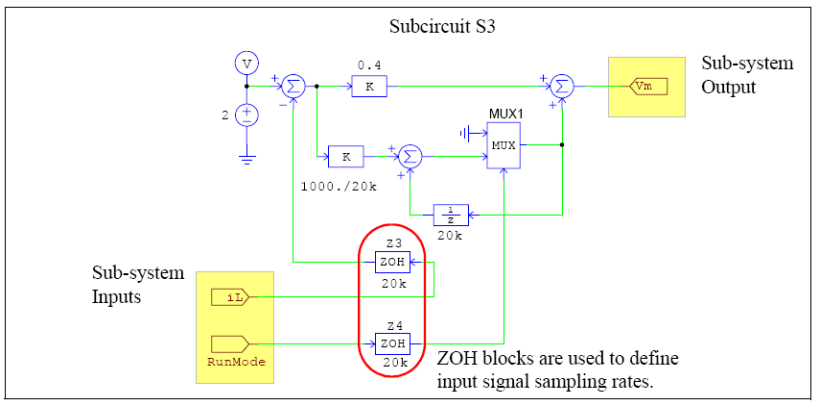


図3.2　サブシステムの制御ブロック

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　イベントコントロール

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。

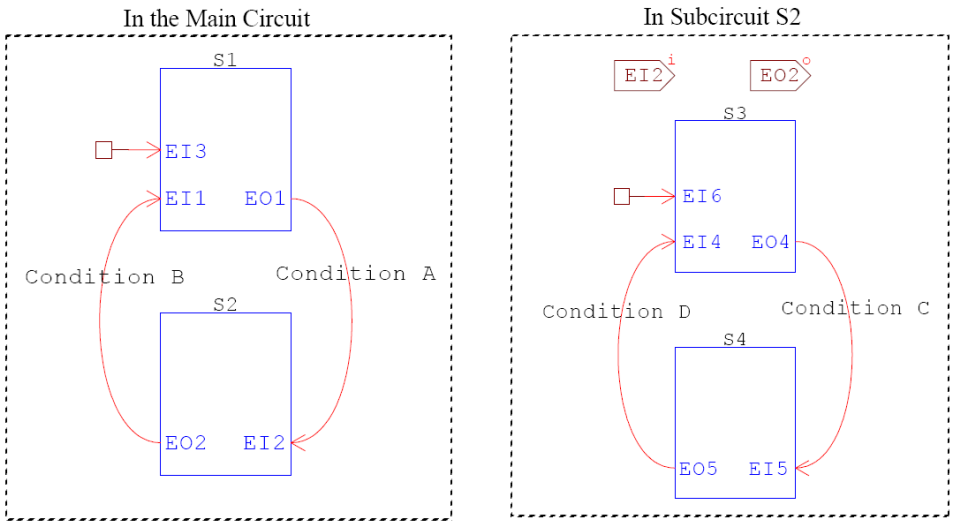


図4.1　状態遷移図

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E01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。

2つ以上の状態は共存できず、１つの状態であればいつでも存在できます。このS1とS2 のようなケースを排他的と称しています。

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、S3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
- 初回エントリーフラグ素子
- ハードウェア割り込み素子（セクション5.4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でき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移行でき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。

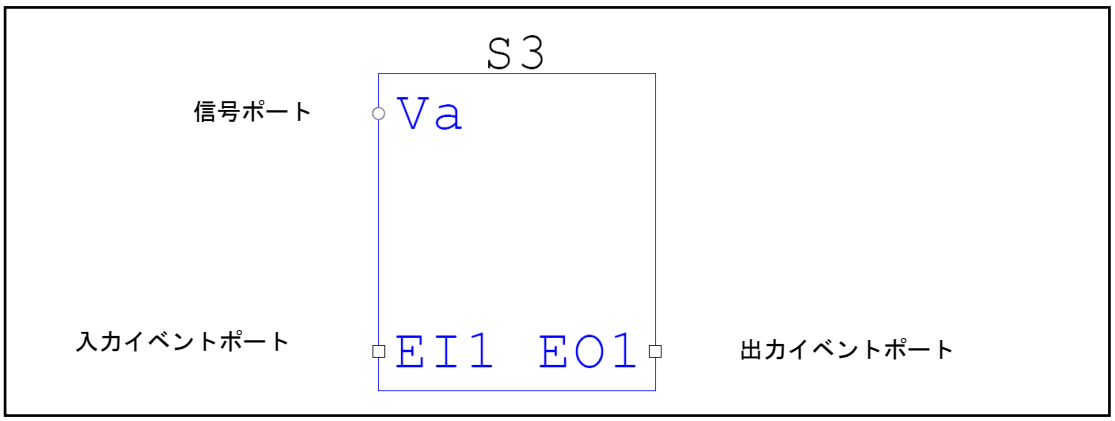


図4.2　入出力イベントポート付きサブ回路

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E01のプロパティウインドウを図4.3に示します。

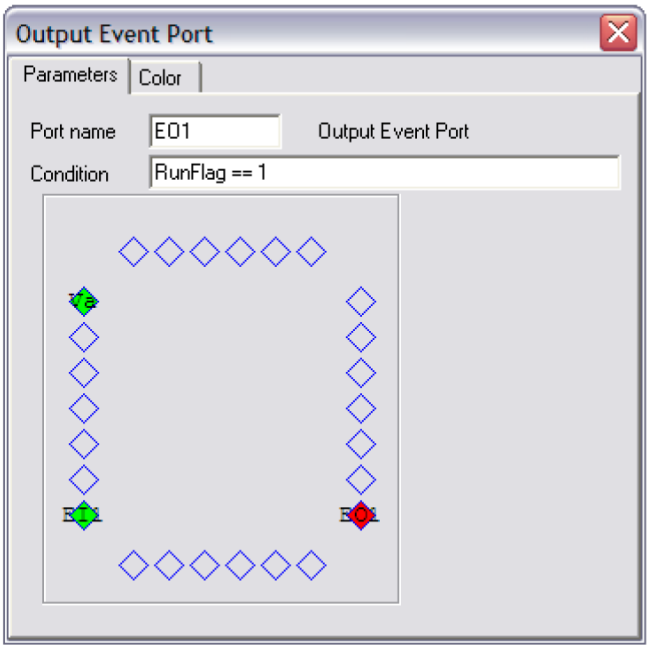


図4.3　出力イベントポート

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

入力または出力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/出力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。

PSIM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力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。

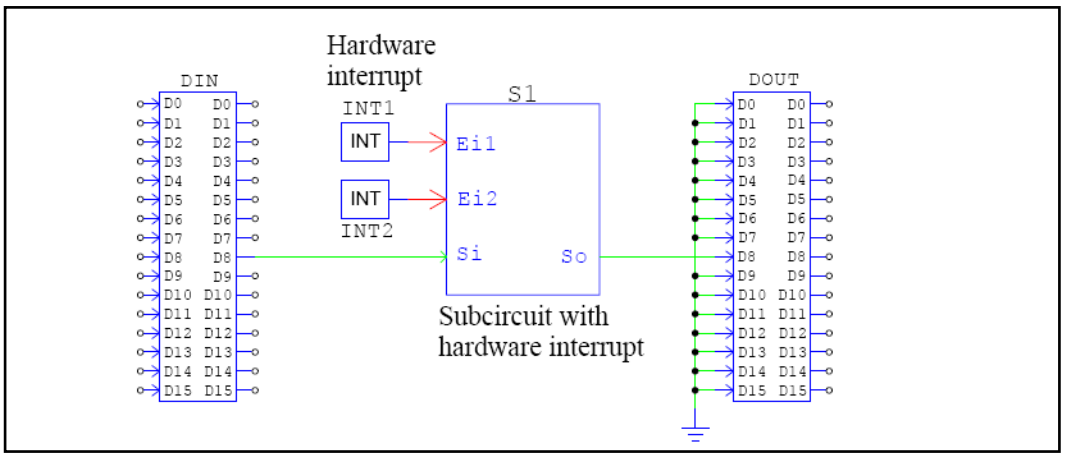


図4.4　ハードウェア割り込み付きサブ回路

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S_iはハードウェアデジタル入力そ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

ックの経由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。

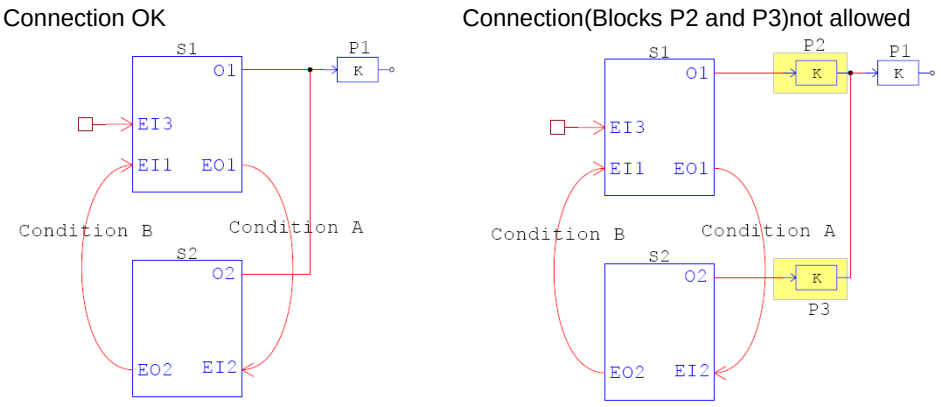


図4.5　2つのサブ回路の接続方法

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。

5 SimCoder ライブラリ

SimCoderは、MywayプラスのPE-Expert4ハードウェア開発プラットホ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・μC）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。

これらはオプション>>設定>>Advancedのタブにある“*Show image next to elements that can be used for code generation*”をチェックするとコード生成できる素子に CG がつきます。SimCoderとSPICE用素子には C/P がつきます。

数式関数ブロック、シンプルCブロック、変数t（時間）、デルタ（タイムステップ）はコード生成のためのSimCoderでは使用できません。

パラメータファイル素子とのこぎり波生成はSimCoderの特殊な使用方法はこの後の章で説明します。

素子→制御ライブラリ:

Proportional block（比例制御器）

Comparator（比較器）

Limiter（リミッタ）

Upper Limiter（上限リミッタ）

Lower Limiter（下限リミッタ）

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）

素子→制御ライブラリ→計算ブロック:

Multiplier（乗算器）

Divider（除算器）

Square-root（平方根ブロック）

Sine

Sine (in rad.)

Cosine

Cosine (in rad.)

Tangent

Arctangent

Arctangent（単位値で入力）

Arctangent 2(in rad.)

Arctangent 2(単位値で入力)Exponential (a^x)（指数ブロック）

Power (x^a)（累乗）

LOG (base e)（自然対数ブロック）

LOG10 (base 10)（常用対数ブロック）

Absolute Value（絶対値ブロック）

Sign Block（符号関数ブロック）

素子→制御ライブラリ→その他機能ブロック:

Multiplexer (2-input)（マルチプレクサ2入力）

Multiplexer (3-input 2-control)（マルチプレクサ3入力2制御）

Multiplexer (4-input)（マルチプレクサ4入力）

Multiplexer (4-input 1-control)（マルチプレクサ4入力1制御）

Multiplexer (4-input 3-control)（マルチプレクサ4入力3制御）

Multiplexer (8-input)（マルチプレクサ8入力）

Multiplexer (8-input 1-control)（マルチプレクサ8入力1制御）

空間ベクトル PWM

空間ベクトルPWM(α/β)

素子→制御ライブラリ→論理素子:

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）

素子→制御ライブラリ→デジタル制御モジュール:

Zero-Order Hold（ゼロ次ホールド）

Unit Delay（単位時間遅れ）

Integrator（積分器）

Integrator(with limiter) （積分器（制限器付き））

Differentiator （離散型微分器）

PI(with limiter)　PI(制限器付き)

External Resetable Integrator（外部リセット付き積分器）

Internal Resetable Integrator（内部リセット付き積分器）

PID(with reset)　PID(リセット付き)

1st-order Low-pass Filter　(一次ローパスフィルタ)

2nd –order Low-pass Filter (二次ローパスフィルタ)

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）

z-domain Transfer Function（ｚ領域伝達関数）

Circular Buffer(single-output)　循環バッファ(単出力)

FIR Filter(file)　FIRフィルタ（ファイル）

Digital Filter(file)　デジタルフィルタ（ファイル）

素子→その他:

Parameter File（パラメータファイル）

素子→その他→その他関数ブロックメニュー:

abc-dqo Transformation（abc-dqo変換）

dqo-abc Transformation（dqo-abc変換）

Clarke Transformation (abc-alpha/beta) （クラーク変換（abc-alpha/beta））

Clarke Transformation(ab-alpha/beta)（クラーク変換（ab-alpha/beta））

Clarke Transformation(ac-alpha/beta)（クラーク変換（ac- alpha/beta））

Inverse Clarke Transformation（alpha/beta-abc）（逆クラーク変換（alpha/beta-abc）

Park Transformation(alpha/beta-dq)　(パーク変換（alpha/beta-dq)）

Park Transformation(alpha/beta/sin/cos-dq)　(パーク変換（alpha/beta/sin/cos-dq)）

Inverse　Park Transformation(dq-alpha/beta)（逆パーク変換（dq-alpha/beta)）

Inverse　Park Transformation(dq/sin/cos-alpha/beta)（逆パーク変換（dq/sin/cos -alpha/beta)）

x/y-r/angle　Transformation(x/y-r/angle　変換)

r/angle-x/y　Transformation(r/angle-x/y　変換)Lookup Table（ルックアップテーブル）

2-D Lookup Table (integer)（2次元ルックアップテーブル(整数)）

2-D Lookup Table (interpolation)（2次元ルックアップテーブル（浮動小数点）

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（シンプルCブロック）

素子→電源メニュー:

Constant（定数素子）

Ground（グランド　接地　G）

Ground (1)（グランド　接地(1)）

Ground (2)（グランド　接地(2)）

素子→電源→電圧源メニュー:

DC（直流電圧源）

DC (battery)（直流電圧源(電池)）

Sine (正弦波電圧源)

3-ph Sine (三相正弦波電圧源)

Triangular (三角波電圧源)

Sawtooth（のこぎり波）

Square (方形波電圧源)

Step (ステップ電圧源)

Step(2-level) (ステップ電圧源(2レベル))

Grounded DC（circlre）（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、Kp=1.23と一度定義したのち、コード内ではKp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Kpとして、パラメータファイル内ではKpを「(global)Kp=0.4」と記述しています。

対象DSPが浮動小数点型F28335の場合、パラメータファイルにKpを下記のように定義します。

(global) Kp = 0.4

対象DSPが固定小数点型F28335の場合、またデータタイプがIQ-24でパラメータファイルにKpを下記のように定義します。

(global_Iq24) = 0.4

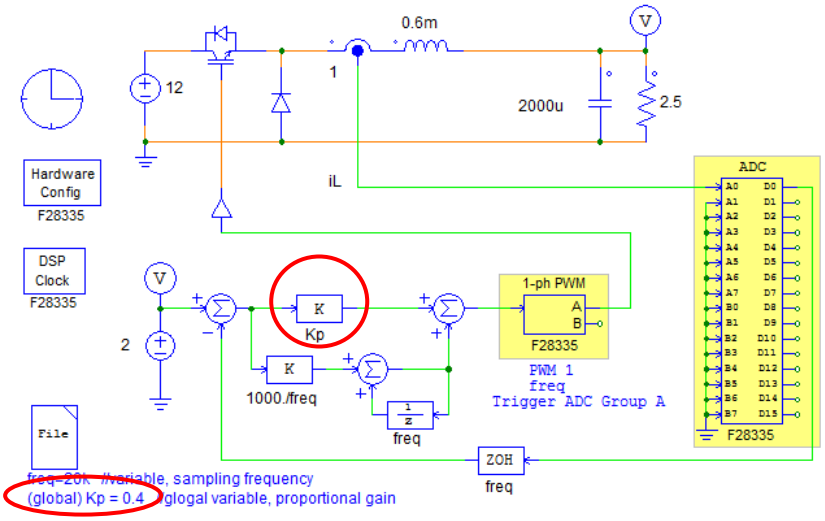


図5.1 降圧回路例1

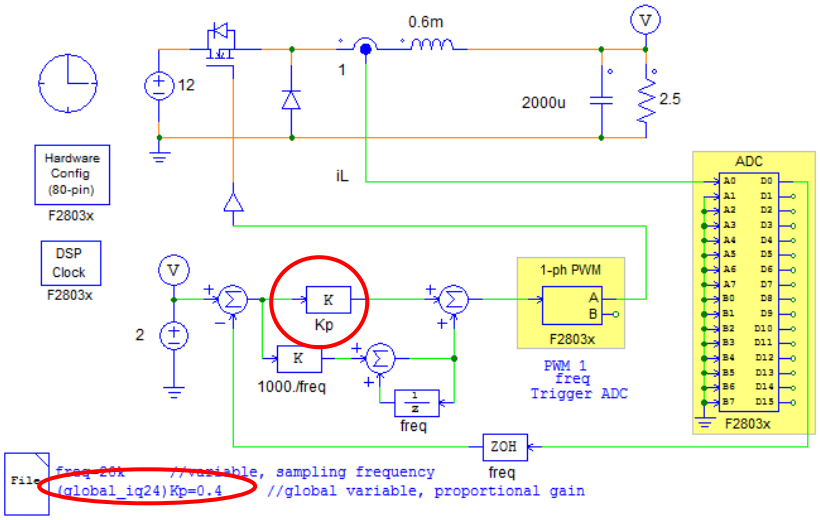


図5.2 降圧回路例2

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

```
/*******************************************************************************
// This code is created by SimCoder Version 9.1 for TI F28335 Hardware Target
//
// SimCoder is copyright by Powersim Inc., 2009-2011
//
// Date: November 08, 2011 11:26:37
*******************************************************************************/
#include      <math.h>
#include      "PS_bios.h"
typedef float DefaultType;
#define       GetCurTime() PS_GetSysTimer()

interrupt void Task();

DefaultType fGbliref = 0.0;
DefaultType fGblUDELAY1 = 0;

DefaultType Kp = 0.4;        The global parameter Kp is defined here.
interrupt void Task()
{
              DefaultType fVDC2, fTI_ADC1, fZOH3, fSUM1, fP2, fSUMP3, fUDELAY1, fP1, fSUMP1;
              PS_EnableIntr();
              fUDELAY1 = fGblUDELAY1;


              fTI_ADC1 = PS_GetDcAdc(0);
              fVDC2 = 2;
#ifdef        _DEBUG
              fGbliref = fVDC2;
#endif
              fZOH3 = fTI_ADC1;
              fSUM1 = fVDC2 - fZOH3;
              fP2 = fSUM1 * (1000./20000);
              fSUMP3 = fP2 + fUDELAY1;
              fGblUDELAY1 = fSUMP3;
              fP1 = fSUM1 * Kp;       The parameter Kp is used here.
              fSUMP1 = fP1 + fSUMP3;
              PS_SetPwm1RateSH(fSUMP1);
              PS_ExitPwm1General();
}
... ...
```

図5.3 生成されたコード

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（素子 → **電源** → **電圧源**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-Expert4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。

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。

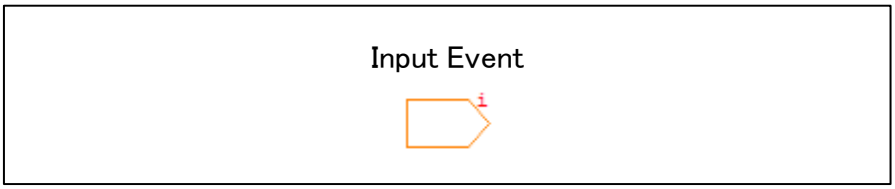
5.2 イベントコントロール素子

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



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。

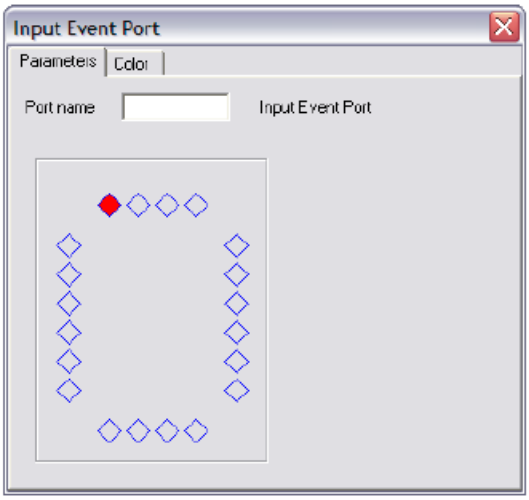


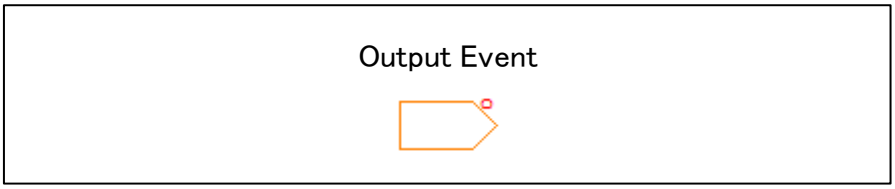
図5.4入力イベントポート

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っ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



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。

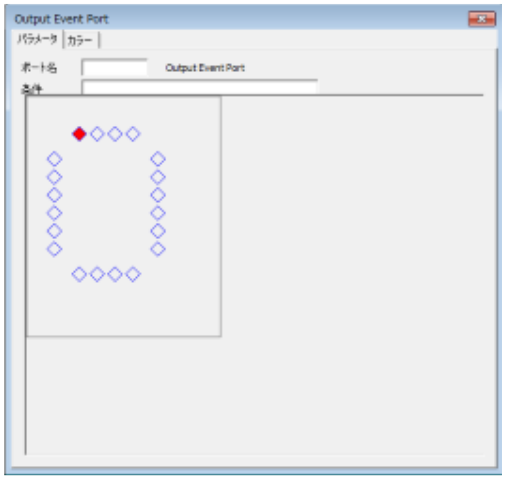


図5.5　出力イベントポート

出力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

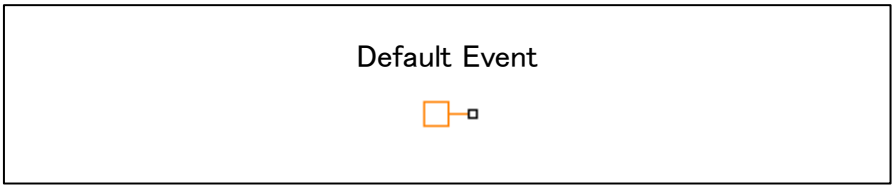
(A > B) && (C > D)

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。

シンボル



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、サブ回路の入力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力イベントポートと出力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。

メニューの　素子>>イベント制御>>イベント接続にあります。

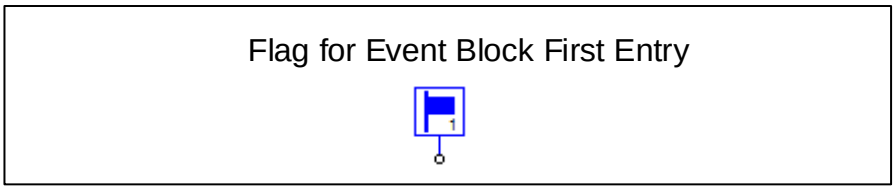
イベント接続ワイヤをダブルクリックして、出力イベントポートの条件式を修正することができます。

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：



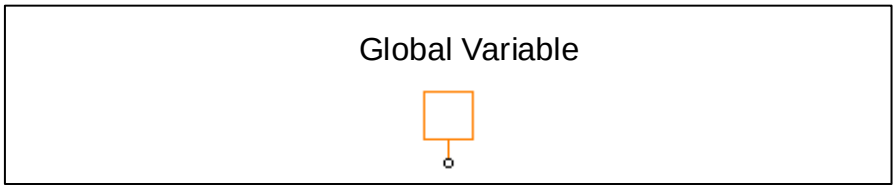
仕様:

パラメータ	機能
イベント・サブ回路ブロック名	グローバル変数名

パラメータの「Event Subcircuti 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

シンボル



仕様

パラメータ	機能
名前	グローバル変数名
初期値	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。

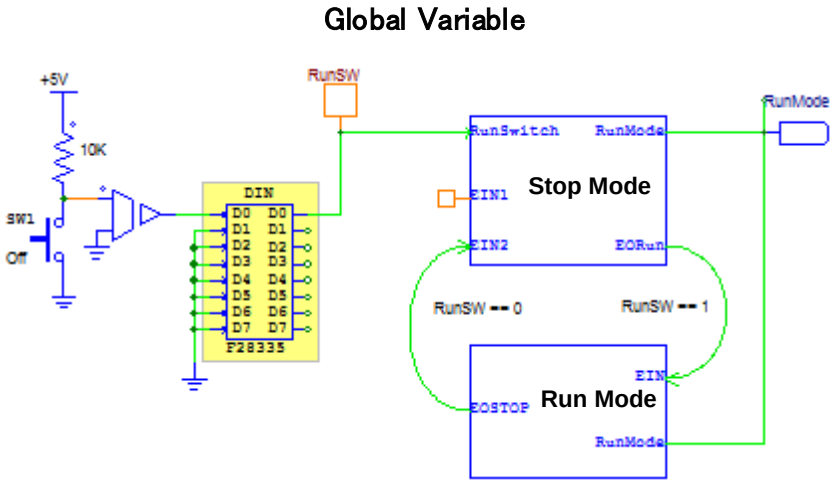


図5.6　グローバル変数の定義方法

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D0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5.7に説明します。

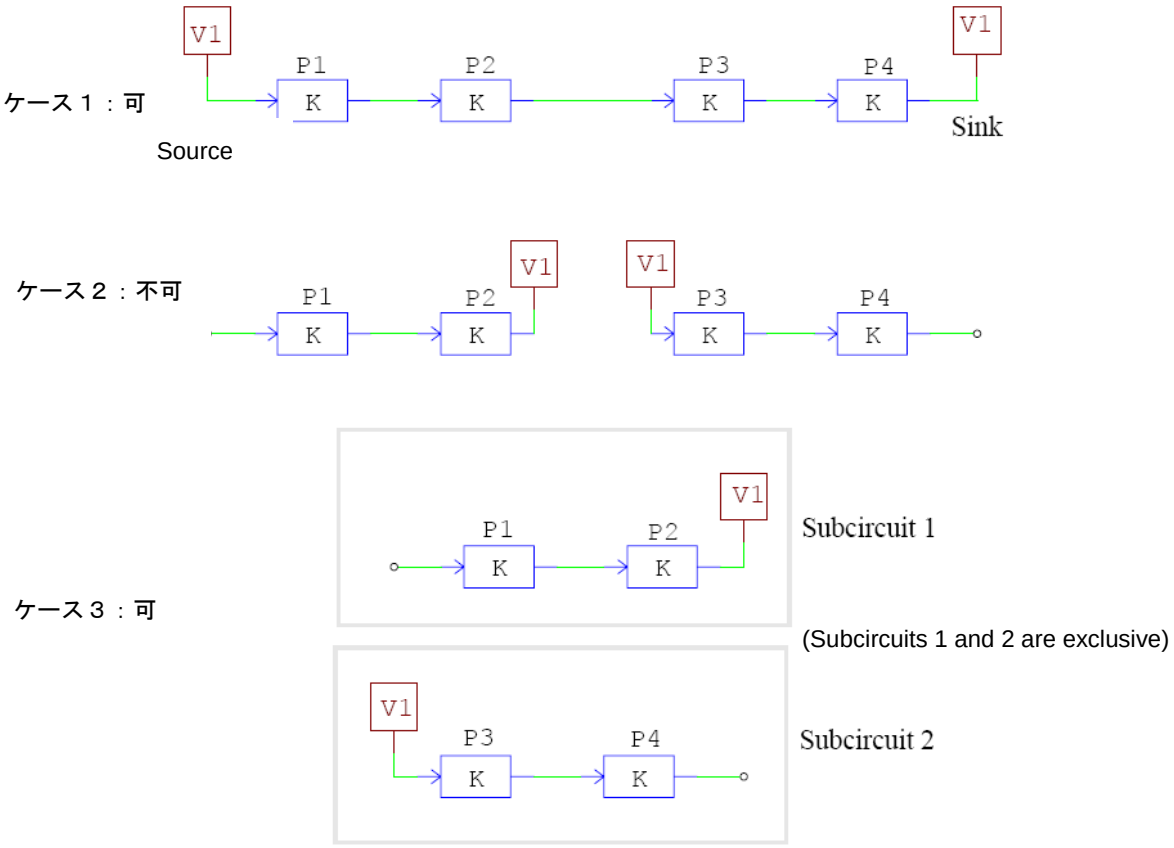


図5.7　グローバル変数の接続方法

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、この使用方法は正しいです。

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。

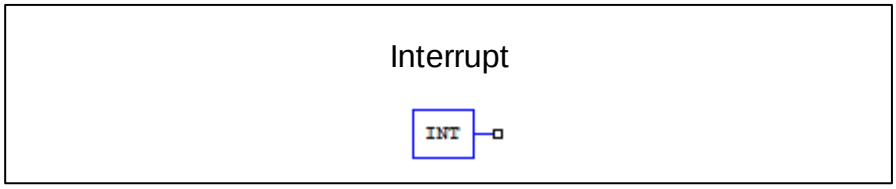
ケース3では、グローバル変数V1はサブ回路1およびサブ回路2で使用されます。システムでは、サブ回路1とサブ回路2は同時に実行しないので、このようなグローバル変数の使用は可能です。

5.4 ハードウェア割り込み

DSPは、デジタル入力、エンコーダ、キャプチャ、PWM生成器(F2833xとF2803x)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:



仕様:

パラメータ	機能
名前	割り込みを発生させる素子名の設定。
チャネル番号 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。 例えばデジタル入力のチャネルD0が割込みを発生するとチャネル番号は0に設定されなければなりません。 このパラメータは -デジタル入力、 -キャプチャ(PE-Expert4ターゲットのみ) だけで使用されます。エンコーダやPWM生成器では使えません。
エッジ検出タイプ	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。 - No edge detection：割り込みは発生しません。 - Rising edge：入力信号の立ち上がりで割り込みが発生します。 - Falling edge：入力信号の立ち下がりで割り込みが発生します。 - 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。。

割り込み素子の使用方法を次の図に示します。

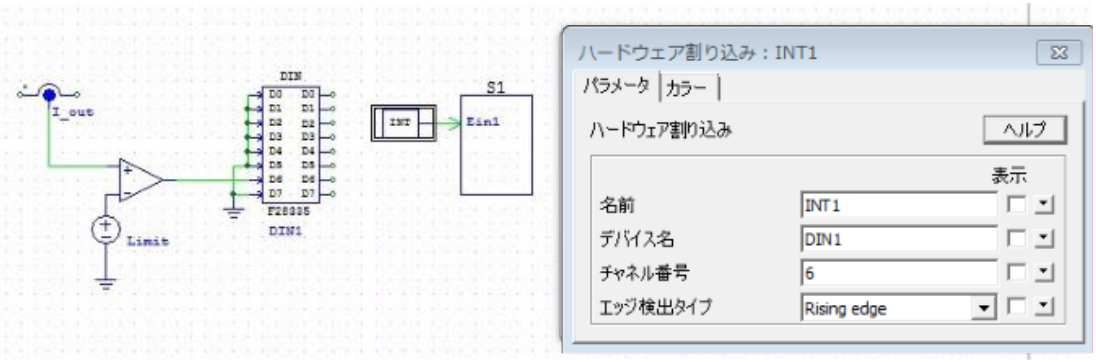


図5.8　割り込み素子の使用方法

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D6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、入力信号の立ち上がりがハードウェア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。

5.5 SimCoder C Block

外部ヘッダファイルとCソースファイルを受け入れることができ、 シミュレーションコードとハードウェアターゲットコードを簡単に組み合わせることができるCブロックです。

SimCoder Cブロックは、PSIM回路でハードウェアコード生成用に使用できるという点で、シンプルCブロックに似ています。但し、シンプルCブロックとは異なり、外部ヘッダファイルとCソースファイルを簡単にインクルードし、初期化を実行し、事前定義されたマクロを定義して呼び出すために、SimCoder Cブロックに関数が用意されています。さらに、 "Flag_Simulation"と呼ばれるフラグが用意されているため、ユーザーはCブロック内にシミュレーションコードとハードウェアコードの両方を持つことができ、フラグを使用して2つを区別することができます。下記はSimCoder Cブロックのイメージとなります。

