PSIMチュートリアル
Cブロックの使い方

Mywayプラス株式会社

1. はじめに

このテクニカルノートではＣブロック*の使用法の説明を行います。

Ｃブロック（C Block）では、ユーザのＣコードを直接入力することができます。Ｃブロックの Ｃコードはコンパイラがなくても、シミュレーション実行時に、PSIM 内蔵のＣインタプリタによって解釈され実行されます。

*注意: Ｃブロックはデモ版では利用できません。

2. Ｃブロックの使い方

Ｃブロックはメニューバーの「Element(素子)」→「Other(その他)」→「Function Blocks(その他関数ブロック)」→「C Block(C ブロック)」をクリックしてＣブロックを選択します。回路図上に置き、ダブルクリックするとＣブロックの設定ウィンドウが表示されます。

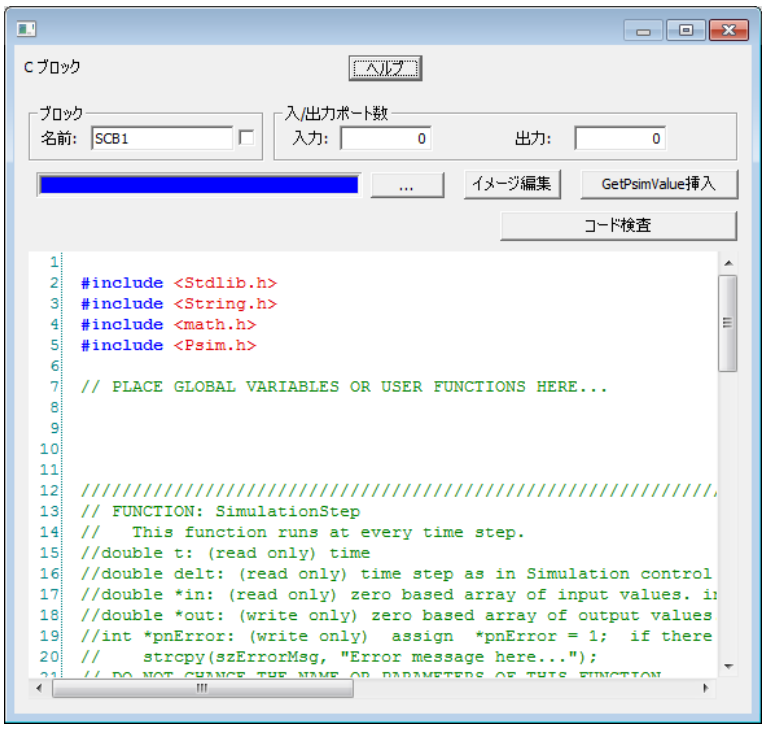
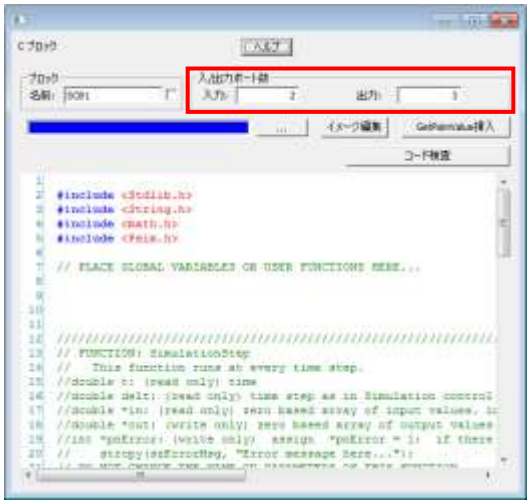


図 1 Ｃブロックのウィンドウ

2.1. 入力と出力のポートの数を入力

「入/出力ポート数」にブロックに入力するポート数と、出力のポート数を入力します。入力した数だけポートが現れます。



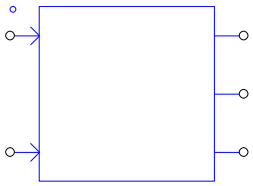


図 2 C ブロックのポート

C コードの中では入力と出力は配列で扱います。

今回の場合は入力ポート数が 2 なので、C ブロックの入力ポートのうち、上から in[0]、in[1]、出力は出力ポートが 3 なので、上から out[0]、out[1]、out[2]です。出力はオープンのままでもシミュレーションを実行できますが､入力はオープンのままシミュレーションを実行することはできません。