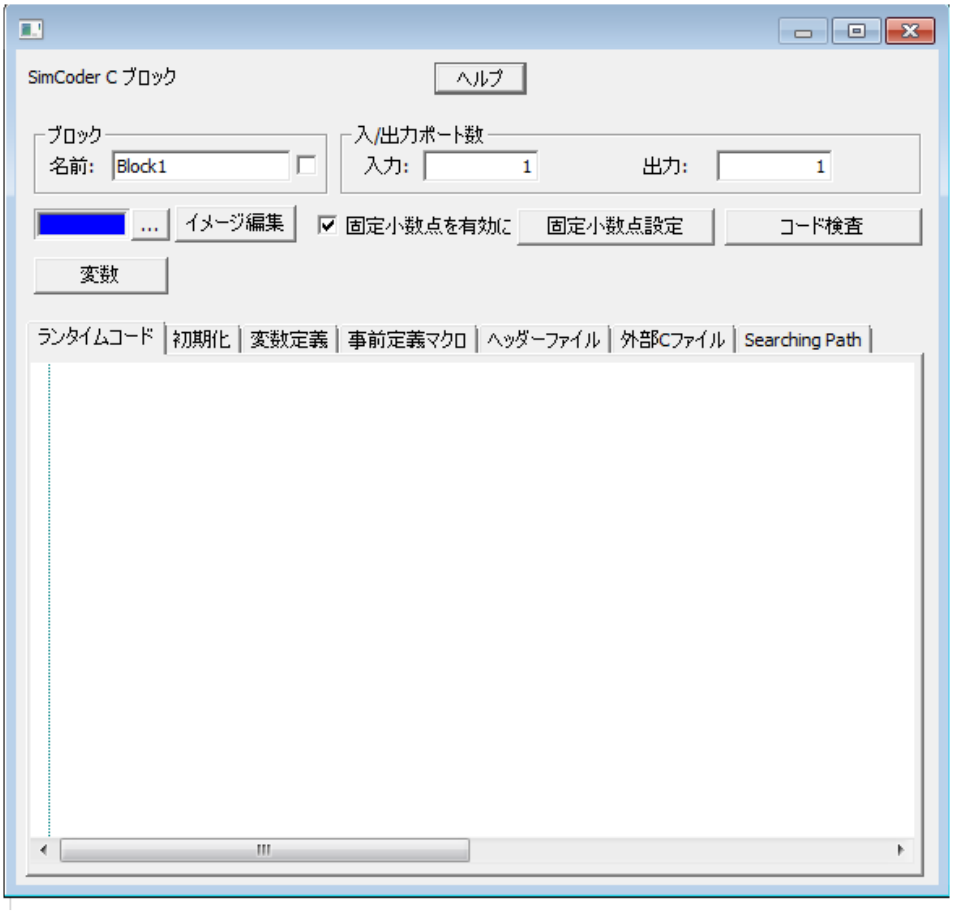


図5.9 SimCoder Cブロック

タブの内容は次のようになります。

タブメニュー	機能
ランタイムコード	毎回実行されるメインコード
初期化	初期化のために最初に1回だけ実行されるコード
変数定義	グローバル変数の定義
事前定義マクロ	外部ヘッダーファイル又はCファイルの前に定義され、コードを前処理するマクロです。 例えば、IQmathライブラリでは、ライブラリが浮動小数点（MATH_TYPE = 1）か固定小数点（MATH_TYPE = 0）かを判断するために"MATH_TYPE"というマクロを使用します。 マクロ "#define MATH_TYPE 1"をこのタブで定義することができます。
ヘッダーファイル	
外部Cファイル	外部ヘッダーファイル.h 外部Cソースファイル

ダイアログWindowに含まれる他の機能は次のようになります。

機能	説明
イメージ編集	Cブロックのイメージを編集します。
固定小数点を有効に	このチェックボックスにチェックを入れると固定小数点のブロックになります。そうでない場合は浮動小数点ブロックとなります。
固定小数点設定	固定小数点ブロックの時にCブロックの入出力のデータフォーマットを定義します。
変数マクロ	メイン回路からCコードへ渡す変数を定義します。変数値としては数値または数式を使用できます。これらの数式にはパラメータファイルで定義された変数やコマンドラインから回路図へ渡された変数を含むことができます。数式はシミュレーション開始前に評価され、t(時間)またはこのブロックの入出力値を含むことはできません。 例： a=15.2 b=Freq/sqrt(2) 'Freq'はパラメータファイルで定義されています。これらの変数はSimCoderでも使えます。SimCoderで生成されたコードには変数の計算式ではなくその評価値を使います。
コード検査	コードの構文検査をします。

SimCoderプリプロセッサーはシンプルコードを生成するためにSimCoderCブロックで使われます。C言語のプロセッサーと区別するために‘##’を使います。次のプリプロセッサーをサポートしています。

##if *condition*

##elif *condition*

##else

##endif

“範例/SimCoder”フォルダのそれぞれのターゲットフォルダにSimCoderCブロックの使い方の例があります。

6 IQmath Library

6.1 概要

TIのIQmath Libraryが浮動小数点コードを固定小数点コードとして接続できるように関数リストとして準備されています。これにより処理速度が上がり精度も高くなります。

IQmath Libraryの詳細につきましては関連するTI資料を参照してください。

6.2 IQmath データタイプ、レンジ及び分解能

ITmath 機能の入出力は 32 ビットの固定小数点型です。

データのタイプは_iq で_iq1 から_iq30、GLOABLE_Q フォーマットは IQ1 から IQ30 のフォーマットとなります。データのレンジと分解能一覧は次の表のようになります。

表 6－ 1 IQmath データタイプ、レンジ及び分解能一覧

データタイプ	レンジ		分解能/精度
	Minimum	Maximum	
_iq30	-2	1.999 999 999	0.000 000 001
_iq29	-4	3.999 999 998	0.000 000 002
_iq28	-8	7.999 999 996	0.000 000 004
_iq27	-16	15.999 999 993	0.000 000 007
_iq26	-32	31.999 999 985	0.000 000 015
_iq25	-64	63.999 999 970	0.000 000 030
_iq24	-128	127.999 999 940	0.000 000 060
_iq23	-256	255.999 999 981	0.000 000 119
_iq22	-512	511.999 999 762	0.000 000 238
_iq21	-1024	1023.999 999 523	0.000 000 477
_iq20	-2048	2047.999 999 046	0.000 000 954
_iq19	-4096	4095.999 998 093	0.000 001 907
_iq18	-8192	8191.999 996 185	0.000 003 815
_iq17	-16384	16383.999 992 371	0.000 007 629
_iq16	-32768	32767.999 984 741	0.000 015 259
_iq15	-65536	65535.999 969 482	0.000 030 518
_iq14	-131072	131071.999 938 965	0.000 061 035
_iq13	-262144	262143.999 877 930	0.000 122 070
_iq12	-524288	524287.999 755 859	0.000 244 141
_iq11	-1048576	1048575.999 511 719	0.000 488 281
_iq10	-2097152	2097151.999 023 437	0.000 976 563
_iq9	-4194304	4194303.998 046 875	0.001 953 125
_iq8	-8388608	8388607.996 093 750	0.003 906 250
_iq7	-16777216	16777215.992 187 500	0.007 812 500
_iq6	-33554432	33554431.984 375 000	0.015 625 000
_iq5	-67108864	67108863.968 750 000	0.031 250 000
_iq4	-134217728	134217727.937 500 000	0.062 500 000
_iq3	-268435456	268435455.875 000 000	0.125 000 000
_iq2	-536870912	536870911.750 000 000	0.250 000 000
_iq1	-1073741824	1 073741823.500 000 000	0.500 000 000

7 TI F2833x Hardware Target

7.1 概要

オプションモジュールTI F2833x 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 F2833xシリーズを搭載した基板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。

TI F2833xハードウェアターゲットはF2833x全てのパッケージに対応しています。次の2ページにわたる図はLQFPのF2833xDSPのピンアサインです。TI F2833xHardware 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 F2833x Hardware Targetには下記の機能があります。

- 3-phase、2-phase、1-phase、とAPWM(Single PWM)生成機能
- 可変周波数PWM
- 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
- デジタル入力出力機能
- SCI基板設定機能、入力出力
- SPI基板設定機能、デバイス、入力出力
- CAN設定機能、入力出力
- キャプチャ、キャプチャステート機能
- エンコーダ、エンコーダステート機能
- Up/Down カウンタ機能
- 割込み時間機能
- 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。

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、SimCoder は PWM 生成ブロックに繋がっていている PWM 生成ブロックと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、生成されるコード内の割り込みサービスルーチンタスク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、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 F2833x 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。

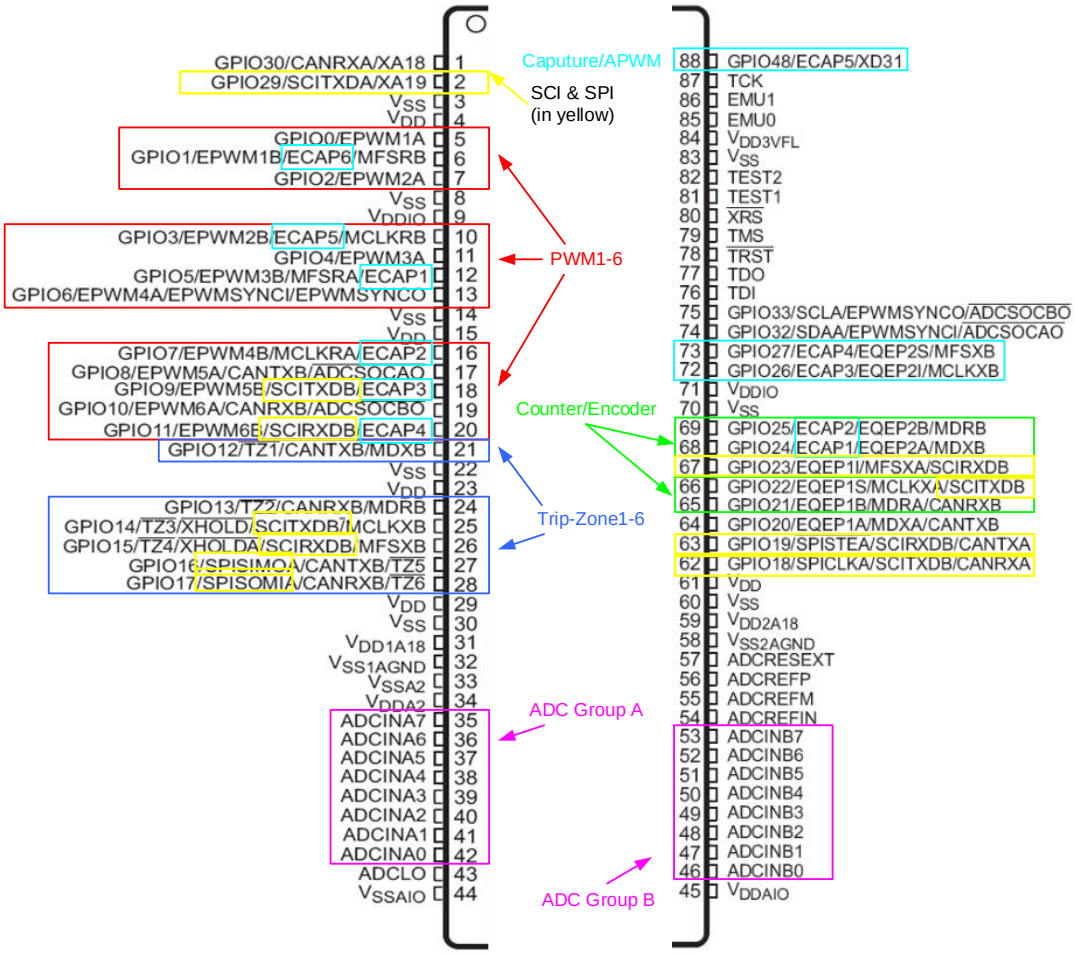


図 7-1　F28335 DSP ポートアサイン（Pin 1-88）

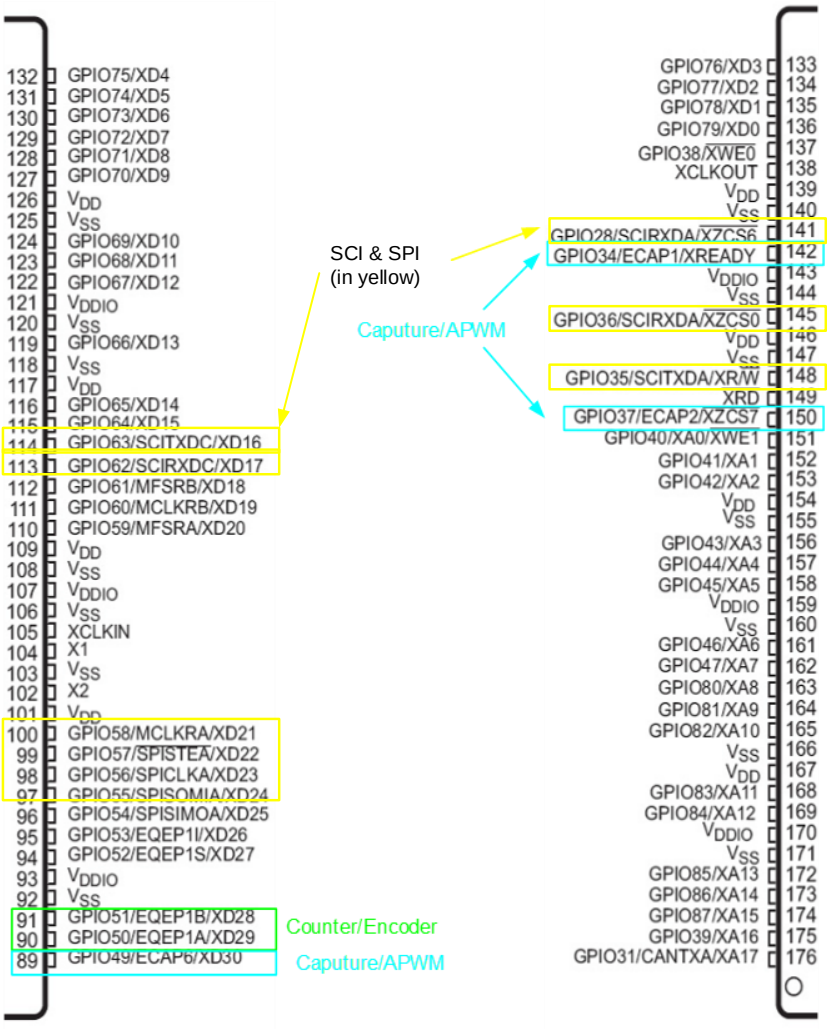
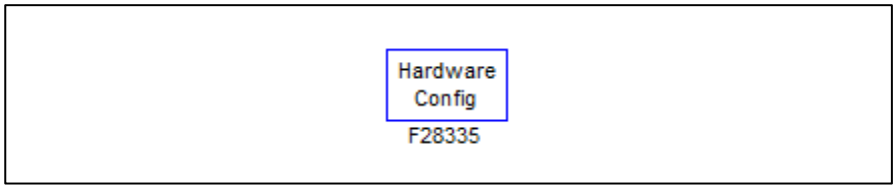


図 7-2　F28335 DSP ポートアサイン（Pin 89-176）

7.2　ハードウェア構成

F2833x には 88 の GPIO ポート（GPIO0～GPIO87）があります。各ポートは違う機能をもっています。しかし、特定の DSP ボード用ではすべてのポートが外部からのアクセスが可能なわけではなくまたいくつかは機能が固定されています。ハードウェアコンフィギュレーションブロックを使うと DSP ボードに SimCoder を設定できます。

シンボル



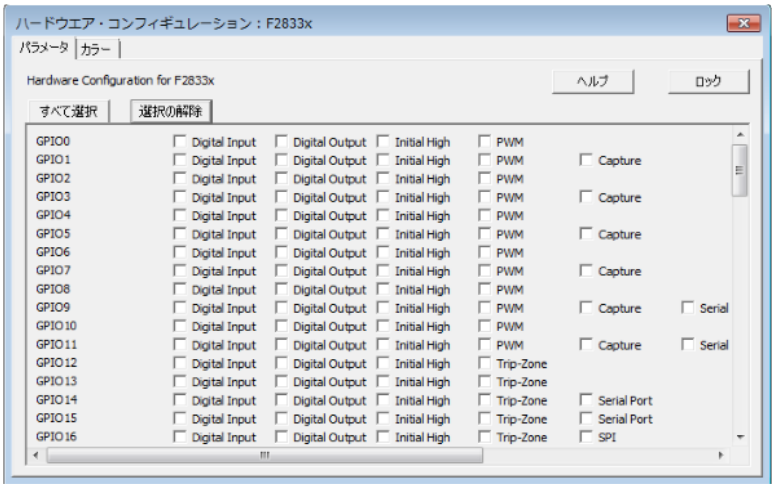


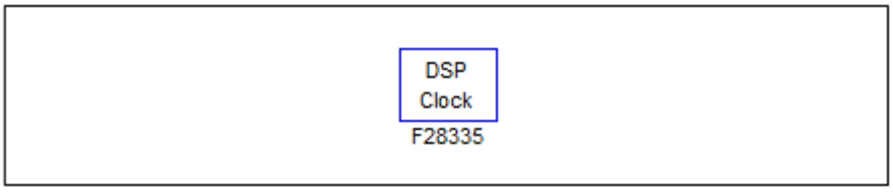
図 7.3 ハードウェアコンフィギュレーションブロックの設定

各々の GPIO ポートのチェックボックスにある機能が使用可能です。SimCoder ではこのボックスがチェックされている機能だけが使用できます。例えばポート GPIO1 では“Digital Input”，“Digital Output”，“PWM”、“Capture”が使用できます。ポート GPIO1 の“PWM”のみをチェックしているにもかかわらず回路のポート GPIO1 で“Digital Input”として使用すると SimCoder はエラーとなります。

7.3 DSP クロック

DSP クロックブロックは外部クロック周波数と F2833x DSP 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
外部 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 速度 (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。

DSP クロックブロックが回路中で使用されていない場合は DSP ブロックのデフォルト値が使用されます。

7.4 PWM 生成器

F2833x 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 互いに相補的な二つの出力をもっています。
4. 位相シフト付 1-phase PWM 生成器：PWM2～6 は互いに相補的な 2 つの出力をもっています。

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 APWM 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ー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。

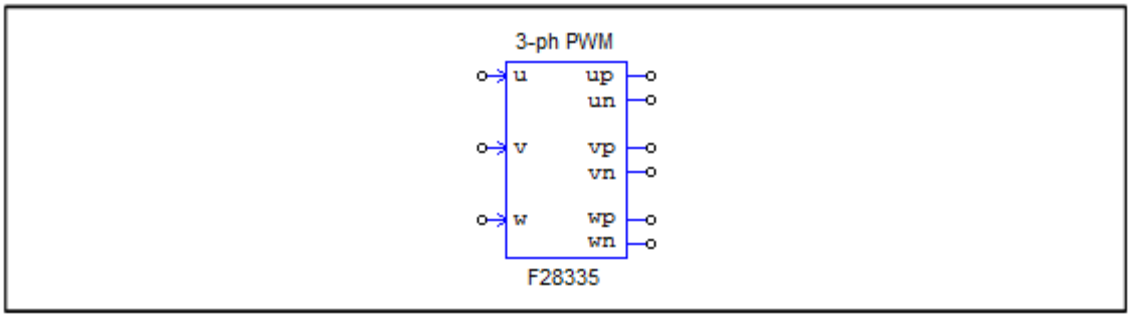
PWM 周波数倍率は F2833x の制限では 1－3 です。(設定は 1 から 100 までできます。3 より大きくしたい場合は、PSIM が未使用の PWM を使って、制御周波数で割り込みを引き起こします。この未使用の PWM は周期的な割り込みを生成するためにしか使用されません、PWM の出力端子は、他の機能に設定することができます。もしシステム内に未使用 PWM がない場合、CPU タイマーが使用されます。

7.4.1 3-phase PWM 生成器：

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。

シンボル:



仕様:

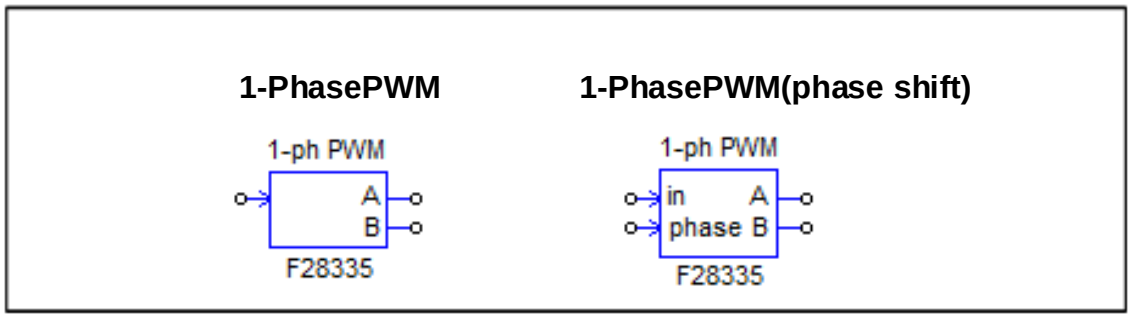
パラメータ	機能
PWM ソース	PWM生成器の出力ピン設定。「3-phase PWM 123」（PWM1~3）か「3-phase PWM 456」（PWM4~6）のどちらかを選択します。
デッドタイム	PWM生成器のデッドタイム（Td）の設定。単位は[sec]
サンプリング周波数	PWM生成器のサンプリング周波数の設定。単位は[Hz] PWM信号のデューティサイクルはここで設定した周波数で更新されます。
PWM周波数倍率	PWM周波数とサンプリング周波数の倍率の設定。 1、2、3の設定ができます。 PWM周波数（制御スイッチで生成されるPWM出力信号の周波数）がサンプリング周波数の何倍かを設定します。 例）サンプリング周波数が50kHzで本設定値が3の場合はPWM周波数が150kHzになります。150kHzのスイッチングでは3回につき1回値の更新を行います。
キャリア波形タイプ	キャリア波形タイプの設定。 - 「Triangular(start low)」：キャリアを三角波、PWM出力の初期値はLow。 - 「Triangular(start high)」：キャリアを三角波、PWM出力の初期値はHIGH。 - 「Sawtooth(start low)」：キャリアをのこぎり波、PWM出力の初期値はLow。 - 「Sawtooth(start high)」：キャリアをのこぎり波、PWM出力の初期値はHIGH。
A/D変換トリガ	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」：A/D変換器のGroup Aへのトリガを有効にします。 - 「Trigger ADC Group B」：A/D変換器のGroup Bへのトリガを有効にします。 - 「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。
A/D変換トリガ位置	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。
トリップゾーンi使用	PWM生成器と繋がる6つの各トリップゾーンの設定。 - 「Disable Trip-Zone i」：対応するトリップゾーン信号を無効にします。 - 「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。 - 「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。
トリップ アクション	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。
ピーク間電圧	キャリア波のピーク間電圧Vppの設定。
オフセット電圧	キャリア波のオフセット電圧Voffsetの設定。
初期入力値u, v, w	u、v、w三相入力ノードの初期値の設定。
最初からPWM 信号出力	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。
シミュレーション 出力モード	シミュレーションの出力モードはSwitchingモードかAverageモードに設定されます。“Switchingモード”へ設定された時にはPWMブロックの出力はPWM信号となります。“Averageモード”へ設定された時にはPWMブロックの出力は平均モード信号となります 例えばPWMブロックの入力をVuとすると出力のupとunは Vup=Vu/(Vpk_pk+Voffset) Vun=－Vu/(Vpk_pk+Voffset) となります。Average mode に設定されるとPWMブロックの出力はaverage mode modelのインバータへ接続されます。

7.4.2 1-phase PWM 生成器（外部位相シフト機能付きを含む）：

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。

シンボル:



仕様:

パラメータ	機能
PWM ソース	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。
出力モード	PWM生成器の出力モードの設定。 - 「Use PWM A&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。 - 「Use PWM A」：PWM出力Aのみ使用します。 - 「Use PWM B」：PWM出力Bのみ使用します。
デッドタイム	PWM生成器のデッドタイム（Td）の設定。単位は[sec]
サンプリング周波数	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 周波数倍率	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3の場合はPWM周波数が150kHzになります。150kHzのスイッチングでは3回につき1回値の更新を行います。
キャリア波形タイプ	キャリア波形タイプとPWM出力信号の初期状態の設定。 - 「Triangular(start low)」：キャリアを三角波、PWM出力の初期値はLow。 - 「Triangular(start high)」：キャリアを三角波、PWM出力の初期値はHIGH。 - 「Sawtooth(start low)」：キャリアをのこぎり波、PWM出力の初期値はLow。 - 「Sawtooth(start high)」：キャリアをのこぎり波、PWM出力の初期値はHIGH。
A/D変換トリガ	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」：A/D変換器のGroup Aへのトリガを有効にします。 - 「Trigger ADC Group B」：A/D変換器のGroup Bへのトリガを有効にします。 - 「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。
A/D変換トリガ位置	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。
トリップゾーンｉ使用	PWM生成器と繋がる6つの各トリップゾーンの設定。 - 「Disable Trip-Zone i」：対応するトリップゾーン信号を無効にします。 - 「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。 - 「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。
トリップアクション	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。
ピーク間電圧	キャリア波のピーク間電圧Vppの設定。

オフセット電圧	キャリア波のオフセット電圧Voffsetの設定。
位相シフト	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)
初期入力値	入力ノードの初期値の設定。
Use HRPWM	高分解能PWMの定義を行います。 *Do not use HRPWM*:高分解能PWMを使用しません。 *UseHRPWM*(校正なし)：校正なしで高分解能PWMを使用します。 *UseHRPWM*(校正あり)：校正ありで高分解能PWMを使用します。
最初からPWM信号	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。
シミュレーション出力モード	シミュレーションの出力モードはSwitchingモードかAverageモードに設定されます。“Switchingモード”へ設定された時にはPWMブロックの出力はPWM信号となります。“Averageモード”へ設定された時にはPWMブロックの出力は平均モード信号となります 例えばPWMブロックの入力をVuとすると出力のupとunは Vup=Vu/(Vpk_pk+Voffset) Vun=－Vu/(Vpk_pk+Voffset) となります。Average mode に設定されるとPWMブロックの出力はaverage mode modelのインバータへ接続されます。

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°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°右に（遅れ）にシフトされます。これは、30°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°の場合、出力は 30°左（進み）にシフトされます。

搬送波

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。

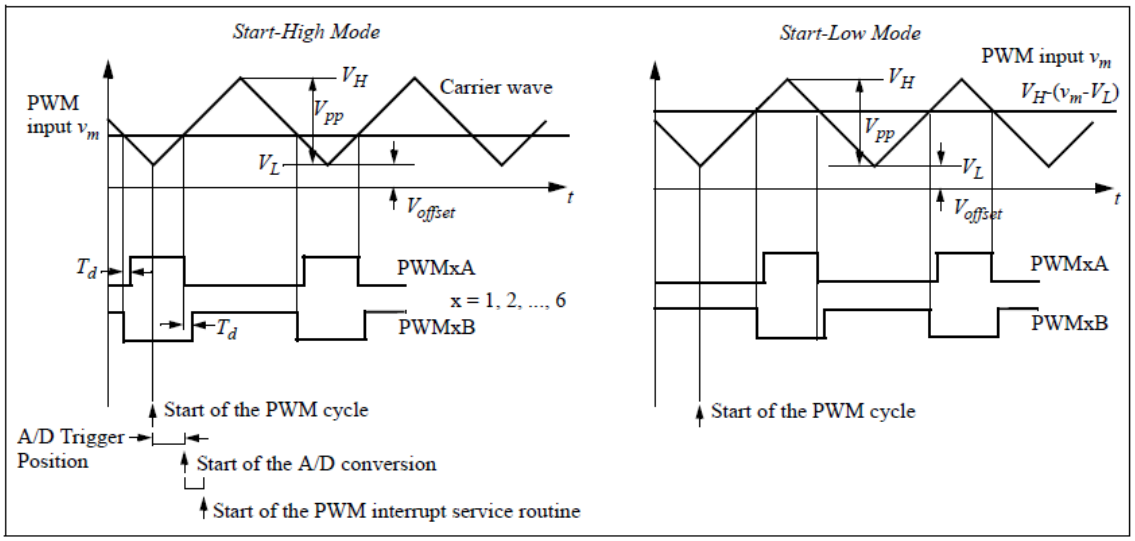


図 7.1 三角搬送波の PWM 生成器の入力波形と出力波形

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。

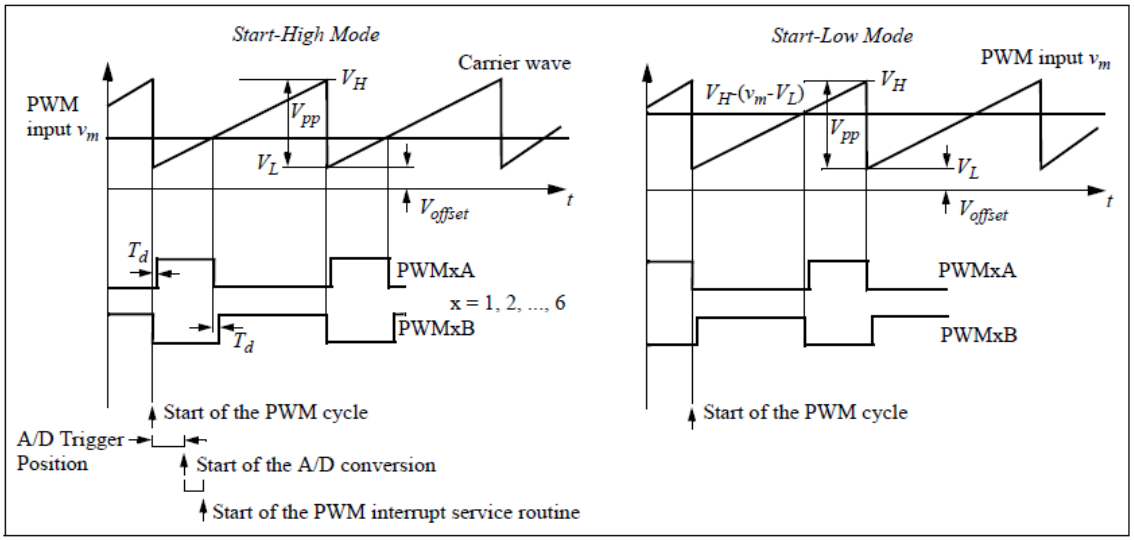


図 7.2 ノコギリ搬送波の PWM 生成器の入力波形と出力波形

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。

キャリア開始モードはスイッチ電流の測定方法によりかわります。3 相インバータの場合はスイッチ電流が測定され、スタートハイモードが選択されます。これは上のスイッチが周期の最初にトップスイッチゲーティング信号が高く、電流が流れていることになります。

一方低いスイッチ電流が測定されているとスタートローモードが選択されます。これは周期の最初に

トップスイッチゲーティング信号が低く、下のスイッチゲーティング信号が高く、電流が流れていることになります。

次のような回路の場合、位相 A と B の下のスイッチ電流が測定されます。この場合はキャリアスタートローモードを選択すべきです。

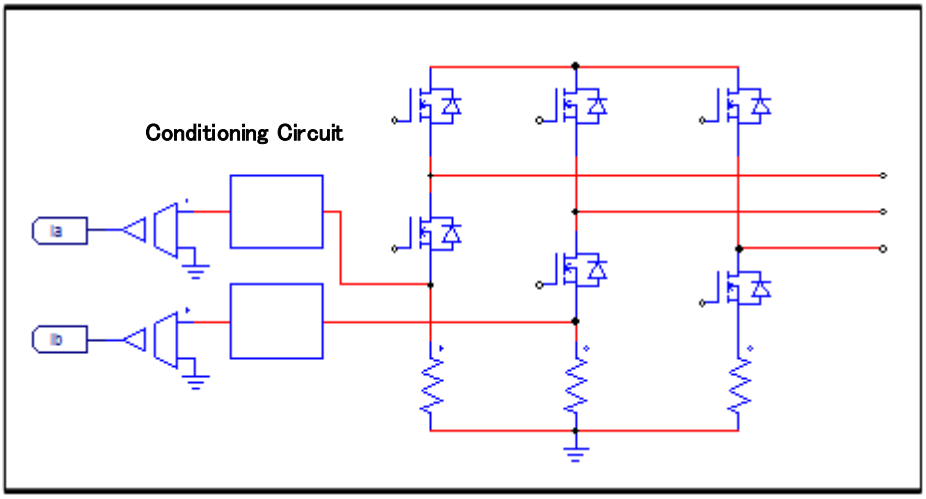


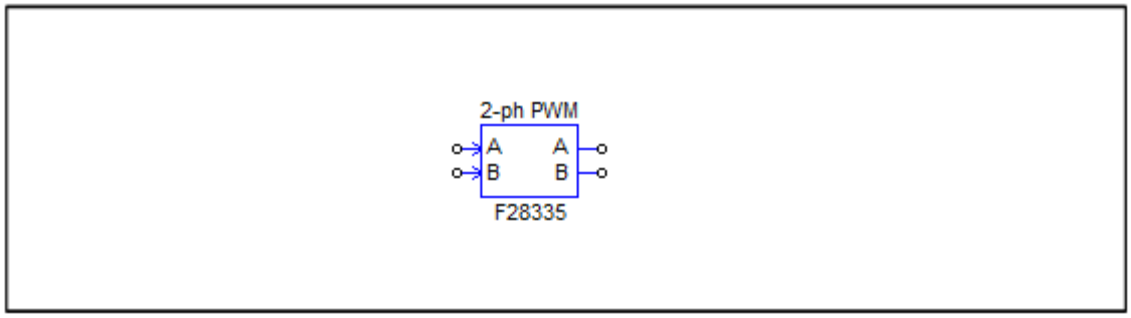
図 7.3 下のスイッチ電流が測定される場合の回路

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ー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 出力のデューティー比 D は”Start-Low”と”Start-High”の両方で「D=Vm/VH」となります。

7.4.3　2-phase PWM 生成器：

2 層 PWM ブロックは 6 つの動作モードがあります。

シンボル:



仕様:

パラメータ	機能
PWM ソース	PWM生成器の出力ピン設定。 PWM1~6までを選択します。
出力モード	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。
サンプリング　周波数	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 周波数倍率	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ーの設定。 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。
A/D変換トリガ	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」：A/D変換器のGroup Aへのトリガを有効にします。 - 「Trigger ADC Group B」：A/D変換器のGroup Bへのトリガを有効にします。 - 「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。
A/D変換トリガ　位置	ADCチャンネルのPWMモジュールによるトリガされた変換位置。トリガされた変換位置は、搬送波の開始時に、または搬送波の途中のいずれかに設定できます。　途中トリガ位置は、動作モード4,5、及び6の場合に有効です。 もしPWMがADCをトリガーしない場合は、この位置はPWM割り込みの位置になります。
トリップゾーンi 使用	PWM生成器と繋がる6つの各トリップゾーンの設定。 -「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 -「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

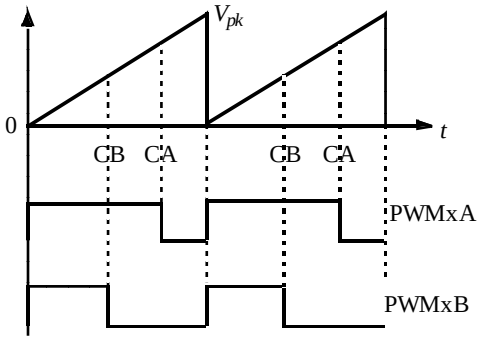
	に再出力されるようになります。
トリップアクション	トリップゾーン信号に対するPWM生成器の動作の設定。 - 「High Impedance」：PWM出力部はハイインピーダンスになります。 - 「PWM A high & PWM B low」：PWMの正の出力はハイレベル、負の出力はローレベルになります。 - 「PWM A low & PWM B high」：PWMの正の出力はローレベル、負の出力はハイレベルになります。 - 「No action」：何も動作を設定しません。
ピーク間電圧	キャリア波のピーク電圧Vpkの設定。
初期入力値A, B	A、B入力ノードの初期値の設定。
最初からPWM信号	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。

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 Vpk 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-pahse 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。

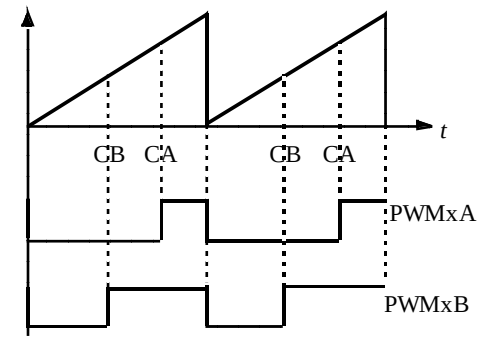
Mode 1:



動作モード２：

動作モード 1 と違い、各出力のターンオンタイミングを決めるモードです。

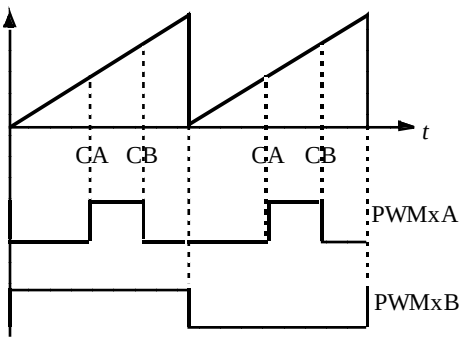
Mode 2:



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。

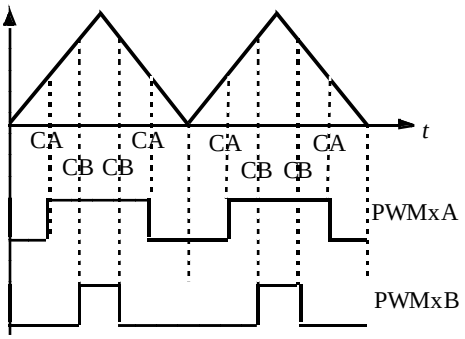
Mode 3:



動作モード 4：

キャリアは三角波になります。各入力は対応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。

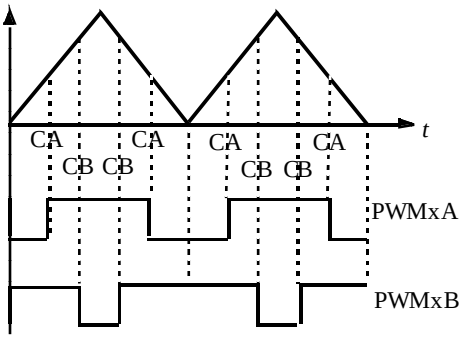
Mode 4:



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。

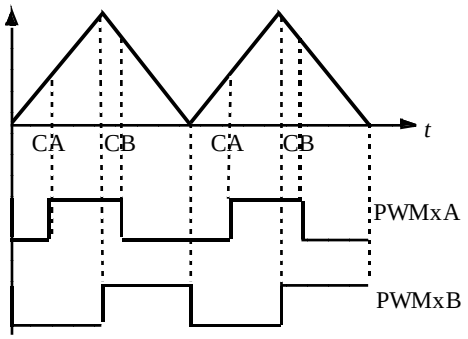
Mode 5:



動作モード 6：

入力 A は PWM A のターンオンのタイミングを決め、入力 B は PWM A 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。

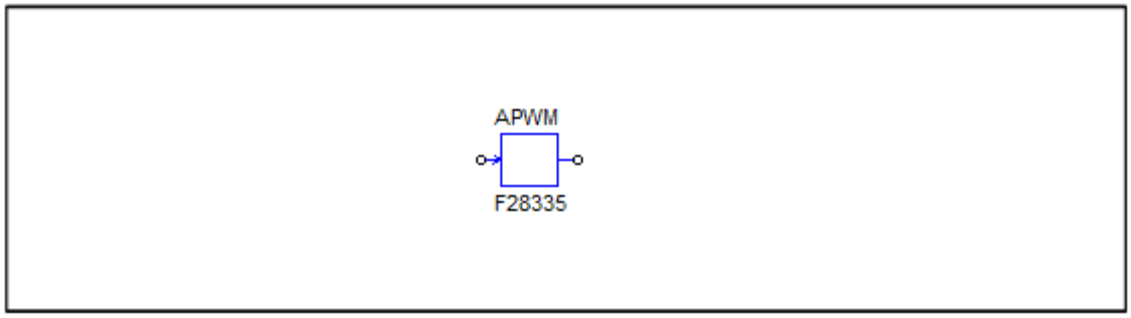
Mode 6:



7.4.4 Single PWM 生成器(APWM)：

SinglePWM 生成器は APWM とも言われキャプチャと同じリソースを共有しています。

シンボル:



仕様:

パラメータ	機能
PWM ソース	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。 - APWM1(GPIO5, GPIO24, GPIO34) - APWM2(GPIO7, GPIO25, GPIO37) - APWM3(GPIO9, GPIO26) - APWM4(GPIO11, GPIO27) - APWM5(GPIO3, GPIO48) - APWM6(GPIO1, GPIO49)
PWM 周波数	PWM生成器の周波数の設定。単位は[Hz]
キャリア波形タイプ	キャリア波形タイプの設定。 次のいずれかになります。: - *のこぎり波 (低でスタート)*: キャリアをのこぎり波、PWM出力の初期値はLOW。 - *のこぎり波 (高でスタート)*: キャリアをのこぎり波、PWM出力の初期値はHIGH。
ストップ出力	PWMジェネレータが停止している時の出力設定。

	- 出力　低: PWM出力は低に設定されます。 - 出力　高: PWM出力は高に設定されます。
ピーク間電圧	キャリア波のピーク間電圧の設定。
オフセット値	キャリア信号のオフセット電圧の設定。
位相シフト	PWM生成器の出力に対しての位相シフト値（単位deg）
初期入力値	入力の初期値の設定。
最初からPWM信号	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。

1相 PWM 生成器と同様に APWM 生成器は他の PWM 生成器に関して位相シフトをもつ PWM 信号を生成します。位相シフトのルールは 1相 PWM 生成器と同じです。

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。

7.4.5 PWM ブロック間の同期

PWM の 3 つのタイプで同期が可能です。位相シフトはお互いの間で定義できます。単相 PWM、単相 PWM(位相シフト付)と APWM(もしくは単層 PWM(キャプチャと共用)です。

単相 PWM ブロックは別の PWM 信号で位相シフトした PWM 信号を生成できます。通常の PWM ブロックには PWM1、2、3 と PWM1、4、5、6 の 2 つのシリーズがあります。

同様に APWM ブロックには位相シフトのための 2 つのシリーズがあります。それは PWM1、APWM1、2、3 と PWM1、APWM4、5、6 です。

位相シフトのための PWM ブロックの定義は次のようになります。

－基準 PWM と位相シフトされる PWM は同じシリーズでなければなりません。すなわち PWM1 が基準、PWM2、3 か PWM4、5、6 が PWM1 に関して位相シフトできます。または、PWM2 が基準となる場合は PWM3 が PWM2 について位相シフトできます。

同様に PWM4 (または 5)は基準となり、PWM5 (または 6)は PWM4(または 5)に関して位相シフトできます。しかし、PWM4、5、6 に対して PWM2 または 3 を基準として使うことはできません。

これはまた APWM についても同様です。例えば PWM1 が基準となる場合、APWM1、2、3 は位相シフトされます。同様に PWM1 が基準の場合、APWM4、5、6 が位相シフトできます。または、APWM4 が基準の場合、APWM5 と 6 は位相シフトできます。

－基準 PWM と位相シフトされる PWM は省略された PWM が使用されないようにシリーズの中で連続でなければなりません。例えばもし PWM1、2、3 すべてが使用されているが PWM2 が PWM1 と 3 と同期していない場合は使用できません。しかしもし PWM2 が回路中で使用されていない場合は PWM1 を基準、位相シフト PWM3 として使用することができます。

例えば正しい設定は次のようになります。最初が基準でその後のブロックが位相シフトとなります。

PWM 1 (基準), PWM 2, PWM 3

PWM 1 (基準), PWM 4, PWM 5, PWM 6

PWM 2 (基準), PWM 3

PWM 4 (基準), PWM 5

PWM 5 (基準), PWM 6

PWM 1 (基準), APWM 1, APWM 2, APWM 3

PWM 1 (基準), APWM 4, APWM 5, APWM 6

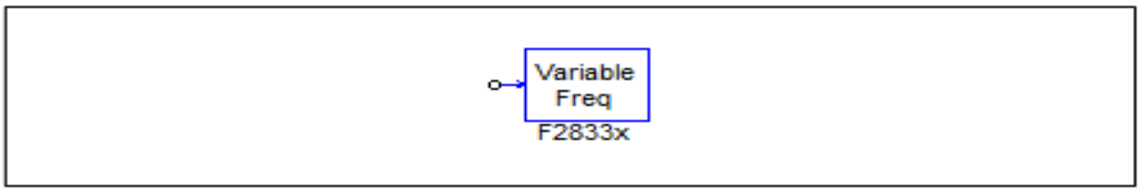
APWM 1 (基準), APWM2, APWM 3

APWM 4 (基準), APWM5, APWM 6

7.5 可変周波数 PWM

可変周波数 PWM ブロックでは PWM 生成器のサンプリング周波数の変更ができます。

シンボル：



仕様

パラメータ	機能
PWM ソース 割り込みポジション調整	PWM 生成器のソース PWW1～6、3 相 PWM123 と PWM456 を選択できます。 PWM 周波数が調整する時、割込み位置を調整するかどうかを指定します。 次のいずれかになります: - *調整しない*: 割込み位置はベース周波数で計算された初期位置に維持します。 - *調整する*: 割り込み位置は、新しい周波数に基づき再計算されます、次の PWM 周期で適用されます。

PWM ブロックと一致するサンプリング周波数は次のような PWM 周期の最初に変更されます。

PWM_Frequency=PWM_Base Frequency／Input_Value

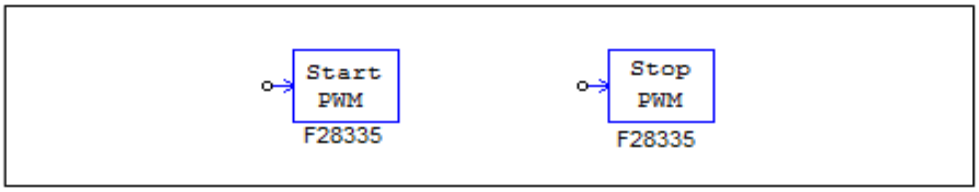
ここで *PWM_Base_Frequency* は PWM_block に一致したサンプリング周波数です。*Input_Value* はこのブロックの入力値です。

割り込み位置が調整されると各サイクルで再計算されます。割込み位置調整は時間がかかるので周波数変更が小さい場合はやらないのがお薦めです。

7.6 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 ソース	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。

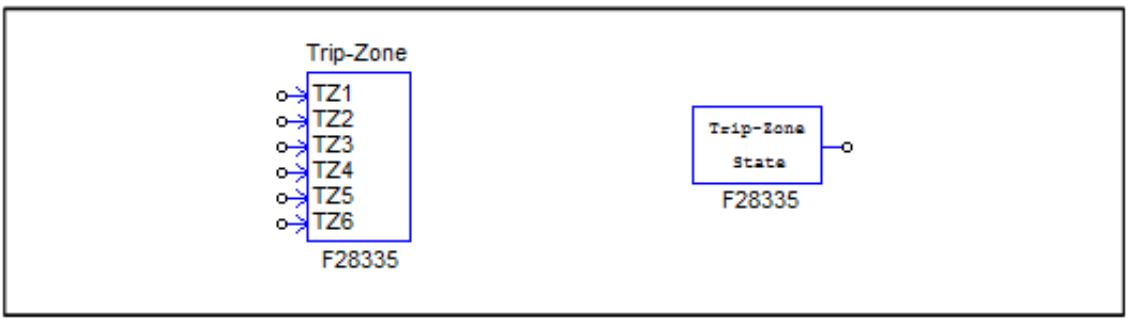
7.7 トリップゾーンとトリップゾーンステート

F2833x 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 ソース	PWM生成器の出力の選択。 - PWM1~6、3-phase PWM123と3-phase PWM456、そしてPWM1~6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。）

7.8 A/D 変換器

F2833x 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 への回路的な入力の範囲は 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、ある主回路電圧値は、最初に電圧センサで制御信号に変換されます。その後、回路信号値を 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）

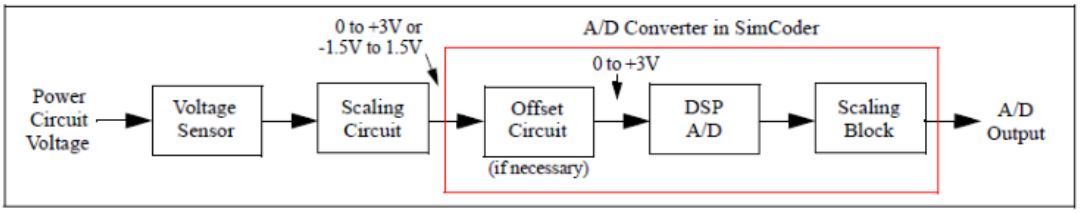


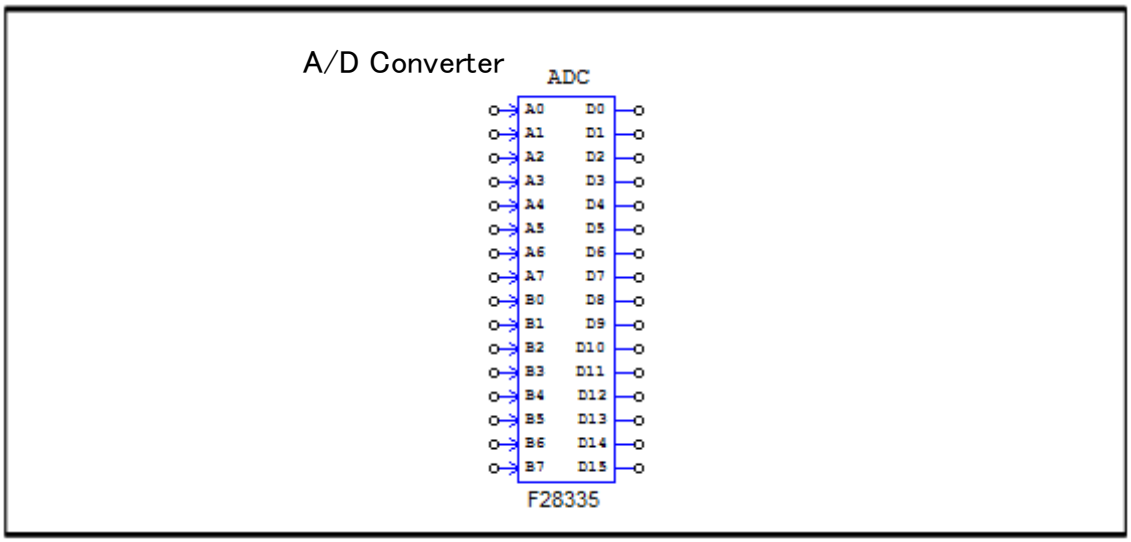
図 7.4 主回路から AD 出力までの処理

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。

これは AC システムアプリケーションが便利に使えるようデザインされています。

※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。 - 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。 - 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。 - 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。
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。 - AC：このオプションはシミュレーションのためのみです。-1.5Vから+1.5Vの入力が可能なac入力になります。このオプションはA/D変換へのオフセット回路を含んでいます。AC信号を0から3Vとする外部シフタが必要な場合に便利です。 - DC：0Vから+3Vの入力が可能なdc入力になります。
Ch Ai or Bi Gain	各A/D変換チャンネルのゲインk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。

操作モード：

A/D 変換は Continuous モードに設定すると自律的に実行され、Start/Stop モードの場合は PWM 生成器からのトリガで実行されます。PWM を使って A/D 変換のトリガ機能をかける場合は次の制約があります。

- A/D 変換はＰWM生成器のみによりトリガがかかります。すなわち複数のＰWMがある場合は１つのＰWM生成器を使って A/D 変換器にトリガをかけ、ｓ

A/D 変換がトリガをかけることはできません。このグループの信号のいくつかが回路で使用されている時はできません。

これらの状態の場合は A/D 変換器は“Continuous”モードに設定しておいてください。

出力スケーリング：

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。

入力オフセットとスケーリング：

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。

多くのアプリケーションではモニターされる回路変数は、特に AC モータードライブシステムの場合 AC 信号となります。このような AC 信号ではそれぞれオフセット回路がハードウェアの回路ボード上で 0 から 3V へ信号レベルをひき上げるためにオフセット回路が組み込まれないとなりません。

F2833xDSP の SimCoder の A/D 変換器はこのような場合に便利です。A/D 出力信号のレベルシフティングやスケーリングの代わりにオフセットポイントや SimCoder　の A/D 変換器のスケーリングファクターを選択して使用できそれに従ってターゲットコードを生成できます。

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。

A/D 変換器チャネル 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 0.01×2=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 × 2 = 100V

回路図は以下になります。

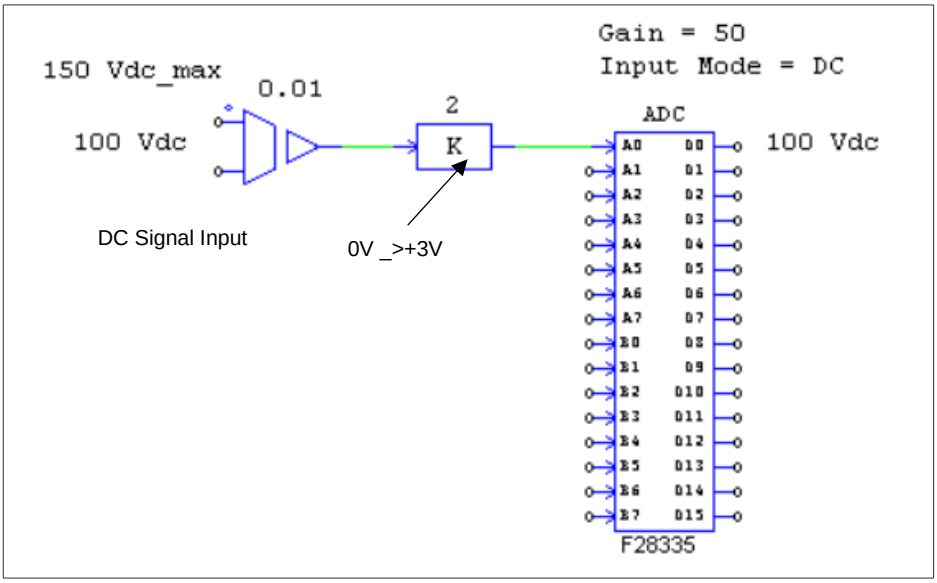


図 7.5 DC 入力場合の A/D 変換処理例

※上図では比例制御器ブロックのゲインを 2 から 1、A/D 変換器のゲインを 50 から 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ない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±75V までの値を取るとして、実際に入力値±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±0.75V が来ますが、DSP 入力可能ピーク値は±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} = +/-75V

V_o = +/-50V

回路図は以下になります。

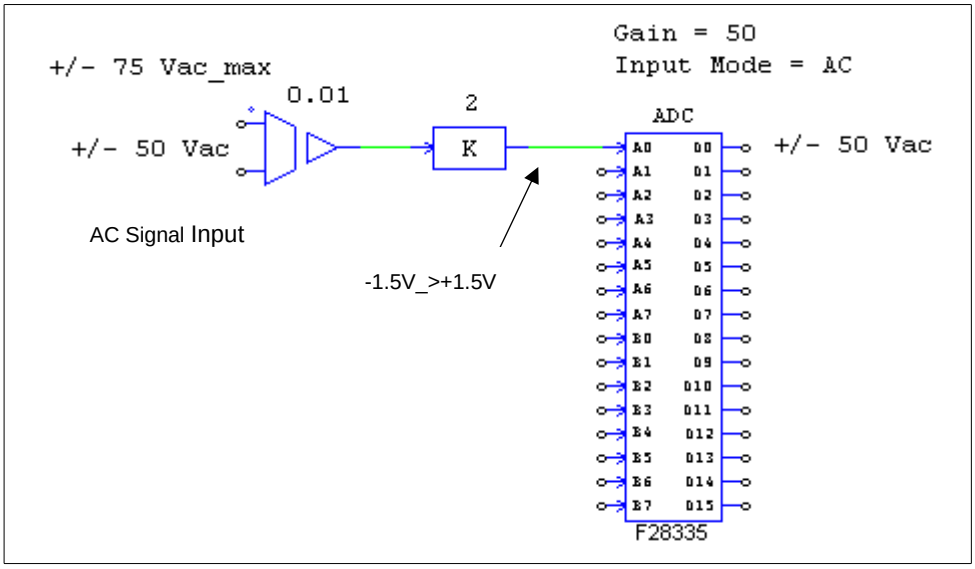


図 7.6 AC 入力場合の A/D 変換処理例

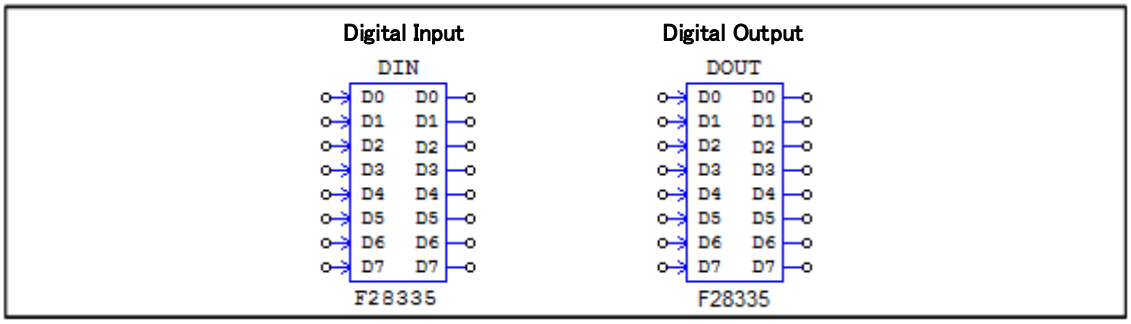
※PSIM の回路では直接 AC 信号を A/D 変換器に入力することができます。これは、ブロックの中にオフセット追加機能が含まれており、自動的に調整を行うためです。実際の DSP は 0V~3V までの入力となり、外部にオフセット回路を追加する必要がある点にご注意ください。

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。

7.9 デジタル入力とデジタル出力

F2833x は 88 本の汎用入出力ポート（GPIO）をデジタル入力、またはデジタル入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
入力ポート i の位置	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。
外部割込みとして使用	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
出力ポート i の位置	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すが、同時に設定した場合、エラーが出力されます。

TMS320F2833x 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。外部割込みの優先順は GPIO0~31 の方が GPIO32~63 より高くなっています。）

7.10 UP/DOWN カウンタ

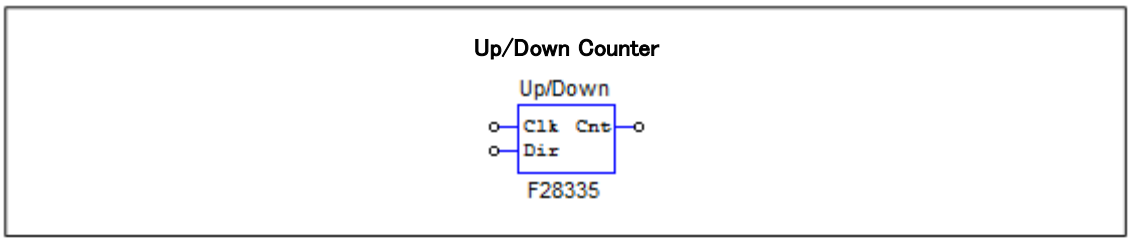
TMS320F28335 は UP/DOWN カウンタを二つ持っており、カウンタ 1 はポート GPIO20,21 または GPIO50,51 で利用でき、カウンタ 2 は GPIO24,25 で利用できます。

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

カウンターソース	カウンタとポートの設定。 - *Counter1 (GPIO20, 21)*：Counter1をGPIO20,21で使用します。 - *Counter1 (GPIO50, 51)*：Counter1をGPIO50,51で使用します。 - *Counter2 (GPIO24, 25)*：Counter2をGPIO24,25で使用します。

※“Clk”はクロック信号入力、“Dir”はカウンタがどちらへ進むかを指定します。“Dir”が 1 の時カウンタは増加し 0 の時は減少します。

※Clk 端子は割り振られた最初の GPIO ポート番号を使います、Dir 端子は二番目の GPIO ポート番号を使います。 例えばカウンター1 を GPIO20,21 に設定した場合、Clk 端子が GPIO20 に、Dir 端子が GPIO21 に割り振られます。

※UP/DOWN カウンタブロックの出力はカウンタ値となります。

※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。

7.11 エンコーダとエンコーダステート

F2833x 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。

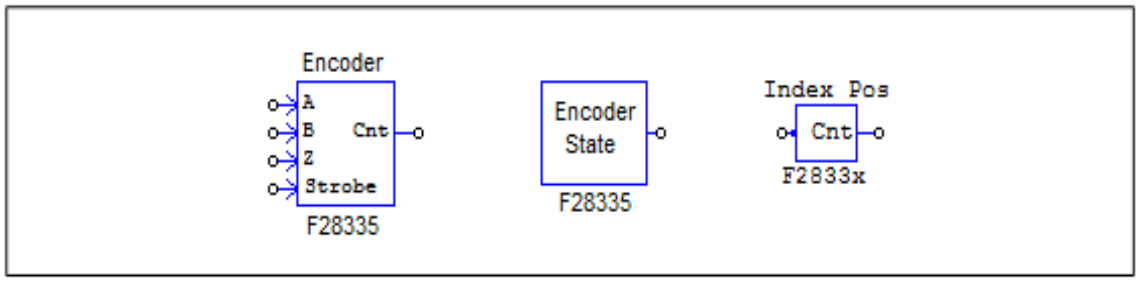
エンコーダステートブロックはどの入力信号（基準信号とストローブ信号）が割り込みを発生さる設定なのかを見ることに使います。

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。

エンコーダインデックス/ストロボ位置ブロックはエンコーダの初期位置をラッチするのに使用されます。この素子の入力が 0 の時、エンコーダカウンタは 0 に設定されます。入力が 1 になるとエンコーダは、インデックス/ストロボイベントの発生を待ち、発生するとエンコーダのカウンタ値をラッチします。

※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
エンコーダ ソース	エンコーダとポートの設定。 - *Encoder1 (GPIO20, 21)*：Encoder1をGPIO20,21で使用します。GPIO22はStrobe、GPIO23はIndex(Z)となります。 - *Encoder1 (GPIO50, 51)*：Encoder1をGPIO50,51で使用します。GPIO52 - はStrobe、GPIO53はIndex(Z)となります。 - *Encoder2 (GPIO24, 25)*：Encoder2をGPIO24,25で使用します。GPIO27はStrobe、GPIO26はIndex(Z)となります。
Z信号使用	Z信号（基準信号）の有効無効設定。 - *No:* 使用しない。 - *Yes(rising edge):* 立上り信号エッジを使用します。 - *Yes(falling edge):* 立下り信号エッジを使用します。
ストローブ信号 使用	ストローブ信号の有効無効設定。 - *No:* 使用しない。 - *Yes(rising edge):* 立ち上がり信号エッジを使用します。 - *Yes(rising/falling edge):*立ち上がり立下り両方の信号エッジを使用します。
カウンタ方向	カウンタの方向の設定。 - *Forward*：カウンタは増加方向に進みます。 - *Reverse*：カウンタは減少方向に進みます。
Z 信号極性	Z信号のトリガの極性 - *Active High* - *Active Low*
ストローブ信号 極性	ストローブのトリガ極性 - *Active High* - *Active Low*
エンコーダ 分解能	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
エンコーダ ソース	状態を観測するエンコーダ番号の設定。 - *Encoder1 (GPIO20, 21)*：GPIO20,21のEncoder1に対して利用します。 - *Encoder1 (GPIO50, 51)*：GPIO50,51のEncoder1に対して利用します。 - *Encoder2 (GPIO24, 25)*：GPIO24,25のEncoder2に対して利用します。

仕様（エンコーダステート）：

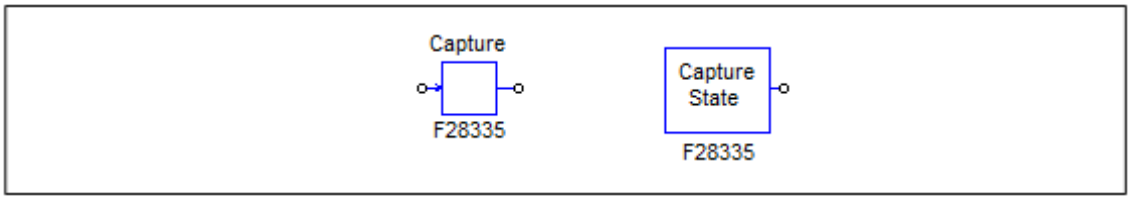
パラメータ	機能
エンコーダ ソース	状態を観測するエンコーダ番号の設定。 - *Encoder1 (GPIO20, 21)*：GPIO20,21のEncoder1に対して利用します。 - *Encoder1 (GPIO50, 51)*：GPIO50,51のEncoder1に対して利用します。 - *Encoder2 (GPIO24, 25)*：GPIO24,25のEncoder2に対して利用します。 Encoder1(GPIO20, 21)とEncoder1 (GPIO50, 51)が同じ内部機能ブロックで使用される場合同時に使うことはできません。
位置タイプ	ラッチカウンタータイプ - Index Pos:Z方向信号が使用されます。 - Strobe Pos:Strobe信号が使用されます。
Positopn Type	ラッチ位置 - 最初のラッチ位置 - 現在のラッチ位置