2.2. C コードを入力する

デフォルトでエディタエリアには下記のテキストが入力されています。

```
#include <Stdlib.h>
#include <String.h>
#include <math.h>
#include <Psim.h>

// PLACE GLOBAL VARIABLES OR USER FUNCTIONS HERE...


////////////////////////////////////////////////////////////////////////
// FUNCTION: SimulationStep
//   This function runs at every time step.
//double t: (read only) time
//double delt: (read only) time step as in Simulation control
//double *in: (read only) zero based array of input values. in[0] is the first node, in[1] second input...
//double *out: (write only) zero based array of output values. out[0] is the first node, out[1] second output...
//int *pnError: (write only)  assign  *pnError = 1;  if there is an error and set the error message in szErrorMsg
//    strcpy(szErrorMsg, "Error message here...");
// DO NOT CHANGE THE NAME OR PARAMETERS OF THIS FUNCTION
void SimulationStep(
                  double t, double delt, double *in, double *out,
                   int *pnError, char * szErrorMsg,
                   void ** reserved_UserData, int reserved_ThreadIndex, void * reserved_AppPtr)
{
// ENTER YOUR CODE HERE...



}


////////////////////////////////////////////////////////////////////////
// FUNCTION: SimulationBegin
//   Initialization function. This function runs once at the beginning of simulation
//   For parameter sweep or AC sweep simulation, this function runs at the beginning of each simulation cycle.
//   Use this function to initialize static or global variables.
//const char *szId: (read only) Name of the C-block
```

```
//int nInputCount: (read only) Number of input nodes
//int nOutputCount: (read only) Number of output nodes
//int nParameterCount: (read only) Number of parameters is always zero for C-Blocks.  Ignore nParameterCount and pszParameters
//int *pnError: (write only)  assign  *pnError = 1;  if there is an error and set the error message in szErrorMsg
//    strcpy(szErrorMsg, "Error message here...");
// DO NOT CHANGE THE NAME OR PARAMETERS OF THIS FUNCTION
void SimulationBegin(
                    const char *szId, int nInputCount, int nOutputCount,
                     int nParameterCount, const char ** pszParameters,
                     int *pnError, char * szErrorMsg,
                     void ** reserved_UserData, int reserved_ThreadIndex, void * reserved_AppPtr)
{
// ENTER INITIALIZATION CODE HERE...


}


//////////////////////////////////////////////////////////////////////
// FUNCTION: SimulationEnd
//   Termination function. This function runs once at the end of simulation
//   For parameter sweep or AC sweep simulation, this function runs at the end of each simulation cycle.
//   Use this function to de-allocate any allocated memory or to save the result of simulation in an alternate file.
// Ignore all parameters for C-block
// DO NOT CHANGE THE NAME OR PARAMETERS OF THIS FUNCTION
void SimulationEnd(const char *szId, void ** reserved_UserData, int reserved_ThreadIndex, void * reserved_AppPtr)
{



}
```

テキストに従い、C コードを書き込んでいきます。C ブロック内の C コードは 4 つのセクションに分かれています。

2.2.1. グローバル変数/ユーザ関数定義

「// PLACE GLOBAL VARIABLES OR USER FUNCTIONS HERE...」以降にグローバル変数やユーザ定義の関数を定義します。

2.2.2. タイムステップごとに呼ばれる関数(SimulationStep)

「// ENTER YOUR CODE HERE...」以降にシミュレーションステップごとに呼ばれる関数を記述します。

2.2.3. 初期化の関数(SimulationBegin)

「// ENTER INITIALIZATION CODE HERE...」以降に初期化のためにシミュレーションの最初に 1 度だけ(パラメータスイープ、AC スイープシミュレーションでは各シミュレーションサイクルの開始時)呼ばれる関数を記述します。スタティック変数やグローバル変数を初期化するのに使用します。

2.2.4. 終了時の関数(SimulationEnd)

シミュレーションの最後に 1 度だけ(パラメータスイープ、AC スイープシミュレーションでは各シミュレーションサイクルの終了時)呼ばれる関数を記述します。任意の割り当てられたメモリの割り当てを解除したり、別のファイルにシミュレーションの結果を保存したりするには、この関数を使用します。