7.12　キャプチャとキャプチャステート

F2833x 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
キャプチャ ソース	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。 - Capture1 (GPIO5, GPIO 24, GPIO 34) - Capture2 (GPIO7, GPIO 25, GPIO 37) - Capture3 (GPIO9, GPIO 26) - Capture4 (GPIO11, GPIO 27) - Capture5 (GPIO3, GPIO 48) - Capture6 (GPIO1, GPIO 49)
イベント フィルタ	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 モード	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
キャプチャ ソース	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。

キャプチャステートブロック出力は 1 か 0 で、1 は立ち上がり信号エッジ、0 は立ち下がり信号エッジとなります。

7.13 シリアル通信インターフェース（SCI）

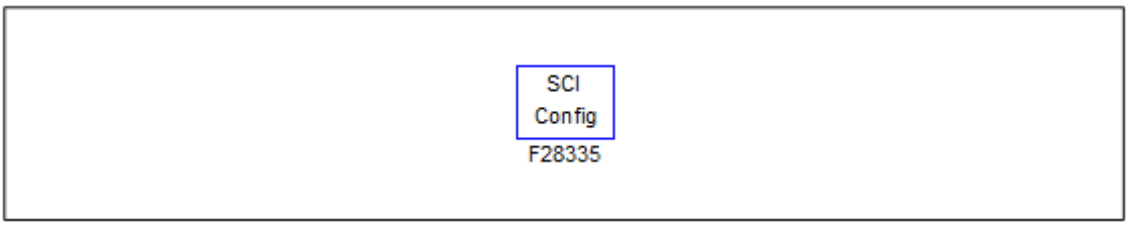
F2833x は、シリアルコミュニケーションインタフェース（SCI）のための関数が用意されています。SCI を介して、DSP 内部のデータは、外部 RS-232 ケーブルを使用してコンピュータに転送することができます。PSIM は、DSP とコンピュータ両方のデータの送受信およびコンピュータ上にデータを表示するための必要なすべての関数を提供します。これは DSP のコードをリアルタイムでモニタ、デバッグ、および調整するために非常に便利な方法を提供します。

以下に説明しますが、SimCoder では SCI 設定、SCI 入力、および SCI 出力の 3 つの SCI の機能ブロックが用意されています：

7.13.1 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 ポート選択	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。 - SCIA (GPIO28, GPIO35)　(GPIO29, GPIO36)との組み合わせ - SCIB (GPIO9, GPIO14, GPIO18, GPIO22)と(GPIO11, GPIO15, GPIO19, GPIO23)との組み合わせ - SCIC (GPIO62, GPIO63)
通信速度(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。
パリティ チェック	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。

出力バッファサイズ	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。

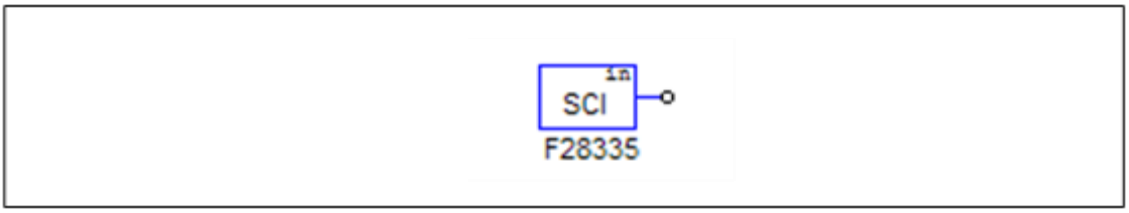
より多くの変数を長い期間にわたって監視することができるように、より多くのデータポイントを収集するためには大きなバッファが好ましい。一方、内部 DSP のメモリは限られており、バッファは通常の DSP の動作を妨げるほど大きすぎてはいけません。バッファサイズを選択する方法の詳細については、"Tutorial - Using SCI for Waveform Monitoring.pdf"を参照してください。

7.13.2 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
初期値	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。

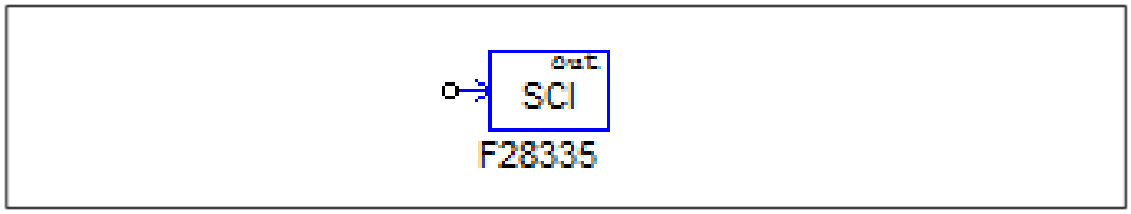
7.13.3 SCI 出力

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
データポイントステップ	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ポイントのうちの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 オシロスコープが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。

7.14 シリアルペリフェラルインターフェース(SPI)

F2833x 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 F833x 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。

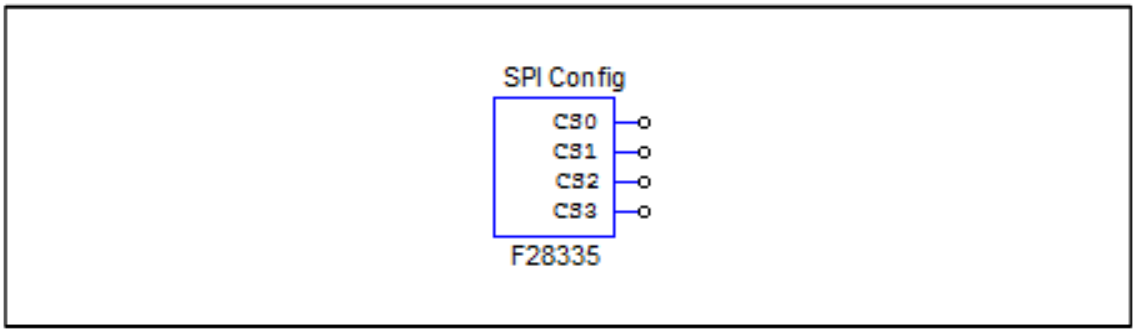
SPI ブロックを使用する方法についての詳細な説明については、ドキュメントを参照してください*" Tutorial - Using SCI for Real-Time Monitoring.pdf"*。

SimCoder では SPI 設定、SPI デバイス、SPI 入力、SPI 出力の 4 つの SCI の機能ブロックが提供されています。

7.14.1 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:



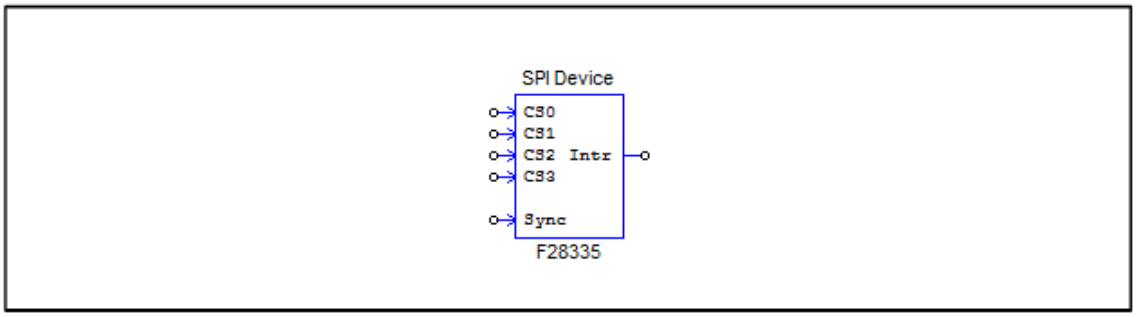
仕様:

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。

7.14.2 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。 -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。 -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。 -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。 -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。
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。 - ハードウェア割り込みなし - 立ち上がりエッジ - 立ち下がりエッジ
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。 - 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 - 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ネルの新しい変換を開始します。 - ワンタイム割り込み： 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。
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ンマで区切られた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。

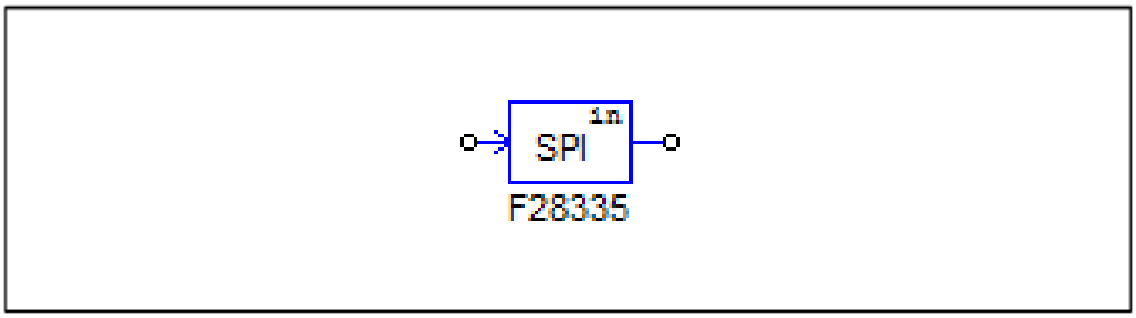
同期信号を使用した SPI の入力デバイスは、通常、割り込みサービスルーチンを入力して DSP をトリガする割り込みピンを必要とします。

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。

7.14.3 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 通信用の入力チャネル、および 1 つの入力チャネルに対応する 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は、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リングされます。
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 $Output = \frac{Result \cdot V_{max} \cdot K_{scale}}{2^{Data\_Length}}$

AC 変換モードの時：

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

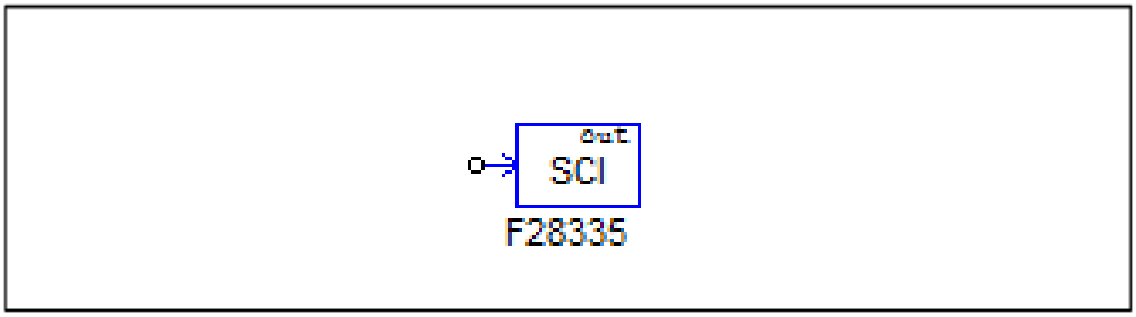
-実際のハードの場合 $Output = \frac{(Result - 2^{Data\_Length-1}) \cdot V_{max} \cdot K_{scale}}{2^{Data\_Length-1}}$

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。

7.14.4 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 通信用の出力チャネル、1 つの出力チャネルに対応する 1 つの SPI 出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リングされます。
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	コンマで区切られた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、 - *"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。 - {}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。 - *MSB.. LSB*は、ワードの有効ビットの位置を定義します。
Sync. Command	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ネルと同期させるコマンドです。(例；0x23、0x43、0x00) 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。

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 $Output = \frac{Result \cdot K_{scale} \cdot 2^{Data\_Length}}{V_{max}}$

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 $Output = 2^{Data\_Length} + \frac{Result \cdot K_{scale} \cdot 2^{Data\_Length-1}}{V_{max}}$

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。

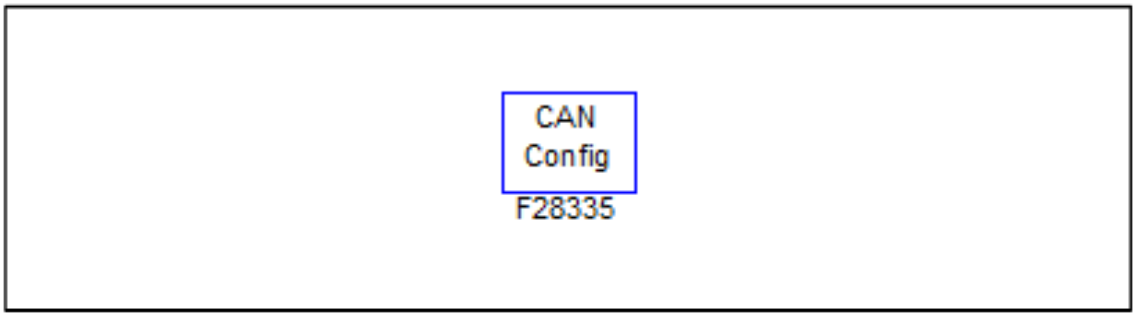
7.15 Controller Area Network （CAN）Bus

F2833x は CAN バスコミュニケーションの機能をもっています。PSIM では CAN バスでで必要な機能を備えています。SimCoder には *CAN Configuration, CAN Input*、*CAN output* の 3 つの機能があります。

7.15.1 CAN Configration

CAN Configuration ブロックは CAN バスソースでデータバイト順や他の設定を定義します。

シンボル:



仕様:

パラメータ	機能
CAN 速度 (MHz)	通信速度（単位はMHz）。 次の周波数、125kHz、250kHz、500kHz、1MHzのどれか1つを選択するか もしくは手入力で設定できます。手入力の場合は1MHzを超えないようご注意ください。
Dataバイト順 Order	データバイトの順番になります。次の 1 つとなります。 - *Least significant byte 1st* - *Most significant byte 1st* “Least significant byte 1st” は最下位バイトが最初に転送され最上位バイトが最後に転送されます。一方“Most significant byte 1st”は最上位バイトが最初に転送され最下位バイトが最後に転送されます。
入力メールボックス数	入力mailboxの数
受信メール紛失検査	もし*Enable*(有効)に設定されると受信メールが無くなった場合、エラーレポート関数の“_ProcCanAErrReport(nErr)”（CAN Aに対して）が呼びだされます。 この関数はエラー状態nErrの数値を返します。 もし、*Disable*(無効)に設定されるとエラーレポート関数は呼ばれません。
バスオフ検査	もし*Enable*(有効)に設定されると受信メールが無くなった場合、エラーレポート関数の“_ProcCanAErrReport(nErr)”（CAN Aに対して）が呼びだされます。 この関数はエラー状態nErrの数値を返します。 もし、*Disable*(無効)に設定されるとエラーレポート関数は呼ばれません。
エラー検査モード	エラーチェックモードを受動か能動かのどちらかを選択できます。受動の場合、エラーカウントが128になったときに割込みが発生します。能動の場合割り込みが毎回発生します。

関数“__ProcCanAErrReport(nErr)”（CAN A に対して）の戻り状態 nErr は 32 ビットの整数です。

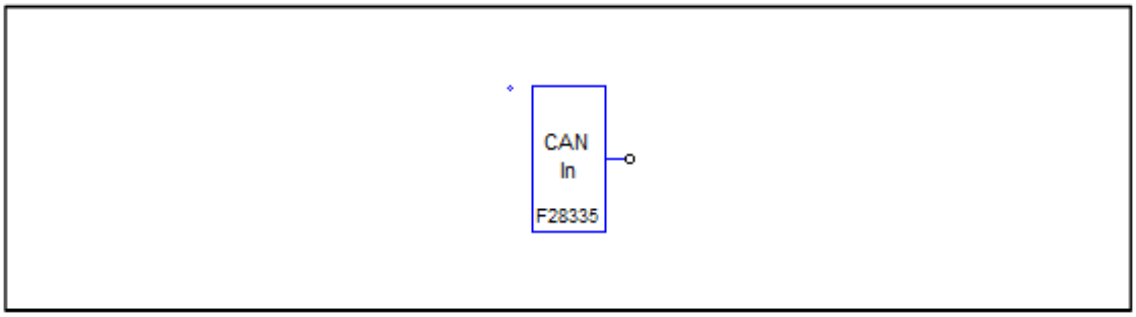
エラーとレジスターCANES 状態からの値を含んでいます。関数“__ProcCanAErrReport(nErr)”からの値が戻るとレジスターCANES はクリアされます。

もし、特定のエラーに対応したい場合は“__ProcCanAErrReport(nErr)”関数に独自のコードを追加することができます。

7.15.2 CAN Input

CAN 入力ブロックは CAN バスから CAN メッセージを受け取ります。

シンボル:



仕様:

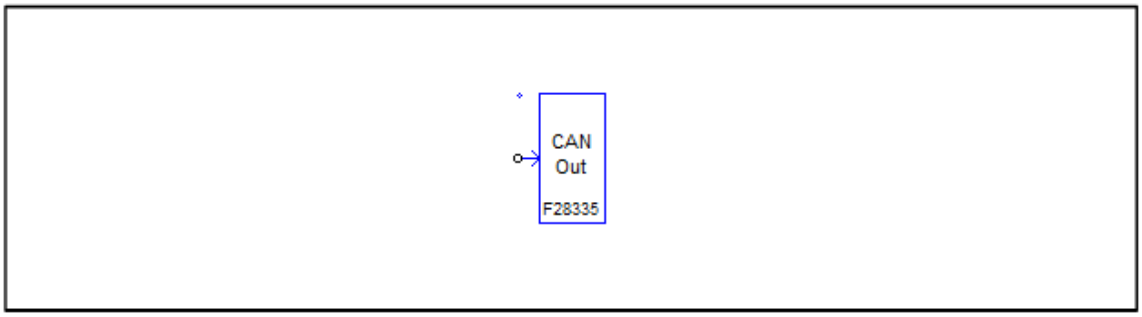
パラメータ	機能
Number of Input	CAN入力数です。8入力まで可能です。
CAN ソース	CANソースはCAN AかCAN Bとなります。
拡張ID使用	Yesに設定された場合、メッセージのIDは29-bitの整数となります。 Noに設定された場合、メッセージのIDは11-bitの整数となります。
メッセージ ID	メッセージのIDです。整数で例えば0x23のようになります。
ローカルマスク	メッセージに対するMASKとなります。整数です。もしメッセージIDのビットがMASKのビットと一致するとメッセージを受信します。そうでない場合は無視されます。例えばmaskが0x380でメッセージIDが0x389の場合、このメッセージが受信されます。もし、メッセージIDが0x480の場合はメッセージは無視されます。
上書き保護フラグ	*Allow overwrite*か*Do not allow overwrite*に設定できます。 例えばメッセージ受信用にメールボックスが設定され、新しいメッセージがCANバス経由で受信されたが古いメッセージがまだメールボックスにあり、何も処理されていない場合、Override flagが”Allow overwrite”に設定されると新しいメッセージが受信され古いメッセージは上書きされます。Flagが”Do not allow overwrite”に設定されていると新しいメッセージは受信されずに古いものはそのまま残ります。
受信頻度	特定期間に入力ブロックで受信したメッセージの数。例えば、最初の入力ブロックで1秒間に20のメッセージを、2番目の入力ブロックで1秒間に30のメッセージを受信する2つの入力ブロックがあるとします。その場合” Receive Message Rate”は最初のブロックに20、2番目のブロックに30と設定されます。
入力 *i* Gain	i 番目の入力に対するゲインでiは1から8で設定できます。出力は入力のゲインの値をかけた値になります。
入力 *i* データ開始位置	メッセージは8データポイントまで可能です。データポイントは1bitから32bitsまで可能です。これはメッセージの現在のデータポイントの開始位置を定義します。
入力 *i*データ終了位置	これはメッセージの現在のデータポイントの終了位置を定義します。
入力 *i* データタイプ	データタイプは、浮動、整数、またはIQ1からIQ30までとなります。
入力 *i*デフォル	SCI入力変数の初期値となります。

トデータ	

7.15.3 CAN Output

CAN 出力ブロックは CAN メッセージを CAN バスへ送ります。

シンボル:



仕様:

パラメータ	機能
Number of Outputs	CAN出力の数です。8出力まで可能です。
CAN ソース	CANソースはCAN AかCAN Bとなります。
拡張ID使用	Yesに設定された場合、メッセージのIDは29-bitの整数となります。 Noに設定された場合、メッセージのIDは11-bitの整数となります。
メッセージ ID	メッセージのIDです。整数で例えば0x23のようになります。
トリガータイプ	トリガのタイプは次のようになります。 - *No trigger*：トリガなし。 - *Rising edge*：トリガソースの立ち上がりエッジでトリガがかかります。 - *Falling edge*：トリガソースの立下りエッジでトリガがかかります。 - *Riding/Falling edge*:とりがソースの立ち上がりと立下りのエッジでトリガがかかります。 立ち上がり／立下りエッジはトリガソースの現在の値と、最終のトリガの値が1以上で差がある場合に適用されます。
トリガーソース	トリガソースはトリガタイプによるグローバル変数の定数にもあります。 トリガタイプが“No Trigger”に設定されるとトリガソースはカウンターリミットを定義します。例えばトリガソースが定数かグローバル変数で値が5の時、5回に１回トリガがかかることを意味しています。すなわちデータが5サイクル毎に送られます。 もしトリガタイプがエッジトリガに設定された場合、トリガソースはグローバル変数のみです。グローバル変数がトリガタイプに依存して立ち上がりか立下りもしくは両方のエッジとなる時にトリガがかかります。
出力 *i* ゲイン	i 番目の出力ゲインでiは1から8で設定できます。出力は入力のゲインの値をかけた値になります。
出力 *i* データ開始位置	メッセージは8個のデータポイントまで可能です。データポイントは1bitから32bitsまで可能です。これはメッセージの現在のデータポイントの開始位置を定義します。
出力 *i* データ終了位置	これはメッセージの現在のデータポイントの終了位置を定義します。

Output *i* Data Type	データタイプは、浮動、整数、またはIQ1からIQ30までとなります。

7.16 割込み時間

割込み時間ブロックは割込みサービスルーチンの時間間隔を測定するのに使われます。

仕様:

パラメータ	機能
Time Output Method	割込み時間測定の定義を行います。 -SCI(time used：使用時間): SCIを使用します。DSPクロックカウントで割込みサービスルーチンとして使われる時間はSCIを介して送信されます。 -SCI(time remaining：残り時間): SCIを使用します。割込みサービスルーチンの残り時間です。DSPクロック数で測定されSCIを介して送信します。time remaininngは現在の最後の割込みから次の割り込みの先頭までの時間として定義されます。 -GPIO0 to GPIO87: GPIOポートを使います。パルスは特定されたGPIOポートで生成されます。パルスは割込みに入る時high(1)にセットされます。割込みが終了する時にはlow(0)にセットされます。オシロスコープでパルス幅を測定できます。
Sampling Frequency	割込みサービスルーチンのサンプリング周波数（単位はHz）

SCI が使われる場合、値は DSP クロックでカウントされます。例えば値が 7500 の時 150MHz の DSP クロックの割込み時間は　7500/150M=50 us となります。

7.17 プロジェクト設定とメモリ配置

TI F2833x 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.sch」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 F2833x 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
- F2833x_Headers_nonBIOS.cmd : ペリフェラルレジスタリンカコマンドファイル
- F28335_FLASH_Lnk.cmd : フラッシュメモリリンカコマンドファイル
- F28335_FLASH_RAM_Lnk.cmd : フラッシュ RAM リンカコマンドファイル
- F28335_RAM_Lnk.cmd : RAM メモリリンカコマンドファイル

注）: リンカコマンドファイルの名前は F28335 でない場合はターゲットハードウェアによりアサインされます。例えばもしターゲットハードウェアが F28334 の場合、ファイル名はそれに応じて

F28334_ FLASH_Lnk.cmd, F28334_ FLASH_RAMLnk.cmd, F28334RAM Lnk.cmd となります。

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 F2833x 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 はリンカコマンドファイル 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 はリンカコマンドファイル 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 はリンカコマンドファイル 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：

生成されたリンカ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：

RAM Memory
0x0000_0x07FF(2K)
Interrupt vectors
Stack
0x8000_0xFFFF(32K*)
Program and data space

Flash Release の設定の場合：

RAM Memory	Flash Memory
0x0000_0x07FF(2K) Interrupt vectors Stack 0x8000-0xFFFF(32K*) data space	0x300000_0x33FFFF(256K**) Program Password etc.

注意：

＊ 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。

- F28335,F28334： 0x8000 から 0xFFFF(32K)
- F28332 ： 0x8000 から 0xDFFF(16K)

もしプログラムとデータ空間を合わせたものが RAM 空間を越えた場合は Flash Release 設定を選択する必要があります。

＊＊ SimCoder によりあらかじめ定義されているプログラム空間に対する flash memory は

- F28335： 0x300000 から 0x33FFFF(256K)
- F28334： 0x320000 から 0x33FFFF(128K)
- F28332： 0x330000 から 0x33FFFF(64K)

となります。

8 TI F2803x Hardware Target

8.1 概要

オプションモジュールTI F2803x 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社製固定小数点型DSP F2803xを搭載した基板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。

TI F2803xハードウェアターゲットはF2803x全てのパッケージに対応しています。

TI F2803x Hardware Targetには下記の機能があります。

- 3-phase、2-phase、1-phase、とAPWM(Single PWM)生成機能
- 可変周波数PWM
- 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
- コンパレータ入力、出力とDAC
- デジタル入力出力機能
- Up/Downカウンタ機能
- エンコーダ、エンコーダステート機能
- キャプチャ、キャプチャステート機能
- SCI基板設定機能、入力出力
- SPI基板設定機能、デバイス、入力出力
- CAN設定機能、入力出力
- 割込み時間機能
- 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。

TI F2803x 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、SimCoder は 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 F2803x 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。次の図は F2803x 80-pin PN QFP ポートアサイン解説します。

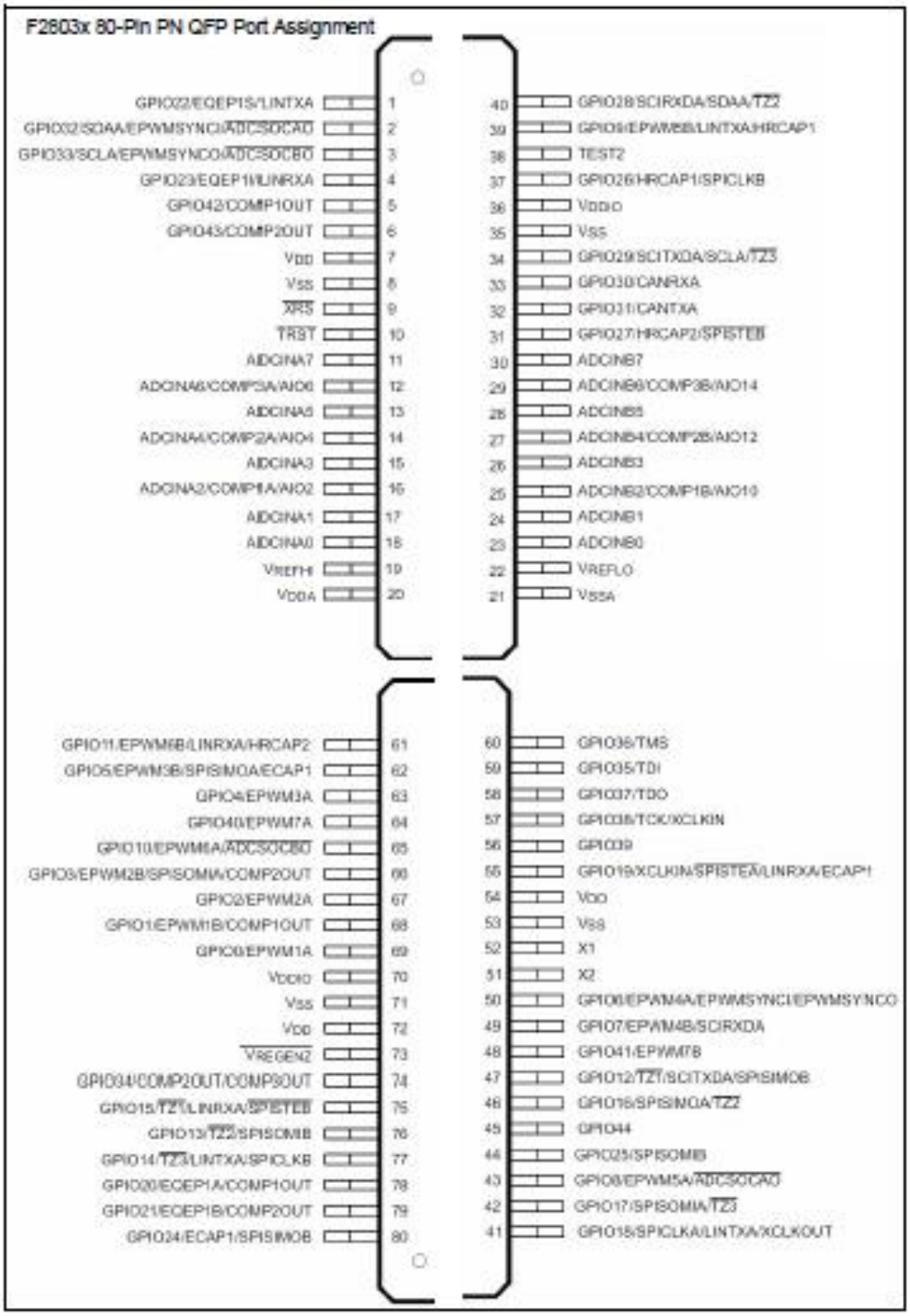
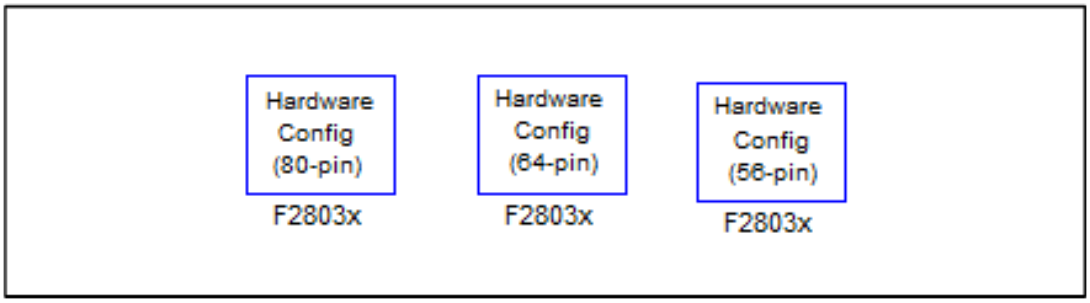


図 8- 1　F2803x 80-pin PN QFP ポートアサイン図

8.2　ハードウェア構成

F2803x には 16 チャネルのアナログ入力と独立した 45 のプログラムマルチプレクサの GPIO ポートがあります。GPIO ポートの大部分はいくつかの機能の 1 つに割り当てられます。例えば GPIO1 のポートはデジタル入力と出力まらは PWM 出力として使用できます。16A/D チャネルのいくつかも同様にデジタル入力、出力、コンパレータ入力として定義できます。それゆえ PSIM 回路図の中で正しく GPIO ポートの機能を定義しないとなりません。

シンボル:



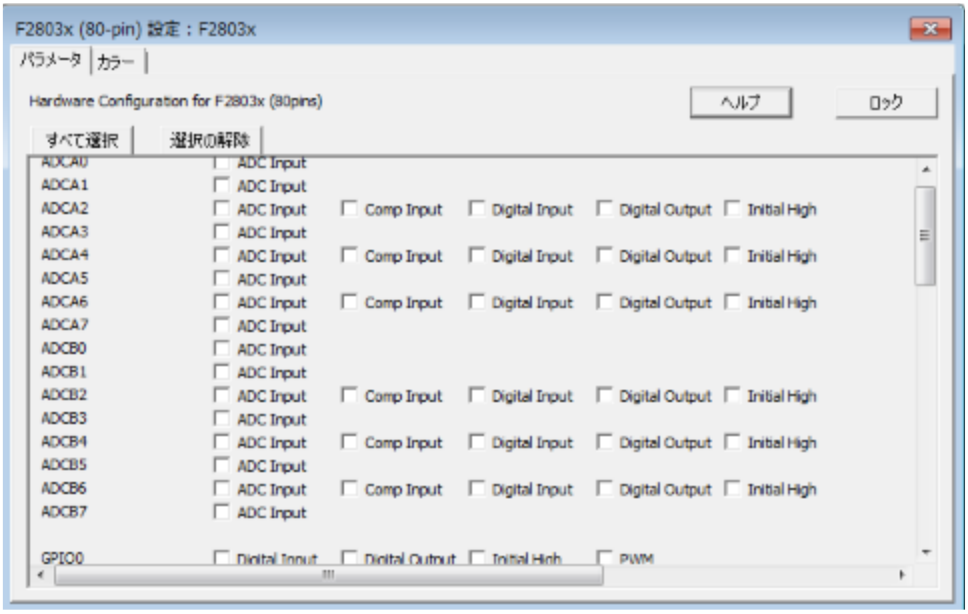


図 8- 2　ブロックのダイアログウィンドウ

ハードウェア構成ブロックは F2803x ハードウェアの I/O ポートを定義します。ポートは正しくアサインしてください。使用しないポートはチェックを入れないようにしてください。

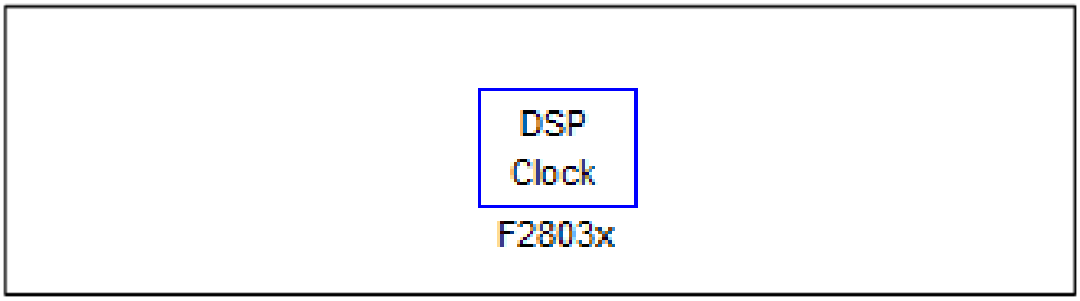
各々の GPIO ポートのチェックボックスにある機能が使用可能です。SimCoder ではこのボックスがチェックされている機能だけが使用できます。例えばポート GPIO1 では“Digital Input”がチェックされるとこのポートは“Digital Input”となります。他の機能としては使用できません

PSIM 回路中で PWM 出力として使用されるとエラーメッセージが出ます。

8.3 DSP クロック

DSP クロックブロックは外部クロック周波数と F2803x DSP 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
DSP クロックソース	F2803xは次の5つの方法でシステムクロックを提供します。 ・Internal Oscillator1(内部発信器1) ・Internal Oscillator2(内部発信器2) ・External Oscillator(外部発信器) ・External clock(GPIO19) ・External clock(GPIO38)
外部クロック（MHz)	DSPボード上の外部クロックの周波数(単位はMHz） 周波数は整数値である必要があり最大値は10MHzです。このパラメータはDSPクロックソースが内部オシレータ1か2と選択された場合は無視されます。
DSP速度（MHz)	DSP速度（単位はMHz）。 入力するDSP速度は整数値でなければなりません。外部クロック周波数の整数倍（1~12倍）でなければなりません。60MHzが最大値です。

DSP クロックブロックが回路中で使用されていない場合は DSP ブロックのデフォルト値が使用されます。

8.4 PWM 生成器

F2803x には、

PWM1(GPIO0、GPIO1)

PWM2(GPIO2、GPIO3)

PWM3(GPIO4、GPIO5)

PWM4(GPIO6、GPIO7)

PWM5(GPIO8、GPIO9)

PWM6(GPIO10、GPIO11)

PWM7(GPIO40、GPIO41)

のように合計で 7 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 7 つの PWM を下記のように使うことができます。

-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。

4. -2-phase PWM 生成器：7 種類あります。PWM1~7 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。

5. -1-phase PWM 生成器：PWM1~7 互いに相補的な二つの出力をもっています。

-位相シフト付 1-phase PWM 生成器：PWM2～6 は互いに相補的な 2 つの出力をもっています。

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 APWM 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。

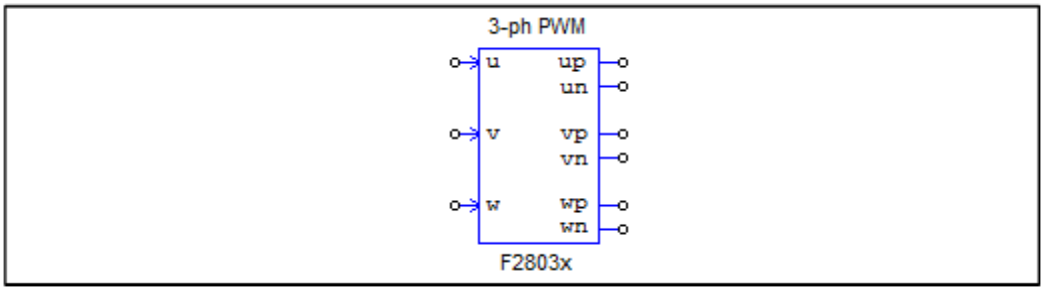
SimCoder の PWM 生成器は内部にスイッチング一回分の遅れを持っており、PWM の出力は入力が入ってから 1 サイクル分遅れて出力されます。これは実際の DSP の持つ遅れをシミュレーションに反映するためです。

PWM 周波数倍率は F2803x の制限では 1ー3 です。(設定は 1 から 100 までできます。3 より大きくしたい場合は、PSIM が未使用の PWM を使って、制御周波数で割り込みを引き起こします。この未使用の PWM は周期的な割り込みを生成するためにしか使用されません、PWM の出力端子は、他の機能に設定することができます。もしシステム内に未使用 PWM がない場合、CPU タイマーが使用されます。

8.4.1 3-phase PWM 生成器：

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。

シンボル:



仕様:

パラメータ	機能
PWM ソース	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。
デッドタイム	PWM生成器のデッドタイム（Td）の設定。単位は[sec]
サンプリング周波数	PWM生成器のサンプリング周波数の設定。単位は[Hz] PWM信号のデューティサイクルはここで設定した周波数で更新されます。
PWM周波数倍率	PWM周波数とサンプリング周波数の倍率の設定。 1～100の設定ができます。 例えばサンプリング周波数が50kHzでスケーリングファクターが3の場合はPWMスイッチング周波数は150kHzになります。ゲーティング信号は50kHzでもしくは3回につき1回のスイッチングサイクルで更新を行います。
キャリア波形タイプ	キャリア波形タイプの設定。 - 「Triangular(start low)」：キャリアを三角波、PWM出力の初期値はLow。 - 「Triangular(start high)」：キャリアを三角波、PWM出力の初期値はHIGH。 - 「Sawtooth(start low)」：キャリアをのこぎり波、PWM出力の初期値はLow。 - 「Sawtooth(start high)」：キャリアをのこぎり波、PWM出力の初期値はHIGH。
A/D変換トリガ	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」：A/D変換器のGroup Aへのトリガを有効にします。 - 「Trigger ADC Group B」：A/D変換器のGroup Bへのトリガを有効にします。 - 「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。
A/D変換トリガ位置	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。
トリップゾーンi使用	PWM生成器と繋がる6つの各トリップゾーンの設定。 - 「Disable Trip-Zone i」：対応するトリップゾーン信号を無効にします。 - 「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。 - 「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。
PWMA DC Trip Scrl(DCAH)	PWMA用のDigital compare(DC)トリップソースDCAHです。PWMAは3つのトップスイッチに対して出力"up","vp","wp"を参照します。 PWMチャネルはDCAHとDCALの2つのDCトリップソースを持っています。 トリップソースは次の中の1つとなります。 - *Do not use*: PWMはこの信号を使用しません。 - *Trip-zone 1*: PWMはtrip-zone 1をdigital compare input signalとして使用します。 - *Trip-zone 2*: PWMはtrip-zone 2をdigital compare input signalとして使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare input signalとして使用します。 - *Comparator 1*: PWM はComparator 1をdigital compare input signalとして使用します。.

	- *Comparator 2*: PWM はComparator 2 をdigital compare input signalとして使用します。. - *Comparator 3*: PWM はComparator 3 をdigital compare input signalとして使用します。.
PWMA DC Trip Scrl(DCAL)	PWMA用のDigital compare(DC)トリップソースDCALです。トリップソースは次のどれかになります。 - *Do not use*: PWM はこの信号を使用しません。 - *Trip-zone 1*: PWMはtrip-zone1をdigital compare入力信号として使用します。 - *Trip-zone 2*: PWMはtrip-zone2をdigital compare入力信号として使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare入力信号として使用します。 - *Comparator 1*: PWMはComparator1をdigital compare入力信号として使用します。 - *Comparator 2*: PWMはComparator2をdigital compare入力信号として使用します。 - *Comparator 3*: PWMはComparator3をdigital compare入力信号として使用します。
PWMA 1-shotEvt(DCAEVT1)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。 - *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。 - *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2 is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
PWMA CBC Evt(DCAEVT2)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。 - *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。 - *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2 is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
PWMB DC Trip Src1(DCBH)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip-zone 1*: PWMはtrip-zone1をdigital compare入力信号として使用します。 - *Trip-zone 2*: PWMはtrip-zone2をdigital compare入力信号として使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare入力信号として使用します。 - *Comparator 1*: PWMはComparator1をdigital compare入力信号として使用します。 - *Comparator 2*: PWMはComparator2をdigital compare入力信号として使用します。 - *Comparator 3*: PWMはComparator3をdigital compare入力信号として使用します。
PWMB DC Trip Src2(DCBL)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip-zone 1*: PWMはtrip-zone1をdigital compare入力信号として使用します。 - *Trip-zone 2*: PWMはtrip-zone2をdigital compare入力信号として使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare入力信号として使用します。 - *Comparator 1*: PWMはComparator1をdigital compare入力信号として使用します。 - *Comparator 2*: PWMはComparator2をdigital compare入力信号として使用します。 - *Comparator 3*: PWMはComparator3をdigital compare入力信号として使用します。
PWMB 1-shot Evt (DCBEVT1)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。

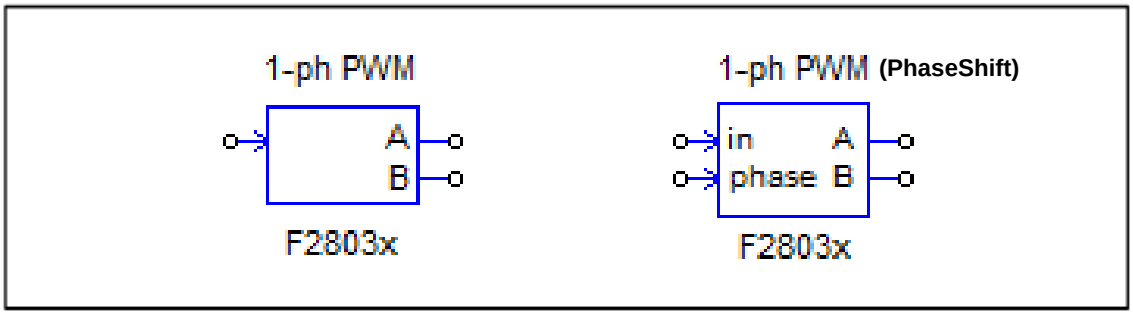
	- *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。 - *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2 is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
PWMB CBC Evt (DCBEVT2)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。 - *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。 - *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2 is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
DC Event Filter Source	- *Do not use*: DC event filterを使用しません。 - *DCAEVT1*: DCAEVT1をfilter sourceとして使用します。. - *DCAEVT2*: DCAEVT2をfilter sourceとして使用します。. - *DCBEVT1*: DCBEVT1をfilter sourceとして使用します。 - *DCBEVT2*: DCBEVT2をfilter sourceとして使用します。
Blanking Window Pos(us)	PWM周期におけるBlanking windowの開始位置を設定します。（単位：us）
Blanking Window Width (us)	Blanking windowの幅を指定します。(単位：us)　幅はハードウェアによって制限され、255/CPU周波数　として計算されます。 例えばCPUのスピードが90MHｚの場合幅の範囲は0から2.83usとなります。
Blanking Window Range	Blanking windowどう適用するかを定義します。 - *In the window*: ブランキング動作の適用範囲はウィンドウ(指定開始位置から指定幅のウィンドウ)内になります - *Out of window*: ブランキング動作の適用範囲は指定ウィンドウ外となります。
Applying Event Filtering	イベントフィルタリングをデジタル比較イベントにどう適用するかを定義します。イベントフィルタリングは次のデジタル比較イベントのいずれかの組み合わせに対応します。：DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2
トリップアクション	トリップゾーン信号にトリガされた時のPWMモジュールの出力を設定します。 - 「High Impedance」：PWMの両出力端子はハイインピーダンスに設定されます。 - 「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設定されます。 - 「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設定されます。 - 「No action」：何も動作を設定しません。
ピーク間電圧	キャリア波のピーク間電圧Vppの設定。
オフセット電圧	キャリア波のオフセット電圧Voffsetの設定。
初期入力値u, v, w	u、v、w三相入力ノードの初期値の設定。
最初からPWM　信号出力	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。
シミュレーション出力モード	シミュレーションの出力モードはSwitching modeかAverage modeに設定されます。“Switching mode”へ設定された時にはPWMブロックの出力はPWM信号となります。“Average mode”へ設定された時にはPWMブロックの出力は平均モード信号となります 例えばPWMブロックの入力をVuとすると出力のupとunは Vup=Vu/(Vpk_pk+Voffset)

	Vun=－Vu/(Vpk_pk+Voffset) となります。Average modeに設定するとPWMブロックの出力はaverage mode modelのインバータへ接続できます。

8.4.2 1-phase PWM 生成器（外部位相シフト機能付きを含む）：

1-phase PWM 生成器と外部入力位相シフト機能付き 1-phase PWM 生成器はほぼ同じ属性を持っています。違いとしては１相 PWM ブロックはパラメータを通して位相を定義し、１相 PWM(位相シフト)ブロックは外部入力からの位相シフトを読みます(シンボルのラベル Phase 部分が該当します)。

シンボル:



仕様:

パラメータ	機能
PWM ソース	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「PWM7」を選択します。
出力モード	PWM生成器の出力モードの設定。 - 「Use PWM A&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。 - 「Use PWM A」：PWM出力Aのみ使用します。 - 「Use PWM B」：PWM出力Bのみ使用します。
デッドタイム	PWM生成器のデッドタイム（Td）の設定。単位は[sec]
サンプリング周波数	PWM生成器のサンプリング周波数の設定。単位は[Hz] PWM信号のデューティサイクルはここで設定した周波数で更新されます。
PWM 周波数倍率	PWM周波数とサンプリング周波数の倍率の設定。 1～100倍に設定ができます。 例えばサンプリング周波数が50kHzでスケーリングファクターが3の場合はPWMスイッチング周波数は150kHzになります。ゲーティング信号は50kHzで、または3回につき1回のスイッチングサイクルで更新を行います。
キャリア波形　タイプ	キャリア波形タイプとPWM出力信号の初期状態の設定。 - 「Triangular(start low)」：キャリアは三角波で、PWM出力の初期値はLow(0)。 - 「Triangular(start high)」：キャリアは三角波で、PWM出力の初期値はHIGH(1)。 - 「Sawtooth(start low)」：キャリアはのこぎり波で、PWM出力の初

	- 期値はLow(0)。 - 「Sawtooth(start high)」：キャリアはのこぎり波で、PWM出力の初 - 期値はHIGH(1)。
A/D変換トリガ	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」：PWMは複数のA/Dチャネルをトリガしま - す。 -
A/D変換トリガ位置	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。
トリップゾーン i 使用	PWM生成器と繋がる6つの各トリップゾーンの設定。 - 「Disable Trip-Zone i」：対応するトリップゾーン信号を無効にします。 - 「One shot」：One shotモードでトリップゾーン信号を使用します。トリガがかかるとPWM出力は手動で開始させなければなりません。 - 「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。
PWMA DC Trip Src1(DCAH)	PWMA用のDigital compare(DC)トリップソースDCAHです。PWMAは3つのトリップスイッチに対して出力"up","vp","wp"を参照します。 PWMチャネルはDCAHとDCALの2つのDCトリップソースを持っています。 トリップソースは次の中の1つとなります。 - *Do not use*: PWMはこの信号を使用しません。 - *Trip-zone 1*: PWMはtrip-zone 1をdigital compare input signalとして使用します。 - *Trip-zone 2*: PWMはtrip-zone 2をdigital compare input signalとして使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare input signalとして使用します。 - *Comparator 1*: PWM はComparator 1をdigital compare input signalとして使用します。. - *Comparator 2*: PWM はComparator 2 をdigital compare input signalとして使用します。. - *Comparator 3*: PWM はComparator 3 をdigital compare input signalとして使用します。.
PWMA DC Trip Src2(DCAL)	PWMA用のDigital compare(DC)トリップソースDCALです。トリップソースは次のどれかになります。 - *Do not use*: PWM はこの信号を使用しません。 - *Trip-zone 1*: PWMはtrip-zone1をdigital compare入力信号として使用します。 - *Trip-zone 2*: PWMはtrip-zone2をdigital compare入力信号として使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare入力信号として使用します。 - *Comparator 1*: PWMはComparator1をdigital compare入力信号として使用します。 - *Comparator 2*: PWMはComparator2をdigital compare入力信号として使用します。 - *Comparator 3*: PWMはComparator3をdigital compare入力信号として使用します。
PWMA 1-shotEvt(DCAEVT1)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。 - *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。

	- *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2 is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
PWMA CBC Evt(DCAEVT2)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。 - *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。 - *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2 is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
PWMB DC Trip Src1(DCBH)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip-zone 1*: PWMはtrip-zone1をdigital compare入力信号として使用します。 - *Trip-zone 2*: PWMはtrip-zone2をdigital compare入力信号として使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare入力信号として使用します。 - *Comparator 1*: PWMはComparator1をdigital compare入力信号として使用します。 - *Comparator 2*: PWMはComparator2をdigital compare入力信号として使用します。 - *Comparator 3*: PWMはComparator3をdigital compare入力信号として使用します。
PWMB DC Trip Src1(DCBL)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip-zone 1*: PWMはtrip-zone1をdigital compare入力信号として使用します。 - *Trip-zone 2*: PWMはtrip-zone2をdigital compare入力信号として使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare入力信号として使用します。 - *Comparator 1*: PWMはComparator1をdigital compare入力信号として使用します。 - *Comparator 2*: PWMはComparator2をdigital compare入力信号として使用します。 - *Comparator 3*: PWMはComparator3をdigital compare入力信号として使用します。
PWMB 1-shot Evt (DCBEVT1)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。 - *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。 - *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
PWMB CBC Evt (DCBEVT2)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。 - *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。 - *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2 is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
DC Event Filter Source	- *Do not use*: DC event filterを使用しません。 - *DCAEVT1*: DCAEVT1をfilter sourceとして使用します。. - *DCAEVT2*: DCAEVT2をfilter sourceとして使用します。. - *DCBEVT1*: DCBEVT1をfilter sourceとして使用します。 - *DCBEVT2*: DCBEVT2をfilter sourceとして使用します。
Blanking Window Pos(us)	PWM周期におけるBlanking windowの開始位置を指定します。（単位：us）
Blanking Window Width (us)	ブランキングウィンドウの幅(単位：us)　幅はハードウェアによって制限され、255/CPUとして計算されます。　例えばCPUのスピードが90MHｚの場合幅の範囲は0から2.83usとなります。

Blanking Window Range	ブランキング動作の適用範囲をを定義します。 - *In the window*: ブランキング動作の適用範囲はウィンドウ(指定開始位置から指定幅のウィンドウ)内になります。 - *Out of window*: ブランキング動作の範囲範囲は指定ウィンドウ外となります。
Applying Event Filtering	イベントフィルタリングをデジタル比較イベントに適用される方法を指定します。イベントフィルタリングは次のデジタル比較イベントの組み合わせに適用できます。 : DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2
トリップ アクション	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A端子がハイレベル、B端子がローレベルになります。 - 「PWM A low & PWM B high」: PWMのA端子はローレベB端子がハイレベルになります。 - 「No action」: 何も動作を設定しません。
ピーク間電圧	キャリア波のピーク間電圧Vppの設定。
オフセット電圧	キャリア波のオフセット電圧Voffsetの設定。
位相シフト	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)
初期入力値	入力ノードの初期値の設定。
Use HRPWM	高分解能PWMの定義を行います。 *Do not use HRPWM*:高分解能PWMを使用しません。 *UseHRPWM*(校正なし): 校正なしで高分解能PWMを使用します。 *UseHRPWM*(校正あり): 校正ありで高分解能PWMを使用します。
最初からPWM信号	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。
シミュレーション出力モード	シミュレーションの出力モードはSwitchingモードかAverageモードに設定されます。“Switchingモード”へ設定された時にはPWMブロックの出力はPWM信号となります。“Averageモード”へ設定された時にはPWMブロックの出力は平均モード信号となります 例えばPWMブロックの入力をVuとすると出力のupとunは Vup=Vu/(Vpk_pk+Voffset) Vun=－Vu/(Vpk_pk+Voffset) となります。Average mode に設定されるとPWMブロックの出力はaverage mode modelのインバータへ接続されます。

Phase Shift

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。例えば F2803xDSP では PWM 位相シフトは次のようになります。

–どのような PWM でもその前の PWM も PSIM 回路に位相シフト PWM として使用されることが必要になります。例えば、PWM1、2、3 は正しい位相シフト系列ですが、PWM1、3、4 は正しい位相シフト系列ではありません。

- –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。

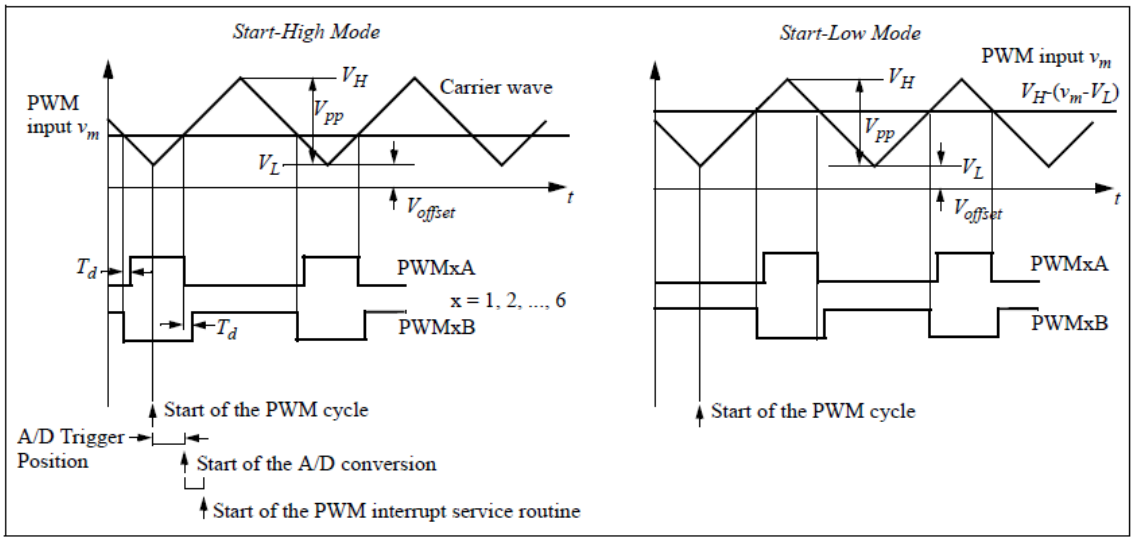
つまり、PWM1 を基準として PWM3、または PWM5 または PWM6 を位相シフトすることはできま

せん。

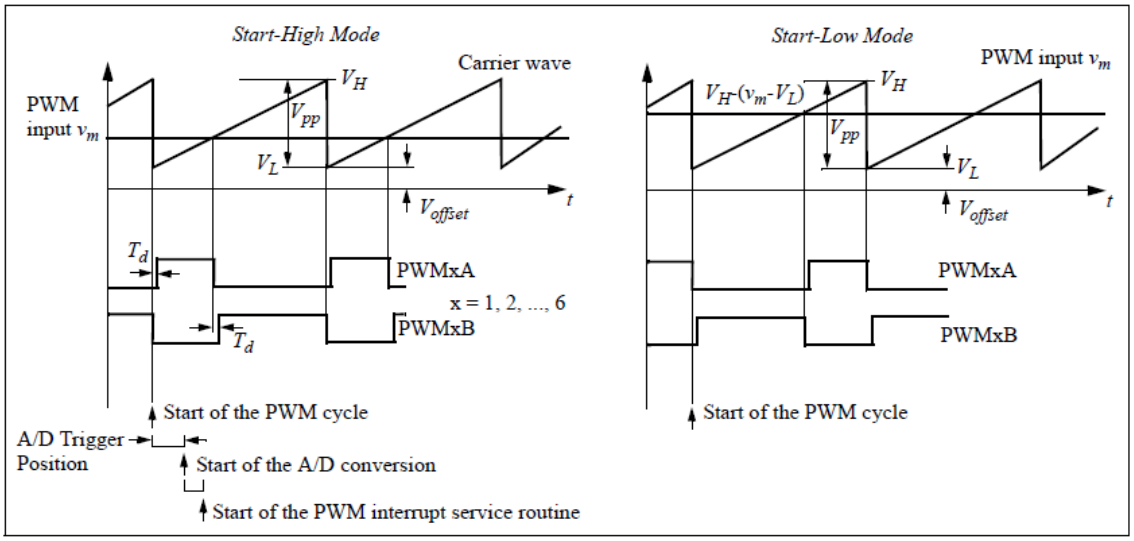
位相シフトの値は、度[deg]になります。値が-30°の場合、出力は基準の PWM 生成器の出力に対してスイッチングサイクルの 30°右に（遅れ）にシフトされます。これは、30°右に PWM の搬送波をシフトすることと同じです。位相の値が 30°の場合、出力は 30°左（進み）にシフトされます。

搬送波

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。

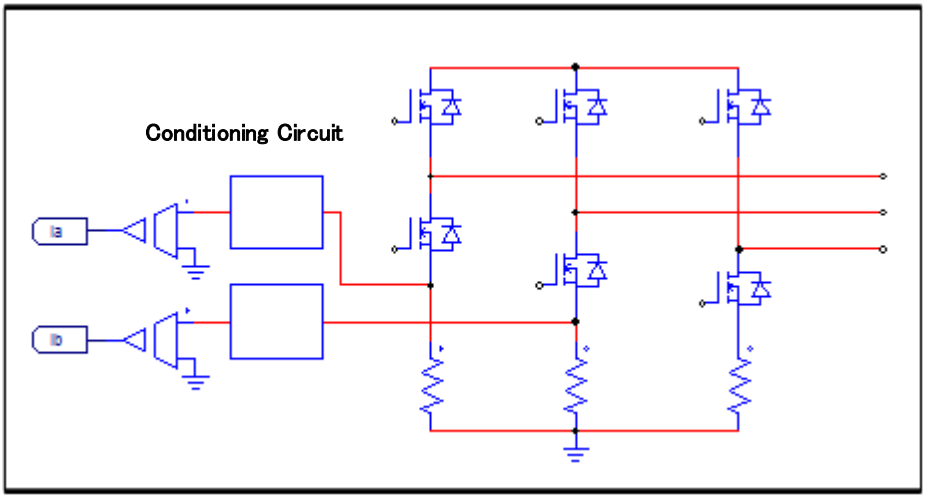
一方、”Start-Low”モードでは、PWM 正出力 PWMA はスイッチングサイクルの開始時に LOW となり、搬送波が「VH－(Vm－VL)」の値より大きい際に HIGH となります。例えば、搬送波が 0～1、つまり VL=0、VH=1 の時、Vm が 0.2 の場合、PWM 出力 PWMA は搬送波が 0.8 より大きい間は HIGH となります。

キャリア開始モードはスイッチ電流の測定方法によりかわります。3 相インバータの場合はスイッチ電流が測定され、スタートハイモードが選択されます。これは上のスイッチが周期の最初にトップスイッチゲーティング信号が高く、電流が流れていることをになります。

一方低いスイッチ電流が測定されているとスタートローモードが選択されます。これは周期の最初に

トップスイッチゲーティング信号が低く、下のスイッチゲーティング信号が高く、電流が流れていることになります。

次のような回路の場合、位相 A と B の下のスイッチ電流が測定されます。この場合はキャリアスタートローモードを選択すべきです。



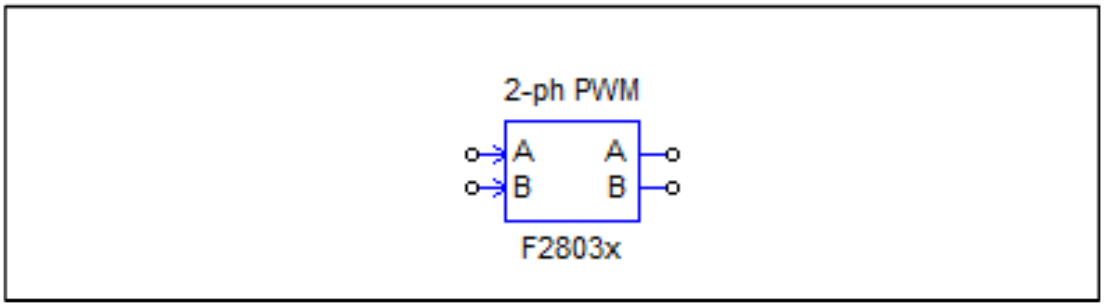
注：”Start-Low”モードでは、PWM 信号を生成するために搬送波と比較する前に PWM 入力”Vm”は”VH－(Vm－VL)”に変換されます。変換によって、”Start-Low”と”Start-High”の両方のモードで同じデューティー比の表現となります。例えば、VL=0,VH=1 のノコギリ波、または VL=－VH の三角波

の場合、PWMA 出力のデューティー比 D は”Start-Low”と”Start-High”の両方で「D=Vm/VH」となります。

8.4.3 2-phase PWM 生成器：

2 相 PWM ブロックは 6 つの動作モードがあります。

シンボル:



仕様:

パラメータ	機能
PWM ソース	PWM生成器の出力ピン設定。 PWM1~7までを選択します。
モード・タイプ	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次の図を参照してください。
サンプリング周波数	PWM生成器のサンプリング周波数の設定。単位は[Hz] PWM信号のデューティサイクルはここで設定した周波数でアップデートされます。
PWM 周波数倍率	PWM周波数とサンプリング周波数の倍率の設定。 1～100倍に設定ができます。例えばサンプリング周波数が50kHzでスケーリグ倍率が3の場合、PWMスイッチング周波数は150kHzとなります。つまりスイッチは150kHzで動作します。がゲーティング信号は50kHzで、または3スイッチングサイクル毎に1回更新されます。
A/D変換トリガ	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 」：A/D変換のトリガ機能を有効にします。 -
A/D変換トリガ位置	A/D変換チャネルのPWMモジュールによるトリガされた変換位置。 トリガされた変換位置は搬送波の開始時にまたは搬送波の途中のいずれかに設定できます。途中トリガ位置は動作モード4,5、及び6の場合に有効です。もしPWMがADCをトリガしない場合はこの位置はPWMの割り込みの位置になります。
トリップゾーン *i* 使用	トリップゾーン*i*信号の使用方法の設定。「Do not use」：PWMはこの信号を使用しません。「One shot」：PWMはトリップゾーン*i*をOne shot信号として使用します。 - 「Cycle by cycle」：PWMはトリップゾーン*i*をcycle-by-cycle信号として使用します。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。