2.3. C コードのチェック

C コードを入力したら C ブロックの「コード検査(Check Code)」をクリックし、コンパイルが通ると下記メッセージが表示されます。

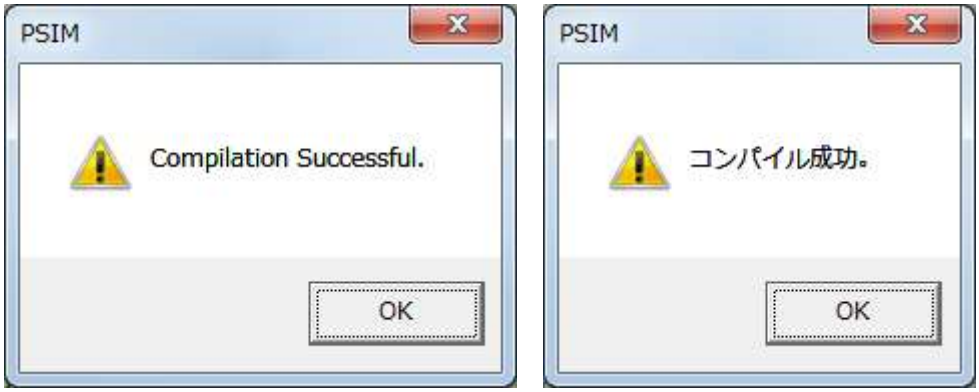


図 3 コンパイル成功

2.4. PSIM で定義される変数について

PSIM のシミュレーションに関する変数は下記のように定義されます。セクションごとに定義される関数が異なるため、使用できる変数が変わります。

有効なセクション		変数名
SimulationStep	C ブロックの入力値	上から 1 つ目のノードは in[0] 2 つ目のノードは in[1]
	C ブロックの出力値	上から 1 つ目のノードは out[0] 2 つ目のノードは out[1]
	シミュレーションの時間	t
	タイムステップ	delt
	エラーフラグ	pnError
	エラーになったときのメッセージ	szErrorMsg
SimulationBegin	C ブロックの名前	szId
	入力ポートの数	nInputCount
	出力ポートの数	nOutputCount
	エラーフラグ	pnError
	エラーになったときのメッセージ	szErrorMsg
SimulationEnd	C ブロックの名前	szId

2.5. ヘッダーファイルの読み込み

ヘッダーファイルをインクルードする場合は、PSIM のメニュー 「オプション」→「パス設定」を開き、「C ブロックインクルードパス」にヘッダーファイルを保存しているフォルダを設定します。

3. シミュレーション回路

C ブロックを使用した回路を作成します。以下にサンプル回路を示します。

サンプル回路は C ブロックに入力した値の実効値を算出して出力します。

【ファイル保存場所】C:¥Program Files¥Powersim¥PSIM***¥examples¥C Block

【ファイル名】test C Block rms.sch

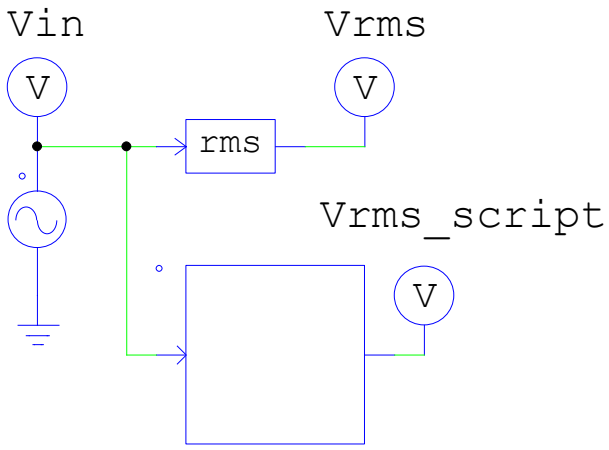


図 4 サンプルのシミュレーション回路

C ブロックのウィンドウを以下に示します。

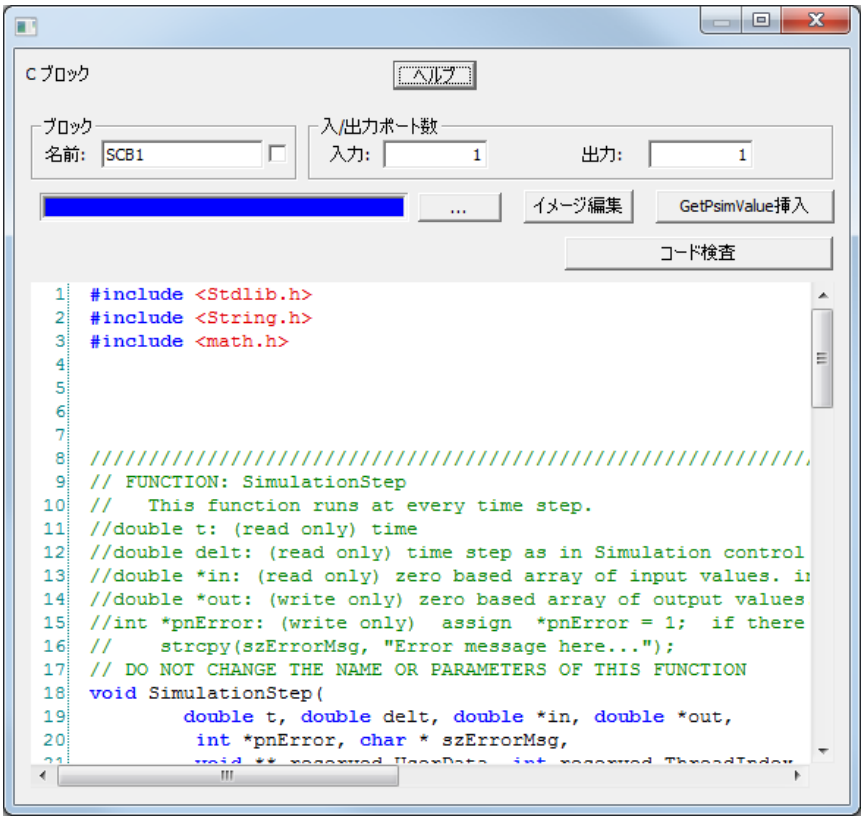


図 5 サンプル回路の C ブロックウィンドウ

入力と出力は１つずつなので、「入/出力ポート数」は入力、出力共に１です。

また、入力したＣコードを下記に示します。

```
#include <Stdlib.h>
#include <String.h>
#include <math.h>



////////////////////////////////////////////////////////////////////
// FUNCTION: SimulationStep
//   This function runs at every time step.
//double t: (read only) time
//double delt: (read only) time step as in Simulation control
//double *in: (read only) zero based array of input values. in[0] is the first node, in[1] second input...
//double *out: (write only) zero based array of output values. out[0] is the first node, out[1] second output...
//int *pnError: (write only)  assign  *pnError = 1;  if there is an error and set the error message in szErrorMsg
//    strcpy(szErrorMsg, "Error message here...");
// DO NOT CHANGE THE NAME OR PARAMETERS OF THIS FUNCTION
void SimulationStep(
                    double t, double delt, double *in, double *out,
                     int *pnError, char * szErrorMsg,
                     void ** reserved_UserData, int reserved_ThreadIndex, void * reserved_AppPtr)
{
    static double nsum=0., sum=0., rms;
    double Tperiod;

    Tperiod=1./60.;

    if (t >= nsum*Tperiod)
    {
                nsum=nsum+1.;
                rms = sqrt(sum*delt/Tperiod);
                sum=0.;
    }

    out[0] = rms;
    sum=sum+in[0]*in[0];

}



////////////////////////////////////////////////////////////////////
// FUNCTION: SimulationBegin
//   Initialization function. This function runs once at the beginning of simulation
//   For parameter sweep or AC sweep simulation, this function runs at the beginning of each simulation cycle.
//   Use this function to initialize static or global variables.
//const char *szId: (read only) Name of the C-block
//int nInputCount: (read only) Number of input nodes
//int nOutputCount: (read only) Number of output nodes
//int nParameterCount: (read only) Number of parameters is always zero for C-Blocks.  Ignore nParameterCount and pszParameters
//int *pnError: (write only)  assign  *pnError = 1;  if there is an error and set the error message in szErrorMsg
//    strcpy(szErrorMsg, "Error message here...");
// DO NOT CHANGE THE NAME OR PARAMETERS OF THIS FUNCTION
void SimulationBegin(
                    const char *szId, int nInputCount, int nOutputCount,
                     int nParameterCount, const char ** pszParameters,
                     int *pnError, char * szErrorMsg,
                     void ** reserved_UserData, int reserved_ThreadIndex, void * reserved_AppPtr)
{


// In case of error, uncomment next two lines. Set *pnError to 1 and copy Error message to szErrorMsg
 //*pnError=1;
 //strcpy(szErrorMsg, "Place Error description here.");

}



////////////////////////////////////////////////////////////////////
// FUNCTION: SimulationEnd
//   Termination function. This function runs once at the end of simulation
//   For parameter sweep or AC sweep simulation, this function runs at the end of each simulation cycle.
//   Use this function to de-allocate any allocated memory or to save the result of simulation in an alternate file.
// Ignore all parameters for C-block
// DO NOT CHANGE THE NAME OR PARAMETERS OF THIS FUNCTION
void SimulationEnd(const char *szId, void ** reserved_UserData, int reserved_ThreadIndex, void * reserved_AppPtr)
{
```

```
}


}
```