PWMA DC Trip Src1(DCAH)	PWMA用のDigital compare(DC)トリップソースDCAHです。PWMAは3つのトップスイッチに対して出力”up”,”vp”,"wp”を参照します。 PWMチャネルはDCAHとDCALの2つのDCトリップソースを持っています。 トリップソースは次の中の1つとなります。 - *Do not use*: PWMはこの信号を使用しません。 - *Trip-zone 1*: PWMはtrip-zone 1をdigital compare input signalとして使用します。 - *Trip-zone 2*: PWMはtrip-zone 2をdigital compare input signalとして使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare input signalとして使用します。 - *Comparator 1*: PWM はComparator 1をdigital compare input signalとして使用します。. - *Comparator 2*: PWM はComparator 2 をdigital compare input signalとして使用します。. - *Comparator 3*: PWM はComparator 3 をdigital compare input signalとして使用します。.
PWMA DC Trip Src2(DCAL)	PWMA用のDigital compare(DC)トリップソースDCALです。トリップソースは次のどれかになります。 - *Do not use*: PWM はこの信号を使用しません。 - *Trip-zone 1*: PWMはtrip-zone1をdigital compare入力信号として使用します。 - *Trip-zone 2*: PWMはtrip-zone2をdigital compare入力信号として使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare入力信号として使用します。 - *Comparator 1*: PWMはComparator1をdigital compare入力信号として使用します。 - *Comparator 2*: PWMはComparator2をdigital compare入力信号として使用します。 - *Comparator 3*: PWMはComparator3をdigital compare入力信号として使用します。
PWMA 1-shotEvt(DCAEVT1)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。 - *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。 - *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2 is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
PWMA CBC Evt(DCAEVT2)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。 - *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。 - *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2 is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
PWMB DC Trip Src1(DCBH)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip-zone 1*: PWMはtrip-zone1をdigital compare入力信号として使用します。 - *Trip-zone 2*: PWMはtrip-zone2をdigital compare入力信号として使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare入力信号として使用します。 - *Comparator 1*: PWMはComparator1をdigital compare入力信号として使用します。 - *Comparator 2*: PWMはComparator2をdigital compare入力信号として使用します。 - *Comparator 3*: PWMはComparator3をdigital compare入力信号として使用します。
PWMB DC Trip Src1(DCBL)	- *Do not use*: PWMはこの信号を使用しません。

	- *Trip-zone 1*: PWMはtrip-zone1をdigital compare入力信号として使用します。 - *Trip-zone 2*: PWMはtrip-zone2をdigital compare入力信号として使用します。 - *Trip-zone 3*: PWMはtrip-zone3をdigital compare入力信号として使用します。 - *Comparator 1*: PWMはComparator1をdigital compare入力信号として使用します。 - *Comparator 2*: PWMはComparator2をdigital compare入力信号として使用します。 - *Comparator 3*: PWMはComparator3をdigital compare入力信号として使用します。
PWMB 1-shot Evt (DCBEVT1)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。 - *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。 - *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
PWMB CBC Evt (DCBEVT2)	- *Do not use*: PWMはこの信号を使用しません。 - *Trip source 1 is low*: PWMはソース信号1がlowの場合tripされます。 - *Trip source 1 is high*: PWMはソース信号1がhighの場合tripされます。 - *Trip source 2 is low*: PWMはソース信号2がlowの場合tripされます。 - *Trip source 2 is high*: PWMはソース信号2がhighの場合tripされます。 - *Trip source 1 low & source 2 high*: PWMはソース1信号がlowで同時にソース2信号がhighの場合tripされます。
DC Event Filter Source	- *Do not use*: DC event filterを使用しません。 - *DCAEVT1*: DCAEVT1をfilter sourceとして使用します。. - *DCAEVT2*: DCAEVT2をfilter sourceとして使用します。. - *DCBEVT1*: DCBEVT1をfilter sourceとして使用します。 - *DCBEVT2*: DCBEVT2をfilter sourceとして使用します。
Blanking Window Pos(us)	PWM周期におけるBlanking windowの開始位置を指定します。（単位：us）
Blanking Window Width (us)	ブランキングウィンドウの幅(単位：us)　幅はハードウェアによって制限され、255/CPUとして計算されます。　例えばCPUのスピードが90MHzの場合幅の範囲は0から2.83usとなります。
Blanking Window Range	ブランキング動作の適用範囲をを定義します。 - *In the window*: ブランキング動作の適用範囲はウィンドウ(指定開始位置から指定幅のウィンドウ)内になります。 - *Out of window*: ブランキング動作の範囲範囲は指定ウィンドウ外となります。
Applying Event Filtering	イベントフィルタリングをデジタル比較イベントに適用される方法を指定します。イベントフィルタリングは次のデジタル比較イベントの組み合わせに適用できます。　：DCAEVT1, DCAEVT2, DCBEVT1, and DCBEVT2
トリップアクション	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。
ピーク間電圧	キャリア波のピーク電圧Vpkの設定。
初期入力値A, B	A、B入力ノードの初期値の設定。
最初からPWM信号	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。

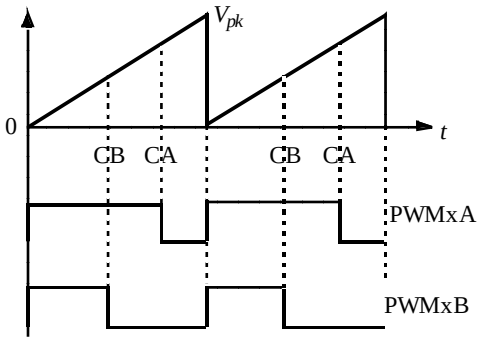
	- 「Do not start」：「Start PWM」関数を実行するまでPWM出力をしません。

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 Vpk 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-pahse 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。

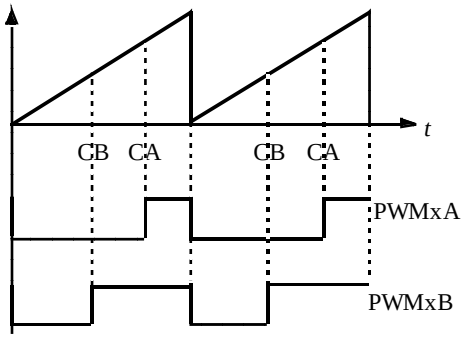
Mode 1:



動作モード 2：

動作モード 1 と違い、各出力のターンオンタイミングを決めるモードです。

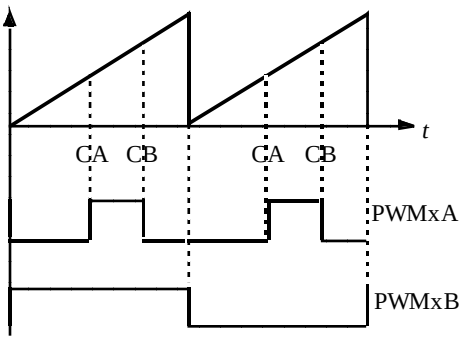
Mode 2:



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。

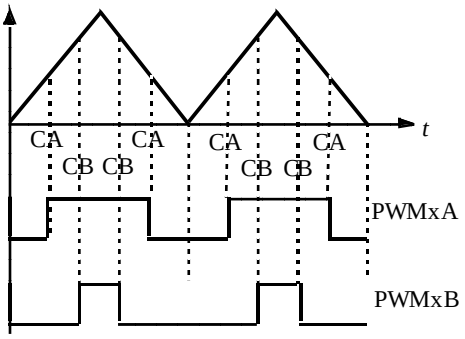
Mode 3:



動作モード 4：

キャリアは三角波になります。各入力は対応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。

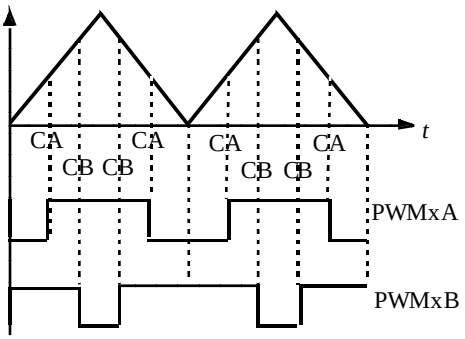
Mode 4:



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。

Mode 5:

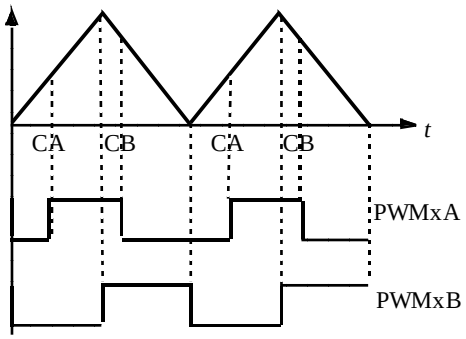


動作モード 6：

入力 A は PWM A のターンオンのタイミングを決め、入力 B は PWM A 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。

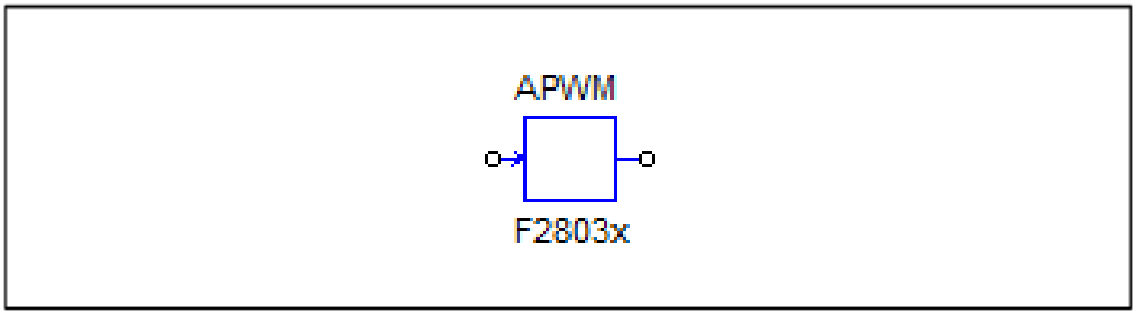
Mode 6:



8.4.4 Single PWM 生成器(APWM)：

SinglePWM 生成器は APWM とも言われキャプチャと同じリソースを共有しています。

シンボル:



仕様:

パラメータ	機能
PWM ソース	Single PWM生成器(APWM生成器)はキャプチャと共用で3つの指定されたGPIOポートを使用できます。 - APWM1(GPIO5, GPIO19, GPIO24) -
PWM 周波数	PWM生成器の周波数の設定。単位は[Hz]
キャリア波形タイプ	キャリア波形タイプの設定。 次のいずれかになります。: - *のこぎり波 (低でスタート)*: キャリアをのこぎり波、PWM出力の初期値はLOW。 - *のこぎり波 (高でスタート)*: キャリアをのこぎり波、PWM出力の初期値はHIGH。
スップ出力	PWMジェネレータが停止している時の出力設定。 - 出力 低: PWM出力は低に設定されます。 - 出力 高: PWM出力は高に設定されます。
ピーク間電圧	キャリア波のピーク間電圧の設定。
オフセット値	キャリア信号のオフセット電圧の設定。
位相シフト	PWM生成器の出力に対しての位相シフト値（単位deg）
初期入力値	入力の初期値の設定。
最初からPWM信号	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。

１相 PWM 生成器と同様に APWM 生成器は他の PWM 生成器に関して位相シフトをもつ PWM 信号を生成します。位相シフトのルールは１相 PWM 生成器と同じです。

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。

8.4.5 PWM ブロック間の同期

PWM の 3 つのタイプで同期が可能です。位相シフトはお互いの間で定義できます。単相 PWM、単相 PWM(位相シフト付)と APWM(もしくは単層 PWM(キャプチャと共用)です。

単相 PWM ブロックは別の PWM 信号で位相シフトした PWM 信号を生成できます。通常の PWM ブロックには PWM1､2､3 と PWM1､4､5､6 の 2 つのシリーズがあります。

同様に APWM ブロックには位相シフトのための 2 つのシリーズがあります。それは PWM1､APWM1､2､3 と PWM1､APWM4､5､6 です。

位相シフトのための PWM ブロックの定義は次のようになります。

－基準 PWM と位相シフトされる PWM は同じシリーズでなければなりません。すなわち PWM1 が基準、PWM2、3 か PWM4､5､6 が PWM1 に関して位相シフトできます。または、PWM2 が基準となる場合は PWM3 が PWM2 について位相シフトできます。

同様に PWM4 (または 5)は基準となり、PWM5 (または 6)は PWM4(または 5)に関して位相シフトできます。しかし、PWM4､5､6 に対して PWM2 または 3 を基準として使うことはできません。

これはまた APWM についても同様です。例えば PWM1 が基準となる場合、APWM1､2､3 は位相シフトされます。同様に PWM1 が基準の場合､APWM4､5､6 が位相シフトできます。または､APWM4 が基準の場合、APWM5 と 6 は位相シフトできます。

－基準 PWM と位相シフトされる PWM は省略された PWM が使用されないようにシリーズの中で連続でなければなりません。例えばもし PWM1、2、3 すべてが使用されているが PWM2 が PWM1 と 3 と同期していない場合は使用できません。しかしもし PWM2 が回路中で使用されていない場合は PWM1 を基準、位相シフト PWM3 として使用することができます。

例えば正しい設定は次のようになります。最初が基準でその後のブロックが位相シフトとなります。

PWM 1 (reference), PWM 2, PWM 3, PWM 4, PWM 5, PWM 6, PWM 7

PWM 2 (reference), PWM 3

PWM 4 (reference), PWM 5

PWM 5 (reference), PWM 6

PWM 6 (reference), PWM 7

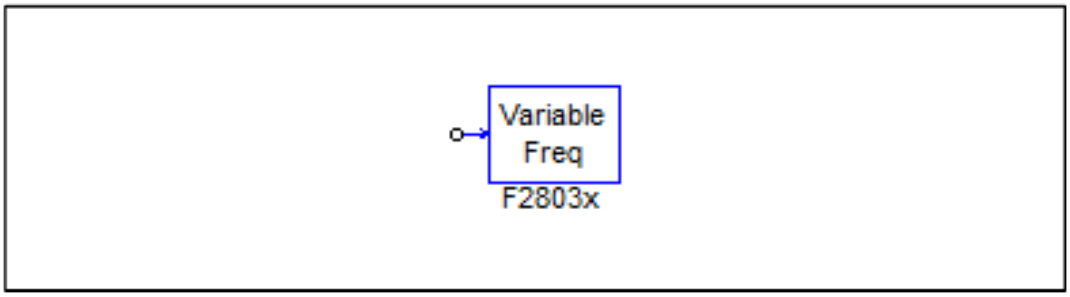
PWM 1 (reference), PWM 2, PWM 3, PWM 4, PWM 5, PWM 6, PWM 7, APWM 1

PWM 1 (reference), APWM 1

8.5 可変周波数 PWM

可変周波数 PWM ブロックでは PWM 生成器のサンプリング周波数の変更ができます。

シンボル：



仕様

パラメータ	機能
PWM ソース	PWM 生成器のソース PWW1～7、3 相 PWM123 と PWM456 を選択できます。
割り込み位置調整	PWM 周波数が調整する時、割込み位置を調整するかどうかを指定します。: - *Do not adjust(調整しない)*: 割込み位置はベース周波数で計算された初期位置を維持します。 - *Adjust(調整する)*: 割り込み位置は、新しい周波数に基づき再計算されます、次の PWM 周期で適用されます。
ランプ補正調整	PWM 周波数を調整する時、コンパレータ DAC ブロックのランプ補償を調整するかどうかを指定します。。 - *Do not adjust(調整しない)*: ランプ補正はベース周波数で計算された初期補正値を維持します。 - *Adjust(調整する)*: ランプ補正は新しい周波数に基づき再計算されます。次の PWM 周期で適用されます。

PWM ブロックと一致するサンプリング周波数は次のような PWM 周期の最初に変更されます。

PWM_Frequency=PWM_Base　Frequency／Input_Value

ここで *PWM_Base_Frequency* は PWM_block に一致したサンプリング周波数です。*Input_Value* はこのブロックの入力値です。

割り込み位置を調整する場合、各サイクルで割込み位置が再計算されます。割込み位置調整は時間がかかるので周波数変化が小さい場合はこの割り込み位置の調整は行わないでください。

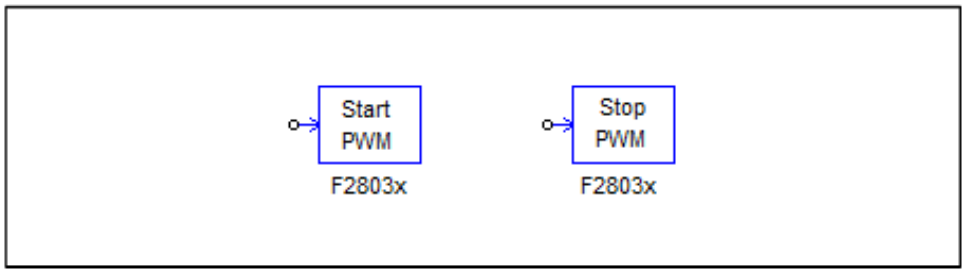
同様にランプ補正を調整する場合、各サイクルで割込み位置が再計算されます。ランプ補正の調整には時間がかかるので周波数変化

が小さい場合はこのランプ補正の調整は行わないでください。

8.6　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 ソース	PWM生成器の出力の選択。 PWM1~7、3-phase PWM123と3-phase PWM456、そしてCapture 1が選択できます。

8.7 トリップゾーンとトリップゾーンステート

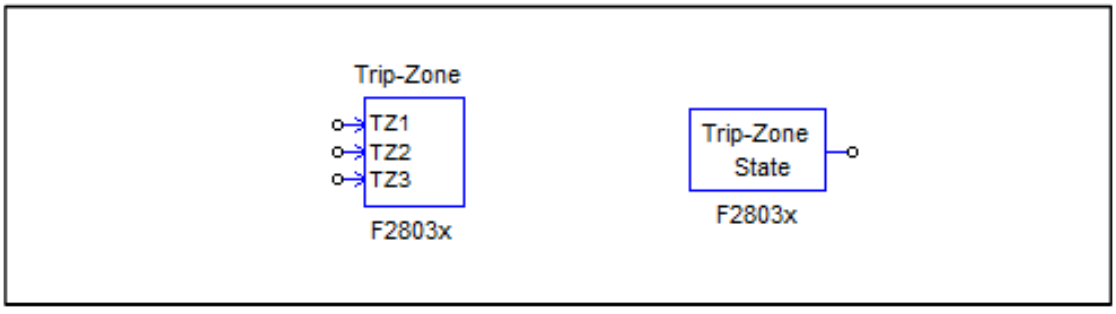
F2803xDSP は 6 つのトリップゾーン入力信号があり、GPIO ポートが 3 つ、コンパレータが 3 つとなっています。コンパレータはトリップ PWM へのデジタル比較器にしか使用できません。

トリップゾーンは外部障害またはトリップ状態を処理するために使用されます。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。

一つのトリップゾーン入力信号は複数の PWM 生成器に使用でき、一つの PWM 生成器は 6 つまでのトリップゾーン入力信号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。

三つの入 GPIO トリップゾーン信号がロー（0）になったとき、トリップ動作をトリガします。デジタル比較器で使うトリップ·ゾーン信号は、指定された（高または低）アクティブレベルでのトリップ動作をトリガします。

シンボル:



仕様（トリップゾーン）:

パラメータ	機能
トリップゾーン*i*使用	トリップゾーンi の使用を定義します。

トリップゾーン*i*用のGPIOポート	トリップゾーン入力信号のGPIOポートを指定します。 トリップゾーン1に対するGPIOポート：GPIO12かGPIO15を選択します。 トリップゾーン2に対するGPIOポート：GPIO13、GPIO16かGPIO28を選択します。 トリップゾーン3に対するGPIOポート：GPIO14かGPIO17かGPIO29 を選択します。
コンパレータ*i*使用	コンパレータ1～3がトリップ·ゾーン入力信号*i*として使用されることを指定します。

仕様（トリップゾーンステート）：

パラメータ	機能
PWM ソース	PWMソースの指定。 - PWM1~7、3-phase PWM123と3-phase PWM456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通りに One-Shot モードまたは Cycle-by-Cycle モードで使用できます。

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。

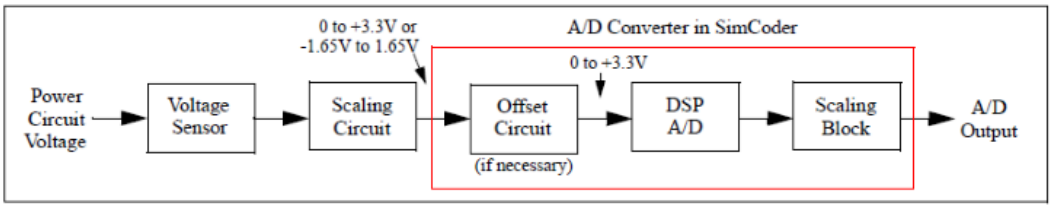
PWM 生成器から割り込みが発生させられたとき、トリップゾーン State ブロックはそれが One-Shot モード、Cycle-by-Cycle モードどちらの Trip 信号なのかを出力します。1 のとき One-Shot モード、0 のとき Cycle-by-Cycle モードとなります。

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。）

8.8 A/D 変換器

F2803x には 12 ビット 16 チャンネルの A/D 変換器があります。

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~+3.3V）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）

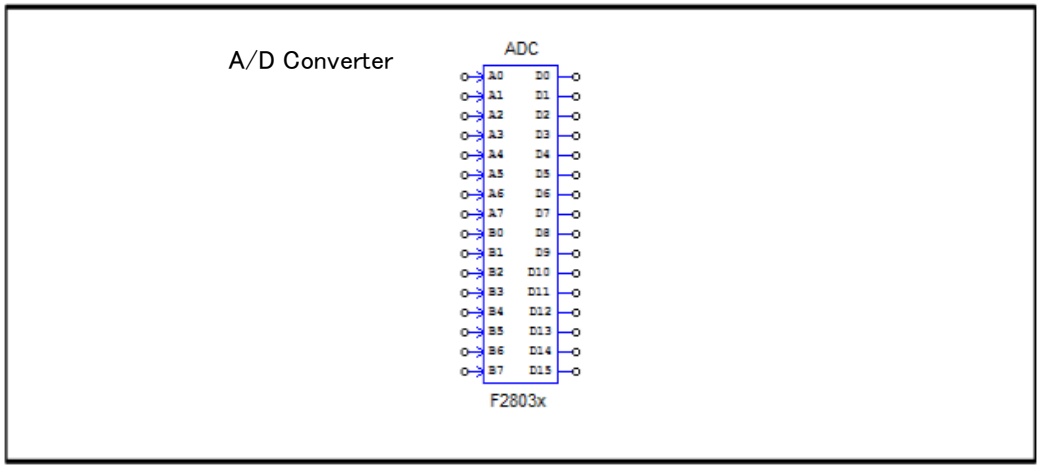


図より分かるように、SimCoder の A/D 変換器は、厳密には実際の DSP 搭載 A/D 変換器と等価ではなく、オフセット機能、A/D 変換機能、スケーリング機能の組み合わせの構成を持っています。これは AC システムアプリケーションが便利に使えるようデザインされています。

多くのアプリケーションで回路変数、特に AC モニタードライブシステムでは AC 信号でモニターされます。これら AC 信号は信号レベルを許容範囲 0 から+3.3V とするために DSP アナログインプットの入力でオフセット回路が回路基板のハードウェアに組み込まれなければなりません。SimCoder の A/D 変換器はそのような場合に便利に使えます。A/D 出力信号のレベルシフトやスケーリングの代わりに SimCoder A／D 変換器のオフセットオプションとスケーリングファクター、の使用を選択するとそれに伴いターゲットコードが生成されます。

※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
Ch A*i* or B*i* モード	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。 - AC：このオプションはシミュレーションのためのみです。-1.65Vから+1.65Vの入力が可能なac入力になります。このオプションはA/D変換へのオフセット回路を含んでいます。AC信号を0から3.3Vとする外部シフタが必要な場合に便利です。 - DC：0Vから+3.3Vの入力が可能なdc入力になります。

Ch A*i* or B*i* ゲイン	各A/D変換チャンネルAiとBiのゲインkの設定。
Conversion Order	A/D変換器の順番となります。もし定義せずに空欄とした場合は変換はA/Dチャネルの番号に沿って実行されます。例えばA0,A2, A4, B1とB3が使用されている場合A0,A2, A4, B1とB3の順に実行されます。もし特定のチャネルを先に実行したい場合、例えばA4,A0,A2,B3,Bとしたい場合はこの順番で設定するとA4はA0,A2の前に実行されるようになります。
ADCINT1 PIE 選択	ADCINT1割り込みが*PIEグループ1*或いは*PIEグループ10*を指定します。
ADCINT2 PIE 選択	ADCINT2割り込みが*PIEグループ1*或いは*PIEグループ10*を指定します。

A/D 変換器は 16 チャネルまであります。SimCoder はそれらのサンプリングレートにしたがってグループに分けられます。最高のサンプリングレートのグループは ADCINT1 割り込みに割り当てられ、2 番目は ADCINT2 割り込みといったように割り当てられていきます。最高サンプリングレートを持つ 2 つの ADC グループは、PIE（周辺割り込み拡張）ベクトルを選択することができます。PIE グループ 1 は PIE グループ１０よりも高い割り込み優先順位を持ちます。 例えば、PWM 割り込みが PIE グループ 3 である場合その割り込み優先順位はグループ 10 より高く PIE グループ 1 よりも低くなります。ユーザは、PWM 割り込み優先順位をある ADC グループの割り込みよりも高く設定したい場合、この ADC グループを PIE グループ 10 に設定する必要があります。

トリガソース：

A/D 変換器は複数のソースからトリガをかけられます。複数の A/D 変換チャネルは同じとりがソースを共用できます。各々の A/D チャネルは次のものからトリガをかけることができます。

- PWM 生成器の 1 つ
- Timer1 か Timer2
- 1 つ以上のトリガソース

回路図でもし A./D チャネルが PWM 生成器と関連していない場合は SimCoder がトリガソースとしてタイマーを選択するよう ZOH ブロックを A/D チャネルの出力に入れなければなりません。

1 つのソースによってトリガされる A/D 変換器チャネルはできません。しかし出力信号は違ったサンプリングレートをもつ回路の一部で使用されます。

出力スケーリング：

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。

入力オフセットとスケーリング：

A/D 変換器の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+3.3V 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±1.65V をピークとしてスケーリングされます。

A/D 変換器の例として DC 入力と AC 入力の 2 つの例を示します。

A/D 変換器チャネル 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 Vi_min＝0V から Vi_max=150V までの値を取ります。A/D 変換機は DC に設定され入力のレンジは 0 から 3.3V となります。実際に入力値 Vi=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 × 0.01 = 1.5V

V_{i_s} = 100 × 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 3.3V なのでゲイン 2.2 を追加します。（回路としては 0.01×2.2=0.022 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なります。

$V_{i\_max\_s\_c}$ = 1.5 × 2.2 = 3.3V

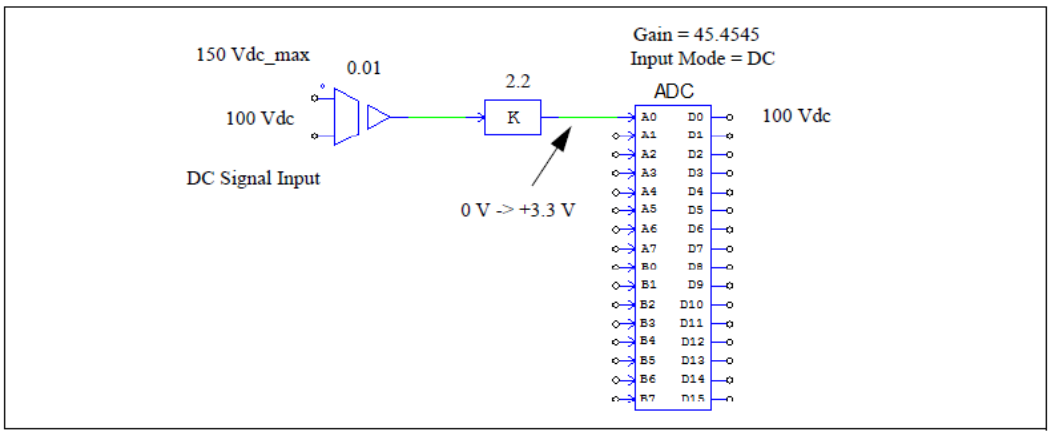
$V_{i\_s\_c}$ = 1 × 2.2 = 2.2V

DSP A/D 変換後のスケー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 45.4545（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} = 45.4545 × 3.3 = 150V

V_o = 45.4545 × 2.2 = 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 2.2 から 1.1、A/D 変換器のゲインを 45.4545 から 90.909 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ない可能性がある点をご注意ください。コードは+3.3V が入力の最大値としてスケーリングしますが、この場合、1.5V が最大値になります。入力値の最大値が+3.3V になるようにスケー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 Vimax=±75V までの値を取るとして、実際に入力値±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±0.75V が来ますが、DSP 入力可能ピーク値は±1.65V なのでゲイン 2.2(1.65/0.75=2.2)を追加します。調整後の入力値は下記のようになります。

$V_{i\_max\_s\_c}$ = +/-1.65V

$V_{i\_s\_c}$ = +/-1.1V

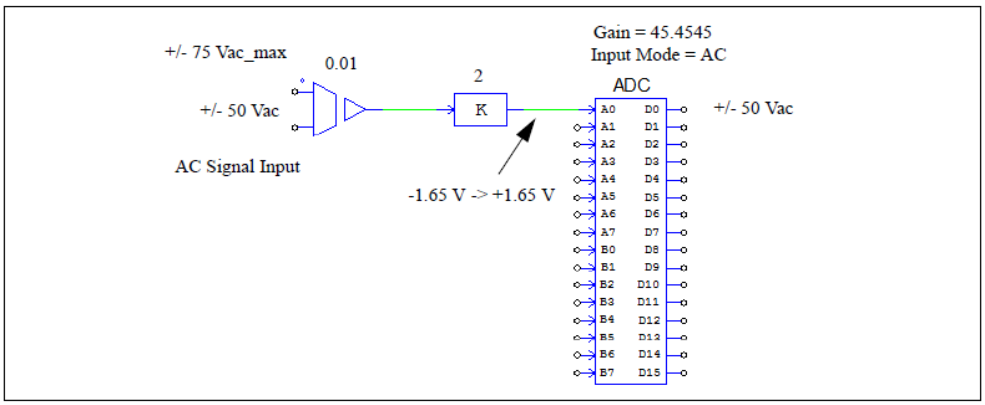
A/D 変換後のスケーリングブロックの出力は、実際に主回路から来た電圧と合わせるとすると、A/D 変換器のゲインは 45.4545（電圧センサと調整用のゲインの逆数）になります。結果として A/D 変換器からの出力は下記になります。

V_{o_max} = +/-75V

V_o = +/-50V

PSIM の A/D チャネルのゲインは 45.4545 に設定されます。

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、A/D 入力モードが AC に設定された時、入力範囲は±1.65V となり、DC オフセットを実行する調整回路の機能は

A/D 変換ブロックにすでに含まれています。実際のハードウェアの回路では AC 信号は A/D 変換器からの要求により 0 から 3.3V 範囲内の値となるため AC 信号はスケーリング及びオフセットされます。

8.9 コンパレータ

F2803x は 3 つのコンパレータモジュールがあります。各々のコンパレータブロックは 2 つの外部アナログ入力か、1 つの外部アナログ入力と他の入力となる内部の DAC リファレンスに使用されています。コンパレータ出力は PWM トリップゾーンと GPIO 出力となります。

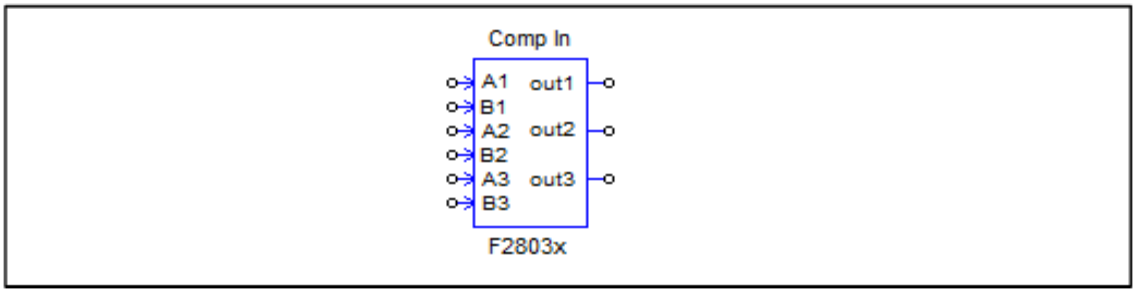
8.9.1 コンパレータ入力

F2803x では 3 つのコンパレータがあります。入力の 3 つのペアは次のような ADC 入力チャネルをもつ ADC/AIP ポートを共用しています。

- Comparator 1 input A - Port ADCA2/AIO2
- Comparator 1 input B - Port ADCB2/AIO10
- Comparator 2 input A - Port ADCA4/AIO4
- Comparator 2 input B - Port ADCB4/AIO12
- Comparator 3 input A - Port ADCA6/AIO6
- Comparator 3 input B - Port ADCB6/AIO14

各々のポートに対して 1 つの機能が対応しています。SimCoder はポートがコンパレーター入力として定義されているにもかかわらず、同じ PSIM 回路図の A/D 変換器のチャネルや AIO として使用されるならば SimCoder はエラーを報告します。

シンボル:



仕様:

パラメータ	機能
コンパレータ *i*	i番目のコンパレータの出力方法の設定。 - *Do not use*: 使用しません。 - *As a normal comparator*: 標準のコンパレータとして使用します。 - *As a Trip-Zone signal*: trip-zone信号として使用します。
出力論理 *i*	コンパレータ出力ロジックの設定。 - *High when A > B*: 入力AがBより大きい時出力はhigh. - *High when A < B*: 入力AがBより小さい時出力はhigh.

比較方法 *i*	コンパレータ入力Aと比較する入力Bの設定。 - *Compare to Input B*: 入力Aを入力Bと比較します。入力Bは外部アナログ信号です。 - *Compare to Constant Value*: 入力Aを定数と比較します。 - *Compare to DAC output*: 入力Aを内部信号のDAC出力と比較します。
定数 *i*	比較方法で定数と比較を選択した場合に有効となります。 定数の範囲は0から3.3Vまでとなります。

すべてのコンパレーターで入力 A は常に外部アナログ入力となります。

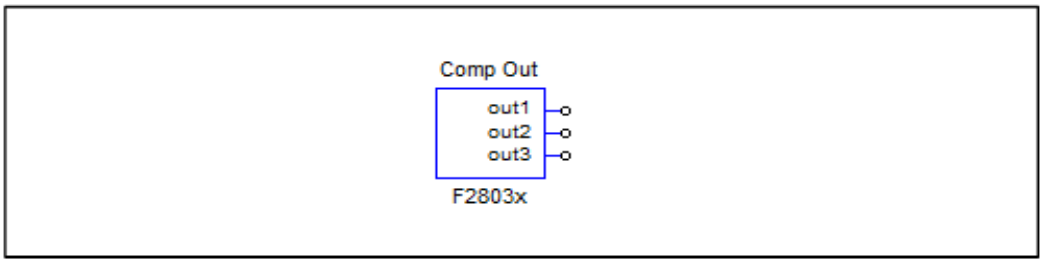
もし入力 B も外部アナログ入力の場合（パラメータの *Compare Method* が *Compare to Input B* として設定される場合），一致するポートは Hardware Configuration block のコンパレータ入力として設定されなければなりません。

もし、定数値と比較する場合、入力 B は回路図でグランドへ接続する必要があります。DAC 出力と比較する場合は入力 B は DAC 出力と接続されます。どちらの場合も対応した ADC/AIP ポートは他の機能に使用することができます。

8.9.2 コンパレータ出力

コンパレータ出力は GPIO 出力と同様に PWMTrip-zone 信号として使用できます。

シンボル:



仕様:

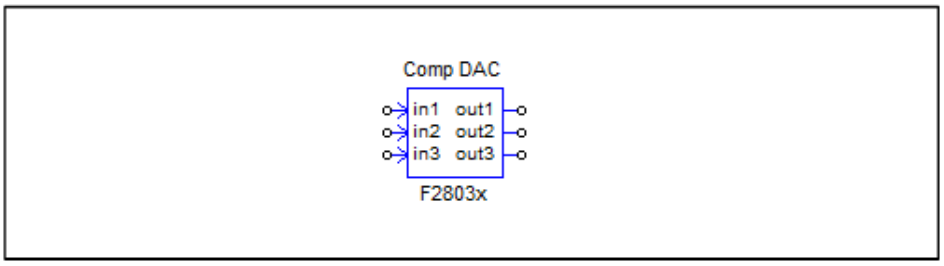
パラメータ	機能
コンパレータ*i*　出力ポート	i番目のコンパレータの出力ポート位置。 - コンパレータ1の場合: GPIO1, 20, or 42 - コンパレータ2の場合: GPIO3, 21, 34, or 43 - コンパレータ3の場合: GPIO34

コンパレータ出力ブロックは入力ブロックと一緒に使用されなければなりません。コンパレータ出力チャネルが使用されると対応するコンパレータ入力チャネルも設定されます。

8.9.3 コンパレータ DAC

F2803x の各コンパレータブロックはコンパレータの反転入力(入力 B)により使用可能な 10bit の DAC コンバータを持っています。

シンボル:



仕様:

パラメータ	機能
DAC i レンジ	i番目のコンパレータDACの入力信号範囲：下限値は0です。
Use Ramp Generator	ランプジェネレーターを使用します。 ランプジェネレーターはコンパレータDACがランプジェネレータ用に設定されます。
Total Ramp Compensation	1つのPWM周期におけるランプジェネレーターの総補正値。1サイクルでのランプ総減少を表わしています。

コンパレータ DAC の出力はコンパレータ入力ブロックの相応反転入力(InputB)に接続できます。ノード out1 はコンパレータ入力ブロックのノード B1 に、ノード out2 はノード B2 へノード out3 はノード B3 へのみ接続できます。

コンパレータ DAC ブロックはコンパレータ入力ブロックと連携して使用してください。単独では使用できません。

ランプジェネレーター機能を使用しない場合､入力値は直接 DAC 出力へ瞬時に適用されます。出力範囲は 0 から 3.3V で出力は次のように計算されます。

DAC 出力信号＝DAC 入力信号＊3.3/DAC レンジランプジェネレーター機能を使用する場合、入力値は保存され次の PWM 周期の初期出力値として使用されます。コンパレータ DAC 出力は PWM 期間内に直線的に減少し、総減少量は総ランプ補償値と同等となります

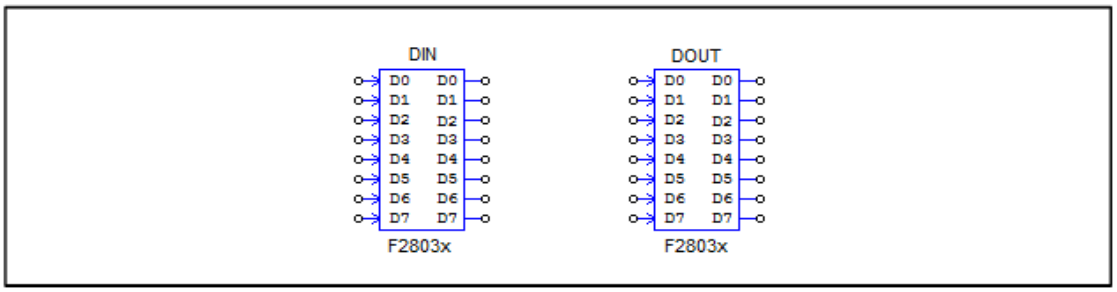
ランプジェネレーター機能を使用する場合、コンパレータ DAC 入力に関連したサンプリングレートはコンパレータに使用されている PWM generator の周波数と同じでなければなりません。

8.10 デジタル入力とデジタル出力

F2803x は 45 本の汎用入出力ポート（GPIO0～44）をデジタル入力、またはデジタル出力に設定可能です。6 本のデジタル入力と出力に使用できるデジタル･アナログ入出力ポート（AIO2,4,6,10,12 と 14）があります。

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。複数の 8 チャンネルのデジタル入出力ブロックは同じ回路上で使用できます

シンボル:



仕様（デジタル入力）：

パラメータ	機能
入力ポート *i* の位置	入力iのポート位置で、iは0~7です。 GPIOポート0～44かAIOポート2,4,6,10,12,14から選択できます。
外部割込みとして使用	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
出力ポート *i* の位置	出力ポートiのポート位置（0~7）です。GPIOポート0～44かAIOポート2,4,6,10,12,14から選択できます。

※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。もし IO ポートがデジタル入力として使用される場合、他のデジタル出力やリフェラルのポートとしては使用できません。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すが、同時に設定した場合、エラーが出力されます。

F2830xDSP は 3 つのマスクされた外部割込み(XINT1～XINT3)をサポートしています。外部割込みの専用のピンはありません。X XINT1～XINT3 の割り込みは GPIO0～31 ピンの入力として使用できます。

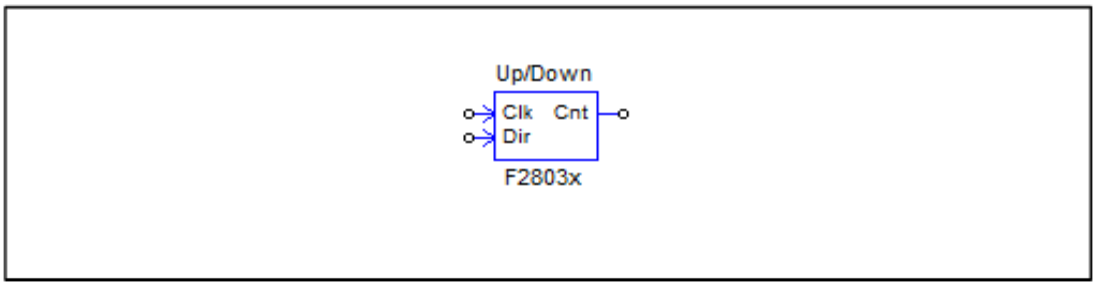
8.11 UP/DOWN カウンタ

F2803x は UP/DOWN カウンタを持っています。入力ポートは

-GPIO20 はクロック信号を指定します。

-GPIO21 はカウント方向を指定します。

シンボル:



仕様:

“Clk”はクロック信号入力、“Dir”はカウンタがどちらへ進むかを指定します。“Dir”が 1 の時カウンタは増加し 0 の時は減少します。

UP/DOWN カウンタブロックの出力はカウンタ値となります。

UP/DOWN カウンタはエンコーダと同じリソースを使用しているたえ同じ GPIO ポートは使用できません。例えば UP/DOWN カウンタ 1 とエンコーダ 1 は同時に使用できません。

8.12 エンコーダとエンコーダステート

F2803x はエンコーダモジュールを利用できます。

－GPIO 20 クロック入力

－GPIO 20 方向入力

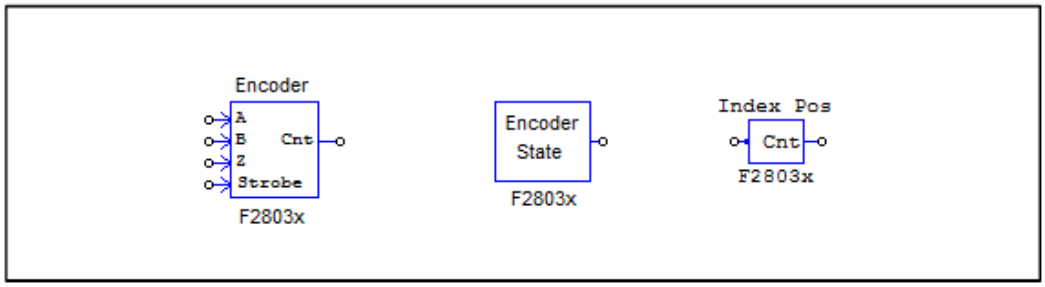
－GPIO 20 Z(もしくは基準信号)入力

－GPIO 20 ストローブ信号入力

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る設定なのかを見ることに使います。ハードウェアの割り込みは Z 信号（基準信号）とストローブ信号から生成され、エンコーダステートの出力はどの信号が割り込み発生させているかを示しています。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。

エンコーダインデックス/ストロボ位置ブロックはエンコーダの初期位置をラッチするのに使用されます。この素子の入力が 0 の時、エンコーダカウンタは 0 に設定されます。入力が 1 になるとエンコーダは、インデックス/ストロボイベントの発生を待ち、発生するとエンコーダのカウンタ値をラッチします。

シンボル:



仕様（エンコーダ）：

パラメータ	機能
Z信号使用	Z信号（基準信号）の有効無効設定。 - *No:* 使用しない。 - *Yes(rising edge):* 立上り信号エッジを使用します。 - *Yes(falling edge):* 立下り信号エッジを使用します。
ストローブ信号使用	ストローブ信号の有効無効設定。 - *No:* 使用しない。 - *Yes(rising edge):* 立ち上がり信号エッジを使用します。 - *Yes(rising/falling edge):*立ち上がり立下り両方の信号エッジを使用します。
カウンタ方向	カウンタの方向の設定。 - *Forward*：カウンタは増加方向に進みます。 - *Reverse*：カウンタは減少方向に進みます。
Z 信号極性	Z信号のトリガの極性 - *Active High*：正論理 - *Active Low*：負論理
ストローブ信号極性	ストローブのトリガ極性 - *Active High*：正論理 - *Active Low*：負論理
エンコーダ分解能	接続するエンコーダの分解能の設定。 0に設定した場合、リセットされません。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にリセットされます。

仕様（エンコーダインデックス／ストロボ位置）：

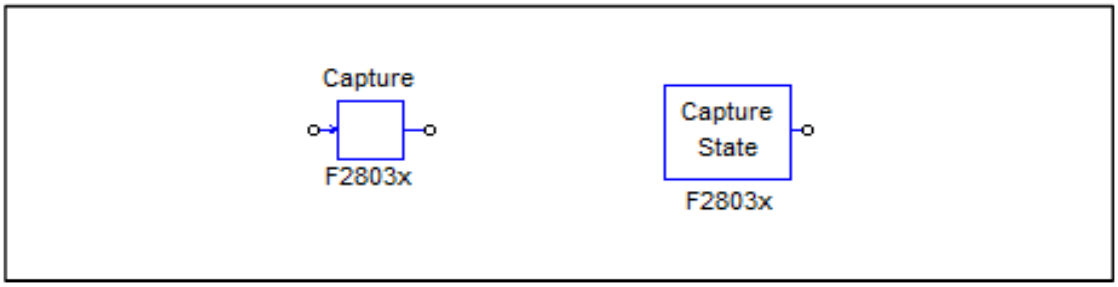
パラメータ	機能
ラッチ位置	ラッチカウンタータイプの選択 - Index Pos:Z方向信号が使用されます。 - Strobe Pos:Strobe信号が使用されます。
位置タイプ	ラッチ位置の選択 - 最初のラッチ位置 - 現在のラッチ位置

8.13 キャプチャとキャプチャステート

F2803x は強化されたキャプチャモジュールを持っています。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。

シンボル:



仕様（キャプチャ）：

パラメータ	機能
キャプチャ ソース	キャプチャとポートの設定。次の3つのGPIOポートの1つをソースとして設定できます。 - Capture1 (GPIO5) - Capture2 (GPIO19) - Capture3 (GPIO24)
イベント フィルタ	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。
タイマモード	キャプチャカウンタタイマモード設定。 Absolute time(絶対時間)またはTime differene(時間差)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
キャプチャ ソース	キャプチャソースです。 Capture1が利用可能な唯一のソースです。

キャプチャステートブロック出力は 1 か 0 で、1 は立ち上がり信号エッジ、0 は立ち下がり信号エッジとなります。

8.14　シリアル通信インターフェース（SCI）

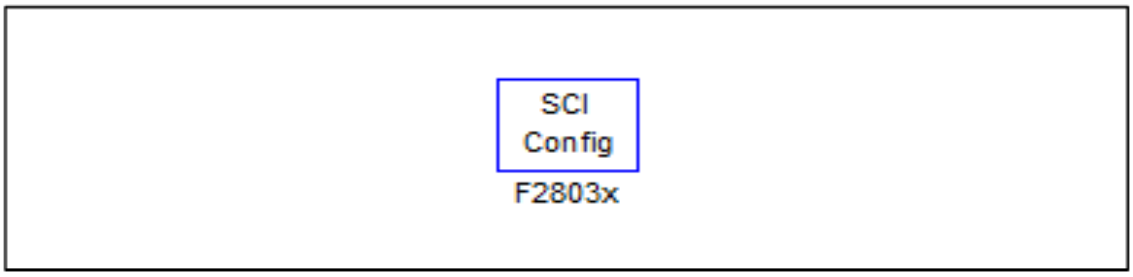
F2803x は、シリアルコミュニケーションインタフェース（SCI）のための関数が用意されています。SCI を介して、DSP 内部のデータは、外部 RS-232 ケーブルを使用してコンピュータに転送することができます。PSIM は、DSP とコンピュータ両方のデータの送受信およびコンピュータ上にデータを表示するための必要なすべての関数を提供します。これは DSP のコードをリアルタイムでモニタ、デバッグ、および調整するために非常に便利な方法を提供します。

以下に説明しますが、SimCoder では SCI 設定、SCI 入力、および SCI 出力の 3 つの SCI の機能ブロックが用意されています：

8.14.1 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
ポート選択	SCIポートの設定。SCIに使用できる次のGPIOポートがあります。 - SCIA：GPIO28とGPIO7に対する組み合わせとしてGPIO29とGPIO12があります。
通信速度(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。
パリティチェック	通信エラーチェック用のパリティチェックの設定。 None(無効)、Odd(奇数)、Even(偶数)のいずれかを選択できます。
出力バッファサイズ	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てはいけません。

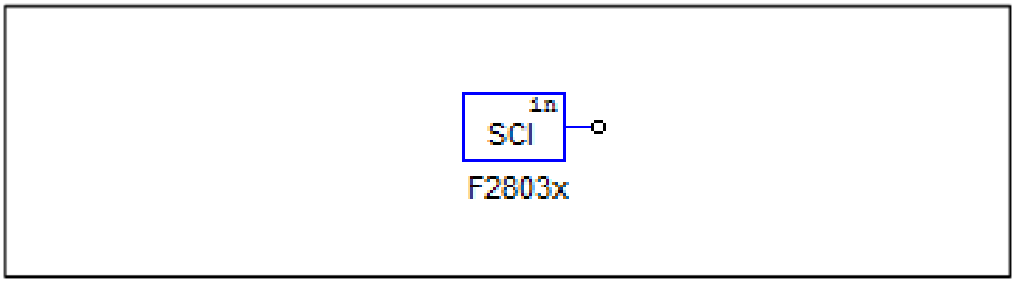
バッファサイズを選択する方法の詳細については、"Tutorial - Using SCI for Waveform Monitoring.pdf"を参照してください。

8.14.2 SCI 入力

SCI 入力ブロックは、変更可能な 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あります）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、リファレンスの変更、制御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
初期値	SCI入力変数の初期値。

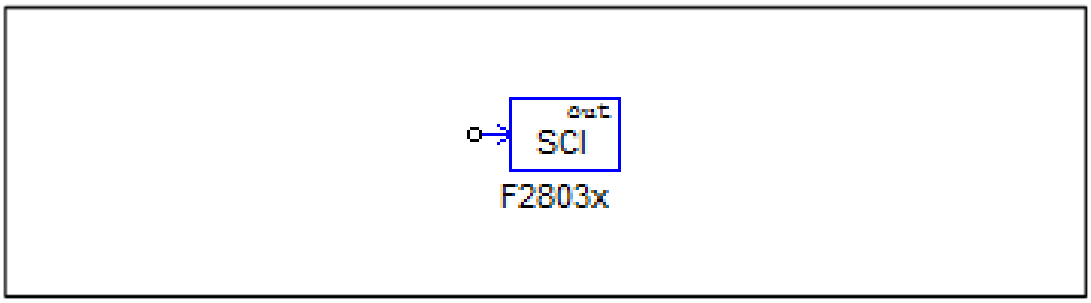
回路図では、SCI 入力は定数として動作します。コードは DSP 上で実行されている間に、ユーザーはこの値を変更することができ、値はシミュレーションの初期値として固定されます。

8.14.3 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あります。）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
データポイントステップ	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ポイントのうちの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 オシロスコープが連続モードの時のみ使用されます。スナップショットモードにある場合、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。

8.15 シリアルペリフェラルインターフェース(SPI)

F2803x 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 F803x 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。

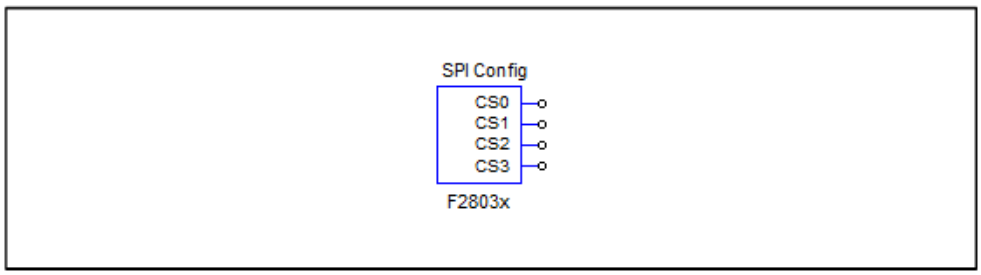
SPI ブロックを使用する方法についての詳細な説明については、ドキュメントを参照してください*" Tutorial - Using SCI for Real-Time Monitoring.pdf"*。

SimCoder では SPI 設定、SPI デバイス、SPI 入力、SPI 出力の 4 つの SCI の機能ブロックが提供されています。

8.15.1 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:

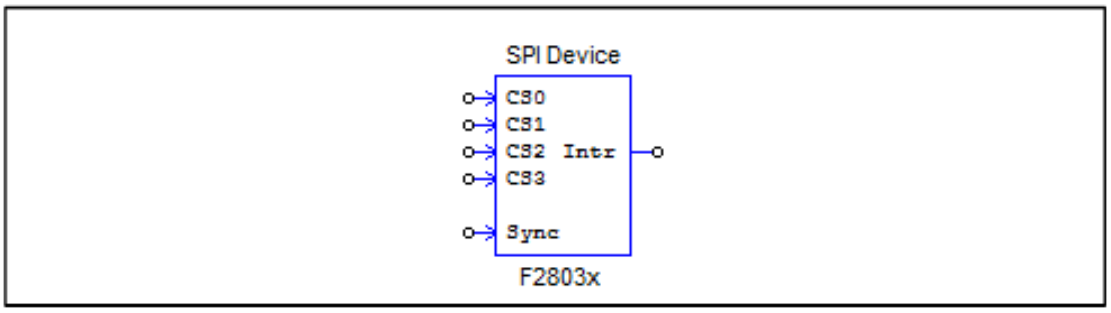
パラメータ	機能
SPI ポート選択	SPIポートの設定。 -SPIA (GPIO 16-19) -SPIA (GPIO 3,5,18,19) -SPIB (GPIO 12-15) -SPIB (GPIO 24-27)
チップセレクト	チップセレクトピンのGPIOポート。PSIMは、最大で16のSPIデバイスを使用する

端子 0,1,2,3	ことができます。これら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ります。 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。
キューサイズ (SPI Buffer Size)	SPI通信バッファサイズ。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。

8.15.2 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
チップセレクト端子	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設定と一致する場合にこのSPIデバイスが選択されます。
通信速度（MHz)	SPIの通信速度（MHz）
クロックタイプ	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。 -立ち上がりエッジ（遅れなし）： クロックは通常はLOW(0)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。 -立ち上がりエッジ（遅れあり）： クロックは通常はLOW(0)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。 -立ち下がりエッジ（遅れなし）： クロックは通常はHigh(1)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。 -立ち下がりエッジ（遅れあり）： クロックは通常はHigh(1)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。
コマンドワード長	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。
同期信号タイプ	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 初期コマンド	SPIデバイスの初期コマンドです。
ハードウェア割り込みモード	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。 - ハードウェア割り込みなし - 立ち上がりエッジ - 立ち下がりエッジ
割込みタイミング	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。 - *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 - *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ネルの新しい変換を開始します。 - *ワンタイム割り込み* : 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。
コマンド間ギャップ (ns)	2つのSPIコマンドの間隔で単位[nsec]です。
変換シーケンス	変換シーケンスを決定するためのコンマで区切られた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。

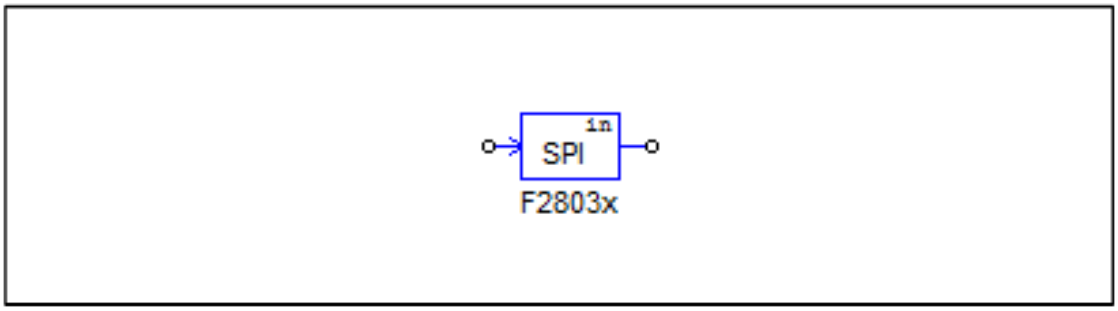
同期信号を使用した SPI の入力デバイスは、通常、割り込みサービスルーチンを入力して DSP をトリガする割り込みピンを必要とします。

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。

8.15.3 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 通信用の入力チャネル、および 1 つの入力チャネルに対応する 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
デバイス名	SPI入力デバイスの名前です。
変換開始コマンド	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)
データ受信コマンド	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)
データビット位置	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は、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。
入力範囲	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。
スケール係数	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リングされます。
AD変換モード	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。
初期値	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 $Output = \frac{Result \cdot V_{max} \cdot K_{scale}}{2^{Data\_Length}}$

AC 変換モードの時：

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

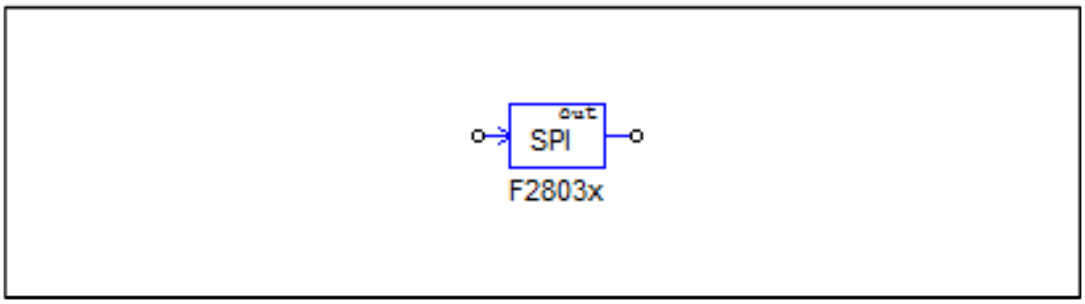
-実際のハードの場合 $Output = \frac{(Result - 2^{Data\_Length-1}) \cdot V_{max} \cdot K_{scale}}{2^{Data\_Length-1}}$

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。

8.15.4 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 通信用の出力チャネル、1 つの出力チャネルに対応する 1 つの SPI 出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
デバイス名	SPI出力デバイスの名前です。
スケール係数	出力スケール係数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リングされます。
出力範囲	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 モード	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。
データ送信コマンド	カンマで区切られた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)
データビット位置	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。 *素子名(ElementName) = {Xn[MSB.. LSB]}* ここで、 - *"素子名"*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。 - {}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。 - *MSB.. LSB*は、ワードの有効ビットの位置を定義します。
同期コマンド	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ネルと同期させるコマンドです。(例；0x23、0x43、0x00) 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。

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 $Output = \frac{Result \cdot K_{scale} \cdot 2^{Data\_Length}}{V_{max}}$

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 $Output = 2^{Data\_Length} + \frac{Result \cdot K_{scale} \cdot 2^{Data\_Length-1}}{V_{max}}$

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。

8.16 Controller Area Network（CAN）Bus

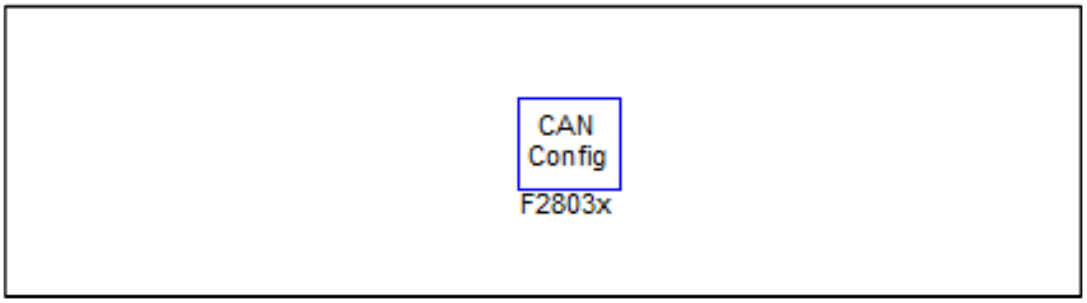
F2803x は CAN バスコミュニケーションの機能をもっています。PSIM では CAN バスでで必要な機能を備えています。SimCoder には *CAN Configuration*, *CAN Input*、*CAN output*

の 3 つの機能があります。

8.16.1 CAN　Configration

　CAN　Configuration ブロックは CAN バスソースでデータバイト順や他の設定を定義します。F2803xDSP には 1 つの CAN ソース CAN A があり、送信用の GPIO31 と受信用の GPIO30 を持っています。

シンボル:



仕様:

パラメータ	機能
CAN 速度 (MHz)	Communication　Speed（単位はMHz）。 次の周波数、125kHz、250kHz、500kHz、1MHzのどれか1つを選択するか もしくは手入力で設定できます。手入力の場合は1MHzを超えないようご注意ください。
データバイト順	データバイトの順番になります。次の 1 つとなります。 - *Least significant byte 1st* - *Most significant byte 1st* “Least significant byte 1st” は最下位バイトが最初に転送され、最上位バイトが最後に転送されます。一方 “Most significant byte 1st”は最上位バイトが最初に転送され、最下位バイトが最後に転送されます。
入力メールボックス数	入力mailboxの数
受信メッセージ紛失検査	もし*Enable*(有効)に設定されると受信メールが無くなった場合、エラーレポート関数の “_ProcCanAErrReport(nErr)” （CAN Aに対して）が呼ばれます。 この関数はエラー状態nErrの数値を返します。 もし、*Disable*(無効)に設定されるとエラーレポート関数は呼ばれません。
バスオフ検査	もし*Enable*(有効)に設定されると受信メールが無くなった場合、エラーレポート関数の “_ProcCanAErrReport(nErr)” （CAN Aに対して）が呼ばれます。 この関数はエラー状態nErrの数値を返します。 もし、*Disable*(無効)に設定されるとエラーレポート関数は呼ばれません。
エラー検査モード	エラーチェックモードを受動か能動かに設定できます。受動の場合、エラーカウントが128になったときに割込みが発生します。能動の場合割り込みが毎回発生します。

関数 “__ProcCanAErrReport(nErr)”（CAN A に対して）の戻り状態 nErr は 32 ビットの整数です。

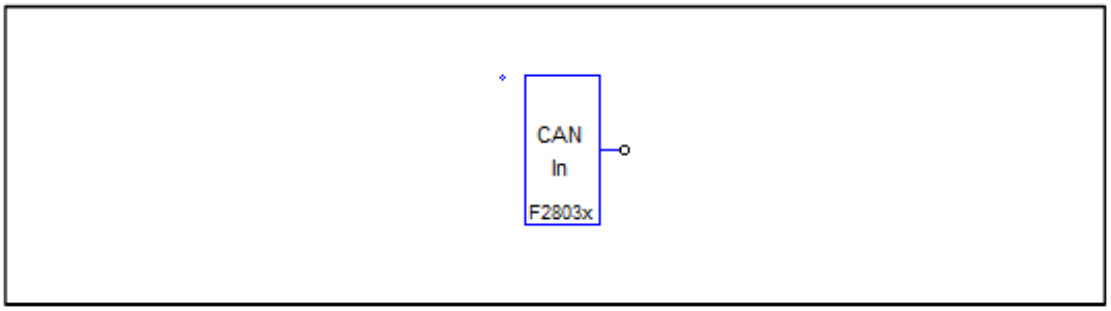
エラーとレジスターCANES 状態からの値を含んでいます。関数 “__ProcCanAErrReport(nErr)”からの値が戻るとレジスターCANES はクリアされます。

もし、特定のエラーに対応したい場合は “__ProcCanAErrReport(nErr)”関数に独自のコードを追加することができます。

8.16.2 CAN Input

CAN 入力ブロックは CAN バスから CAN メッセージを受け取ります。

シンボル:



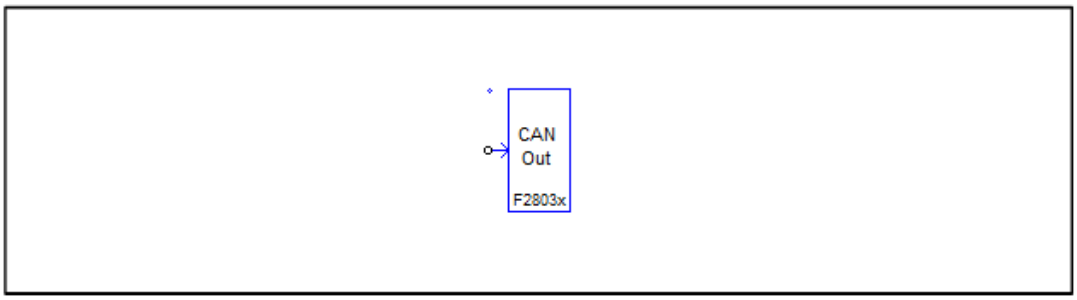
仕様:

パラメータ	機能
入力信号数	CAN入力数です。8入力まで可能です。
拡張ID使用	Yesに設定された場合、メッセージのIDは29-bitの整数となります。 Noに設定された場合、メッセージのIDは11-bitの整数となります。
メッセージID	メッセージのIDです。整数で例えば0x23のようになります。
ローカルマスク	メッセージに対するMASKとなります。整数です。もしメッセージIDのビットがMASKのビットと一致するとメッセージを受信します。そうでない場合は無視されます。例えばmaskが0x380でメッセージIDが0x389の場合、このメッセージが受信されます。もし、メッセージIDが0x480の場合はメッセージは無視されます。
上書き保護フラグ	*Allow overwrite*(上書き可)か*Do not allow overwrite*(上書き不可)に設定できます。 例えばメッセージ受信用にメールボックスが設定され、新しいメッセージがCANバス経由で受信されたが古いメッセージがまだメールボックスにあり、何も処理されていない場合、上書き保護フラグが”Allow overwrite”に設定されると新しいメッセージが受信され古いメッセージは上書きされます。Flagが”Do not allow overwrite”に設定されていると新しいメッセージは受信されずに古いものはそのまま残ります。
受信頻度	特定期間に入力ブロックで受信したメッセージの数。例えば、最初の入力ブロックで1秒間に20のメッセージを、2番目の入力ブロックで1秒間に30のメッセージを受信する2つの入力ブロックがあるとします。その場合” Receive Message Rate”は最初のブロックに20、2番目のブロックに30と設定されます。
入力 *i* ゲイン	i　番目の入力に対するゲインでiは1から8で設定できます。出力は入力のゲインの値をかけた値になります。
入力 *i* データ開始位置	メッセージは8データポイントまで可能です。データポイントは1bitから32bitsまで可能です。これはメッセージの現在のデータポイントの開始位置を指定します。
入力 *i*データ終了位置	これはメッセージの現在のデータポイントの終了位置を指定します。
入力 *i* データタイプ	データタイプは、浮動、整数、またはIQ1からIQ30までとなります。
I入力 *i*デフォルトデータ	SCI入力変数の初期値となります。

8.16.3 CAN Output

CAN 出力ブロックは CAN メッセージを CAN バスへ送ります。

シンボル:



仕様:

パラメータ	機能
出力信号数	CAN出力の数です。8出力まで可能です。
拡張ID使用	Yesに設定された場合、メッセージのIDは29-bitの整数となります。 Noに設定された場合、メッセージのIDは11-bitの整数となります。
メッセージ ID	メッセージのIDです。整数で例えば0x23のようになります。
トリガタイプ	トリガのタイプは次のようになります。 - *No trigger*：トリガなし。 - *Rising edge*：トリガソースの立ち上がりエッジでトリガがかかります。 - *Falling edge*：トリガソースの立下りエッジでトリガがかかります。 - *Riding/Falling edge*:とりがソースの立ち上がりと立下りのエッジでトリガがかかります。 立ち上がり／立下りエッジはトリガソースの現在の値と、最終のトリガの値が1以上で差がある場合に適用されます。
トリガソース	トリガソースはトリガタイプによるグローバル変数の定数にもあります。 トリガタイプが“No Trigger”に設定されるとトリガソースはカウンターリミットを定義します。例えばトリガソースが定数かグローバル変数で値が5の時、5回に１回トリガがかかることを意味しています。すなわちデータが5サイクル毎に送られます。 もしトリガタイプがエッジトリガに設定された場合、トリガソースはグローバル変数のみです。グローバル変数がトリガタイプに依存して立ち上がりか立下りもしくは両方のエッジとなる時にトリガがかかります。
出力 *i* ゲイン	i番目の入力に対するゲインでiは1から8で設定できます。出力は入力のゲインの値をかけた値になります。
出力 *i* 開始位置	メッセージは8データポイントまで可能です。データポイントは1bitから32bitsまで可能です。これはメッセージの現在のデータポイントの開始位置を定義します。
出力 *i* データ終了位置	これはメッセージの現在のデータポイントの終了位置を定義します。
出力 *i*データタイプ	データタイプは、浮動、整数、またはIQ1からIQ30までとなります。

8.17　割込み時間

割込み時間ブロックは割込みサービスルーチンの時間間隔を測定するのに使われます。

仕様:

パラメータ	機能
Time Output Method	割込み時間測定の設定を行います。 -SCI(使用時間): SCIを使用します。DSPクロックカウントで割込みサービスルーチンとして使われる時間はSCI出力を経て送られ測定されます。 -SCI(残り時間): SCIを使用します。DSPクロックカウントの割込みサービスルーチンの残り時間はSCI出力を経て送られ測定されます。残り時間は現在の最後の割込みから次の割り込みの最初までの時間として定義されます。 -GPIO0 to GPIO44又はAIP2 からAIO14 : GPIOポートを使います。パルスは特定されたGPIOポートで生成されます。パルスは割込みに入った時high(1)にセットされます。割込みが終了した時にはlow(0)にセットされます。オシロスコープでパルス幅を測定できます。
サンプリング周波数	割込みサービスルーチンのサンプリング周波数（単位はHz）

SCI が使われる場合、値は DSP クロックでカウントされます。例えば値が 6000 の時　60MHz の DSP クロックの割込み時間は　6000/60M=100 us となります。

8.18　プロジェクト設定とメモリ配置

TI F2803x 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.sch」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 F2803x 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
- DSP2803x_Headers_nonBIOS.cmd　：ペリフェラルレジスタリンカコマンドファイル
- F28035_FLASH_Lnk.cmd　：フラッシュメモリリンカコマンドファイル
- F28035_FLASH_RAM_Lnk.cmd　：フラッシュ RAM リンカコマンドファイル
- F28035_RAM_Lnk.cmd　：RAM メモリリンカコマンドファイル

注）: リンカコマンドファイルの名前は F28035 でない場合はターゲットハードウェアによりアサインされます。例えばもしターゲットハードウェアが F28034 の場合、ファイル名はそれに応じて F28034_ FLASH_Lnk.cmd, F28034_ FLASH_RAMLnk.cmd, F28034RAM Lnk.cmd となります。

更に下記のファイルが必要となります。

- PsBiosRamF03xFixpt.lib　：PSIM フォルダ内の SimCoder F2803x ライブラリ

- PsBiosRomF03xFixpt.lib ：PSIM フォルダ内の SimCoder F2803x ライブラリ
- IQmath.lib ：PSIM の¥lib フォルダ内の TI IQmath.lib
- 2803x_IQmath_BootROMsymbols.lib ：PSIM の¥lib フォルダ内の IQmath のシンボルライブラリ

これらのラブラリ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(例で言うと”test.c”と”test.proj”)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 2 つ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 はリンカコマンドファイル F2803x_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 はリンカコマンドファイル F2803x_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 はリンカコマンドファイル F2803x_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 のコードが最も早いです。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 Flash RAM Release へ移行していきます。

メモリ配置：

生成されたリンカ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：

RAM Memory
0x0000_0x07FF(2K)
Interrupt vectors
stack
0x8000-0x9FFF(8K*)
Program and data space

Flash Release の設定の場合：

RAM Memory
0x0000_0x07FF(2K)
Interrupt vectors
stack
0x8000-0x9FFF(8K*)
data space

Flash Memory
0x3E8000_0x3F7FFF(64K)**
Program
Password
etc.

注意：

＊ RAM メモリでは SimCoder はプログラム空間やデータ空間を定義します。

- F28035,F28034, F28032： 0x8000 から 0x9FFF(8K)
- F28031 ： 0x8000 から 0x97FF(6K)
- F28030 ： 0x8000 から 0x8FFF(4K)

もしプログラムとデータ空間を合わせたものが RAM 空間を越えた場合は Flash Release 設定を選択する必要があります。

＊＊ SimCoder によりあらかじめ定義されているプログラム空間に対する flash memory は

- F28035 と F28034： 0x3E8000 から 0x3F7FFF(64K)
- F28033、F28032、F28031： 0x3F0000 から 0x3F7FFF(32K)
- F28030： 0x3F4000 から 0x3F7FFF(16K)

となります。

9 PE-Expert4 Hardware Target

9.1 概要

SimCoder は Myway プラス製の PE-Expert4 DSP 開発プラットホームのためのコードを生成できます。以下のボードに対応しています。

- DSP ボード MWPE4-C6657、コアDSPボード
- PEVボード MWPE4-PEV、PWM生成ボード
- ADボード MWPE4-ADC、アナログ信号入力拡張ボード

複数のサンプルレートをもつシステムに対するコード生成の場合、SimCoderはPWMサンプリングレートではPWM発生器割込みを使用します。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。

PE-Expert4システムではデジタル入力、エンコーダ、キャプチャがハードウェアの割り込みを生成できます。割込みはこのマニュアルで説明されている割り込みブロックを使って割込みサービスルーチン(割込みサービスルーチンを表示しているサブ回路)に関連づけなければなりません。PE-Expert4ではデジタル入力、エンコーダ、キャプチャ、は同じ割り込みサービスルーチンで扱われるので、すべての割り込みブロックはサブ回路ブロックに接続されていないとなりません。

次に、ハードウェアについて説明します。

9.2 DSP ボード

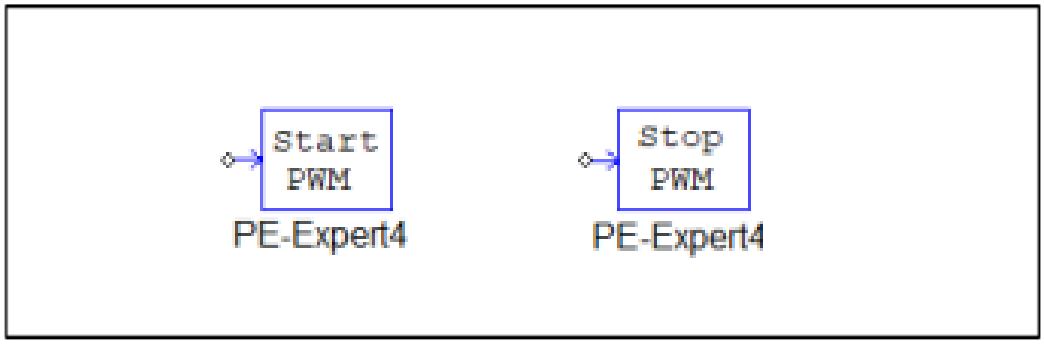
PE-Expert4のDSPボードはコアボードとなります。以下の機能やハードウェア素子が含まれます。

ースタート・ストップPWM
ーLED出力
ータイマー割込み優先

9.2.1 スタート・ストップ PWM

これらのブロックは、PE-Expert4のPWM機能を開始または停止します。

シンボル:



仕様:

パラメータ	機能
ボードタイプ	Expert4ボードのボードタイプは、PEVボードになります。
ボード番号	起動するPWM生成機を含むPEVボードの番号。

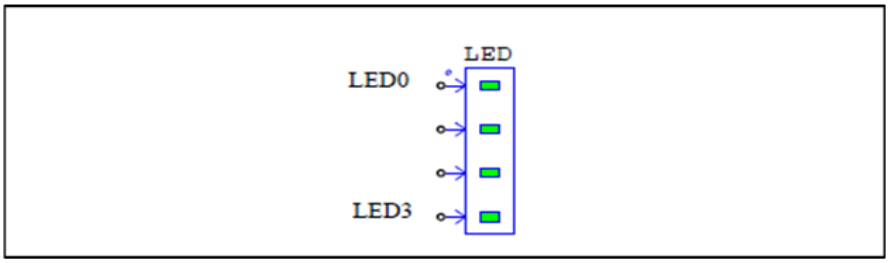
PWMを開始するには、「Start PWM」素子の入力に高論理信号（1）を適用します。PWMを停止するには、「Stop PWM」素子の入力に高論理信号（1）を適用します。

9.2.2 LED 出力

LED 出力素子は DSP ボード(MWPE4-C6657)で提供されています。

DPS ボードには 4 つの LED(LED0,LED1,LED2,LED3)があります。入力が 0.3 以上となった場合に On します。それ以外は Off です。

シンボル:

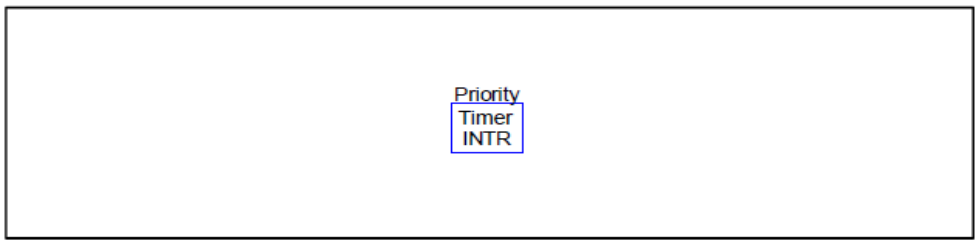


この素子はハードウェア実装のみでシミュレーションでは無視されます。LED 値を表示するためには電圧プローブを入力に接続してください。ﾄﾞｯﾄのある入力が LED0 に対応しています。

9.2.3 タイマー割り込み優先設定

この素子はタイマー割り込み優先順位を設定します。

シンボル:



仕様:

パラメータ	機能
タイマー0の割り込み優先順位	タイマー0の割込み優先を設定。
タイマー1の割り込み優先順位	タイマー1の割込み優先を設定。

この素子はオプションです。システムにタイマー割り込み優先素子がない場合 SimCoder は自動的にタイマ 0 を割込み優先順位 10 に、タイマ 1 を割込み優先順位 11 に設定します。

9.3 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。

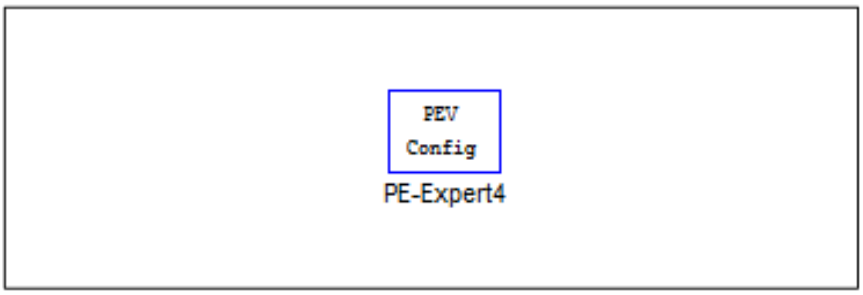
－PWM生成器及びスタート・ストップ機能
－A/Dコンバータ
－デジタル入力／キャプチャ／カウンタ
－デジタル出力
－エンコーダ
－PWMボード構成

PWM発生器、A/D変換器、デジタル入力/出力、アップダウンカウンタ、エンコーダ、キャプチャを含むハードウェア入出力素子はサブ回路内には配置できませんのでご注意ください。主回路のトップレベルにのみ置けます。スタート/ストップPWM関数素子はメイン回路でもサブ回路でも置けます。

9.3.1 PEV ボード設定

この素子は PEV ボードの構成を定義します。PEV ボード素子をのいずれかを PSIM 回路が使用する場合、この設定ブロックを使う必要があります。

シンボル:



仕様:

パラメータ	機能
ボード番号	PEVボードのボード番号(0から4)
イントiの割込みソース	現在のシステムでイントiを使用する割込みソースを特定します。
イントiの割込み優先	現在のシステムでイントiを使用する割込み優先を特定します。

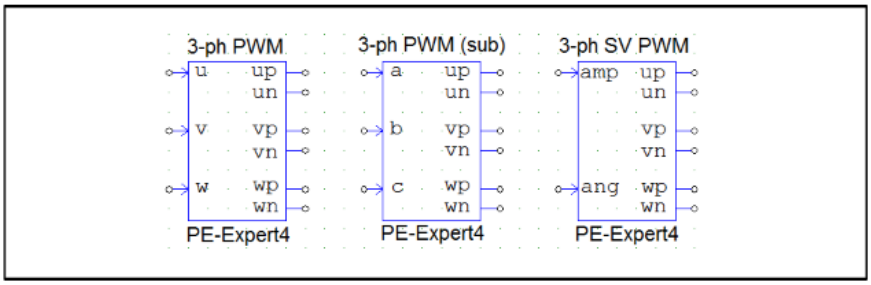
素子は各割り込み信号の割り込みソースとともに割込み優先度も定義します。

ADC/PWM の組み合わせの場合, PWM が ADC をトリガしない場合 PSIM は PWM を割込みソースとして設定するので割込みソースを PWM に設定する必要があります。もし PWM が ADC をトリガする場合、PSIM は割込みソースとして ADC を設定するので ADC を割込みソースに設定する必要があります。

9.3.2 PWM 発生器

PWM信号を発生するために3つの素子が準備されています。PWM発生器、PWM(sub)発生器、空間ベクトルPWM発生器です。PWM(sub)発生器はPWM素子に基づいた内蔵モジュールです。

シンボル:



PWM発生器の仕様:

パラメータ	機能
ボード番号 スケーリングファクタ	PWM発生器を含んでいるPEVボードのボード番号 スケーリングファクタは、1～1000の範囲の整数で、2つのPWMによる割り込み間のPWM周期の数を指定します。
割込みモード	割り込みモードは次の設定があります。 *Interrupt at peak*: 割り込みがPWM搬送波のピーク(山)を基準にします。 *Interrupt at valley*: 割り込みがPWM搬送波のバレー(谷)を基準にします。 *Interrupt at peak/valley*: 割り込みがPWM搬送波のピーク(山)/バレー（谷）の両方を基準にします。

	割込みが山/谷の両方でに設定されている場合、搬送波周波数の2倍で割込みが発生します。搬送波の周波数が10kHzの場合、割り込みレートは20kHzとなります。
割込み位置	割込み位置の範囲は-0.5から+0.5です。値が正の場合PWM周期の開始後です。値が負の場合はPWM周期の開始前となります。 A/D変換器が回路に使用されていてその変換モードが“PWMトリガ”に設定されると、このパラメータはPWMをトリガとしたADC変換完了割り込みのタイミング設定となります。PWMがトリガをかけない場合はPWM割り込みの設定となります。 0の場合　A/D変換器(又はPWM割込み)はPWM周期の最初でトリガされます。 +0.5の場合　A/D変換器(又はPWM割り込み)はPWM半周期でトリガされます。
同期モード	同期モードは次から選択できます。 ・Disable ・Slave(Sync.iを使用) ・Master(Sync.iを使用) iが1～８の場合
デッドタイム	PWM発生器のデッドタイム　単位：秒
制御周波数	PWM発生器の周波数　単位：Hz
最大振幅値	搬送波のピーク間値。 このパラメーターはPWM（sub）専用です。
オフセット値	搬送波のDCオフセット値。このパラメーターはPWM（sub）専用です
U相、V相、W相初期値（またはA相、B相、C相）	PWM生成器の位相u、v、wの初期入力値。またはPWM（sub）発生器の位相a、b、cの初期入力値。
初期振幅	PWM発生器の初期振幅値。空間ベクトルPWM専用です。。
初期角度	PWM発生器の初期角度　単位：ラジアン　空間ベクトルPWMのみ。
PWM信号出力の初期状態	「Start」を設定すると最初からPWM信号を発生します。 「Do not start」を設定すると「PWMゲート信号スタート」素子を使用してPWM信号を開始する必要があります。

PWM生成器は、三相システムのために、正弦波変調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、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。搬送波はデューティサイクルは0.5の三角波です。

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信号を生成します。入力の「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-2πから4πまでです。

PWMジェネレータの入力範囲は-1から1までのため、また、PSIMシミュレーションでは、通常PWM信号は、変調信号を搬送信号と比較することによって生成され、変調信号範囲は-1から1までではないので、スケーリングは変調信号がPWM生成器へ送られる前に必要となります。

PWM(sub)素子はPWM素子と入力スケーリングオフセット回路とから構成されており、次のようになります。コンパレータベースのPWMからPWMハードウェア生成素子への切換えに便利です。

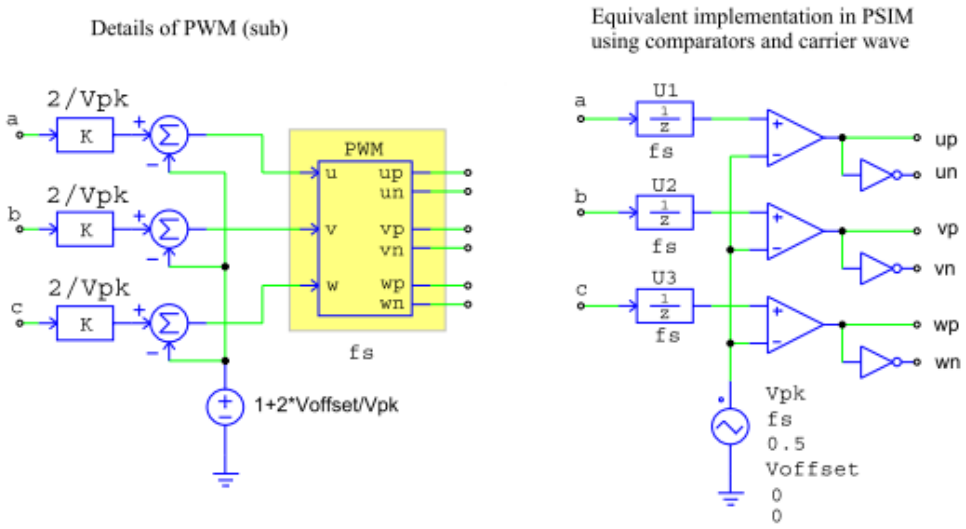


図 9-3-1　PWM(sub)発生器の詳細

PWM(sub)の詳細を**図9-3-1**の左側に示します。この回路はスケーリングオフセット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のコンパレーター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9-3-1**の右側に示します。

キャリアソースのパラメータは以下となります。

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

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、搬送波は三角波でそのデューティサイクルは0.5です。

スケーリング係数は、PWMによる割り込みが発生する頻度を指定します。

パラメータ「制御周波数」は、サンプリング周波数を指定します。 この周波数は、PWMスイッチング周波数と異なる場合があります。

制御周波数fsmp、スケーリング係数Kscale、およびPWMキャリア周波数fswの関係は、次のように計算されます。

パラメーター「割り込みモード」が「Interrupt at peak」または「Interrupt at valley」として設定されている場合：

*fsw = Kscale * fsmp*

たとえば、Kscale = 3の場合、PWMによる割り込みは3 PWM周期ごとに1回発生します。

パラメーター「割り込みモード」が「Interrupt at peak/valley」として設定されている場合：

*fsw = Kscale * fsmp / 2*

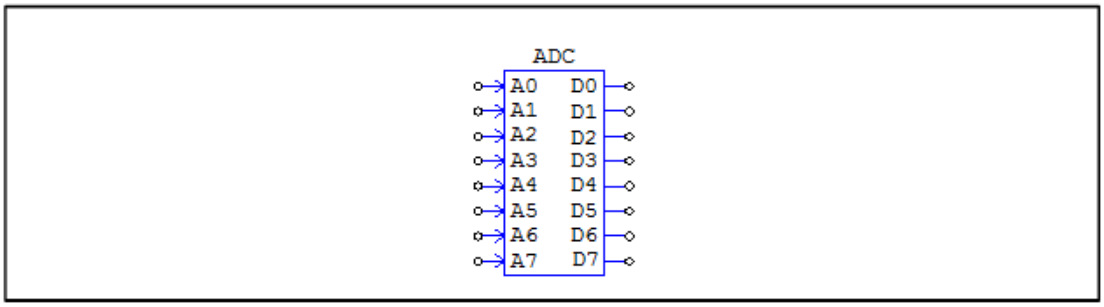
たとえば、Kscale = 3の場合、PWMによる割り込みは3 x 2 = 6 PWM周期ごとに1回発生します。

注）PWMとPWM(sub)生成器がシミュレーション実行される時、デッドタイムは無視されシミュレーションに考慮されません。

9.3.3 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
ボード番号	A/D変換器を含んでいるPEVボードのボード番号
AD変換モード	A/D変換器モード(連続かPWMトリガかの選択)
AD*i* レンジ	i番目のA/Dチャンネルの出力レンジ　V_{range}(i=0から7)
AD*i* オフセット	i番目のA/Dチャンネルの出力オフセット　V_{offset}(i=0から7)

A/D変換器の入力レンジは-5V～+5Vです。A/D変換器の出力レンジは $-V_{range}$～V_{range}です。出力は下記に基づいてスケーリングされます:

$$Vo = Vi \times V_{range} / 5 - V_{offset}$$

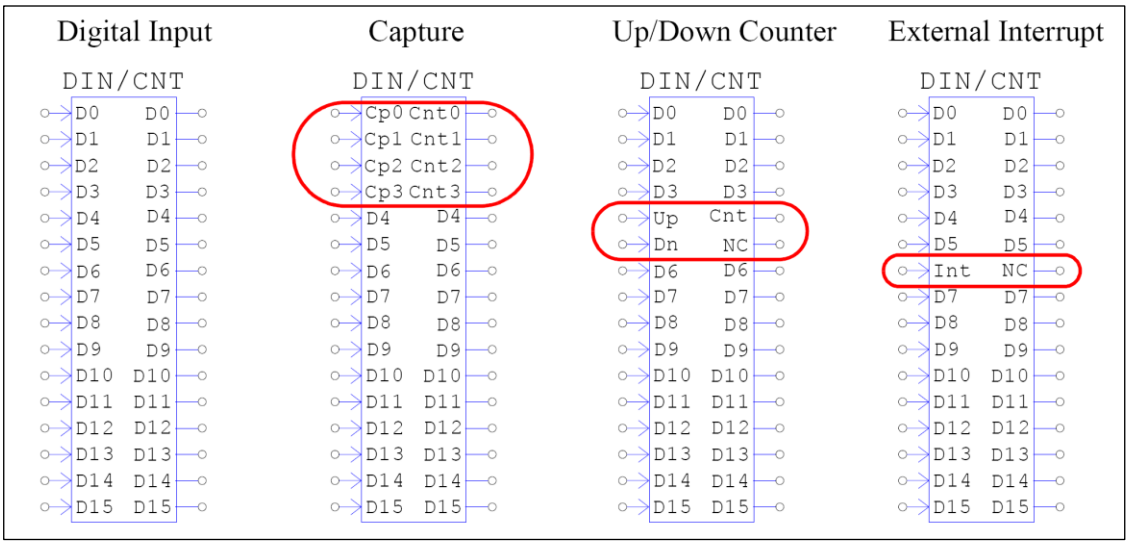
例えば、A/D入力V_i =2V、*A/D レンジ*V_{range}=20、V_{offset}=0.5の場合、出力*Vo*は*Vo=2 * 20 / 5-0.5=7.5V*になります。

9.3.4 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次に示します

シンボル:



仕様:

パラメータ	機能
ボード番号	PEVボードのボード番号
入力/キャプチャ *i*	以下の1つになります：インデックス範囲は0から3、それぞれがD0からD3です。 ■ *Digital Input i*：入力ピンD_iはデジタル入力になります。 ■ *Capture i*：入力ピンD_i はキャプチャの入力になり、出力ピンD_iはキャプチャ*i*の出力なります。入力・出力ピンのキャプションはCap_iとCnt_iにに変化します。
カウンタソース *i*	インデックス範囲は0から3、それぞれがD0からD3です。パラメータは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
入力4と5/カウンター	以下の1つになります： ■ *Digital Input4 and 5*:入力ピンD4およびD5はデジタル入力になります。 ■ *Counter*：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。出力ピンD5のキャプションはCnt(カウンタ出力用)に変更され、ピンD6はNC（接続なし）に変更されます。
カウンターモード	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。
入力6/外部割込み	以下の1つになります： ■ *Digital Input 6*:入力ピン6はデジタル入力になります。 ■ *External Interrupt*: このピンは外部割り込みの入力になります。入力ピンはInt(Interrupt)および出力ピンはNC（接続なし）に変化します。

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(0)からHigh(1)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フリーランタイマ(PEVボードの32ビットフリーランカウンタ)のカウンタ値を保存し出力します。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」入力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」入力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」入力にパルスが入力されたときに「direc」入力が0であればカウントアップし、そして「direc 」入力が1ならばカウントダウンします。

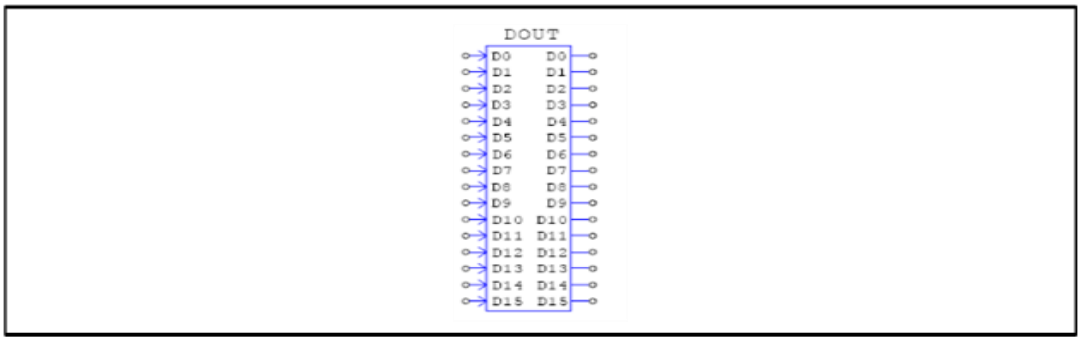
外部割り込みとして:

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

9.3.5 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:



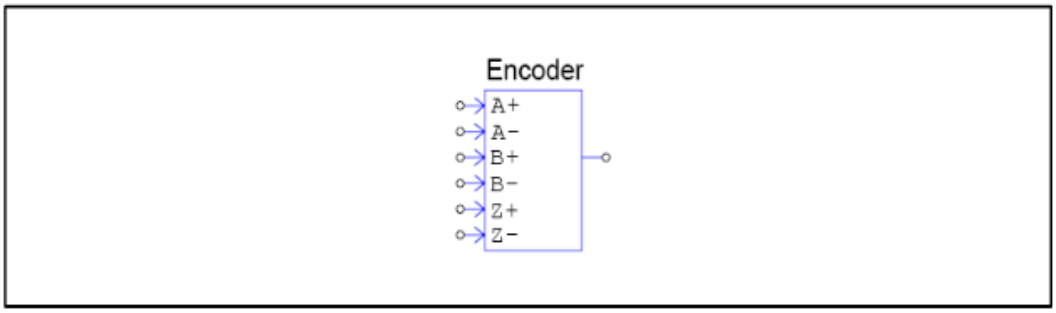
仕様：

パラメータ	機能
ボード番号	デジタル出力素子を含んでいるPEVボードのボード番号

9.3.6 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
ボード番号	エンコーダを含んでいるPEVボードのボード番号
Z信号の使用	Z信号が使用有無の指定
カウント方向	カウントタ方向(「Forward(順方向)」か「Reverse(逆方向)」)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。
エンコーダ分解能	エンコーダ分解能を設定します。エンコーダカウンターは0と分解能-1の間でカウントアップ/カウントダウンします。

エンコーダ出力値の出力はカウンター値となります。また、割り込みは入力信号**Z+** と**Z-**により生成されます。エンコーダはZ信号により割込みをトリガします。TOはエンコーダ割り込みルーチンを実装しており、割り込み素子を割込みソース（エンコーダ素子に名前を入力）とどのエッジトリガが割込むか(立下りエッジ、立ち上がりエッジもくは立ち上がり/立下りエッジなのか)を定義して使用する必要があります。この素子はイベント接続ワイヤによってイベントサブ回路に接続されます。

9.5 PE-Expert4 ランタイムライブラリ

PE-Expert4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-Expert4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-Expert4ランタイムライブラリを以下の表に示します。

表 9-5- 1　PSIM 素子と PE-Expert4 ランタイムライブラリ対応表

PSIM素子	PE-Expert4ランタイムライブラリ
Sine or Sine (in rad.)	mwsin(float x)
Cosine or Cosine (in rad.)	mwcos(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)

SimCoder User Manual

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： https://www.myway.co.jp
Eメール： sales@myway.co.jp