シミュレーションステップごとに計算を行う必要があるため、「FUNCTION:SimulationStep」内の「// ENTER YOUR CODE HERE...」以下に実効値を計算する処理を記載します。
関数の定義、初期化、終了の処理はありませんので、他のエリアには処理はありません。
「コード検査(Check Code)」をクリックし、コンパイルが通ることを確認します。

4. シミュレーション結果

シミュレーションを実行します。

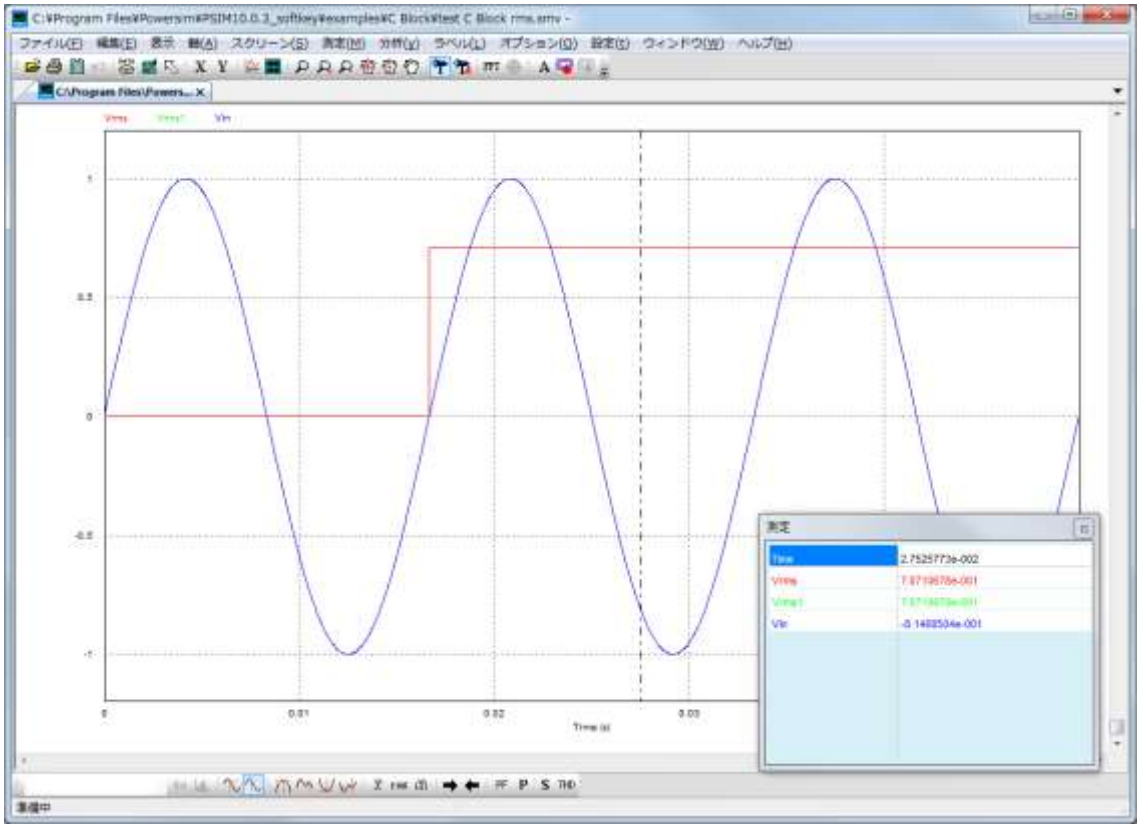


図 6 シミュレーション結果

RMS ブロックを使って計算した結果(Vrms)と C ブロックを使って計算した結果(Vrms1)が一致していることが分かります。RMS ブロックは各周期の初めにのみ更新される仕様となっているため、最初の周期は rms=0 となり、2 周期目から計算されます。C ブロックも同様の動作となるように計算しています。

ご注意

1. 本資料に記載された製品の仕様は、予告なく変更することがあります。
2. 本資料の内容については、万全を期しておりますが、万一ご不明な点などがありましたら、弊社までお申しつけください。
3. 本資料に記載された情報に起因する損害または特許権その他権利の侵害に関しては、弊社は一切の責任を負いません。
4. 本資料によって第三者または弊社の特許権その他権利の実施権を許諾するものではありません。
5. 弊社の書面許諾なく、本資料の一部または全部を無断で複製することを固くお断りします。
6. 本資料に記載されている会社名、商品名は、各社の商標または登録商標です。

発行：Myway プラス株式会社
〒222-0022
横浜市西区花咲町 6-145　横浜花咲ビル
TEL：045-548-8831
FAX：045-548-8832

ホームページ：http://www.myway.co.jp
Ｅメール：　sales@myway.co.jp