PSIMチュートリアル

General-DLLブロックの使い方

Tutorial – How to Use General DLL Block October 2016

Mywayプラス株式会社

はじめに

固定数の入力と出力を持つ単純な DLL ブロック（3 つの入力と 3 つの出力を持つ DLL ブロックなど）とは異なり、一般的な DLL ブロックは PSIM とカスタム DLL ファイルとのインターフェースにおいてより高い柔軟性と機能を提供します。 このファイルは、一般的な DLL ブロックの規則と基本を説明しています。

一般的な DLL ブロックを使用すると、ユーザーは C または C ++でコードを記述し、それを Windows DLL としてコンパイルし、PSIM にリンクすることができます。 エクスポートされた関数は 4 つあります。そのうちの 3 つは PSIM シミュレーションエンジンによって使用され、1 つはユーザーインターフェイスによって使用されます。

ユーザーインターフェース機能は **REQUESTUSERDATA** です。 3 つのシミュレーション機能は **SimulationStep**、**SimulationBegin**、および **SimulationEnd** です。そして、 2 つの追加関数：**GetPsimValue** と **GetPsimText** があります。これらは、タイムステップ、合計時間、回路図ファイルパス、その他の回路パラメータなどの情報を取得するためにシミュレーション関数内で呼び出すことができます。 各関数には、非 Unicode バージョンと Unicode バージョンがあります。 それらについて以下に説明します。

1. 関数の説明

1.1. 関数 SimulationStep ：

＜非-Unicode版＞

```
void SimulationStep (
  double t,
  double delt,
  double *in,
  double *out,
  int *pnError,
  char * szErrorMsg,
  void ** ptrUserData,
  int nThreadIndex,
  void * pAppPtr
  )
```

＜Unicode版＞

```
void SimulationStepW (
  double t,
  double delt,
  double *in,
  double *out,
  int *pnError,
```

```
    wchar_t* szErrorMsg,
    void ** ptrUserData,
    int nThreadIndex,
    void * pAppPtr
    )
```

この関数は、必須の DLL ルーチン内の唯一の関数で、各タイムステップで PSIM によって呼び出されます。他のすべての関数はオプションです。

DLL ルーチンは PSIM から値を入力として受け取り、計算を実行し、結果を PSIM に送り返します。 ノードの割り当ては次のとおりです。入力ノードは左側にあり、出力ノードは右側にあります。 順序は上から下です。各パラメータについて以下に説明します。

double t :

(Read only) 時間[s]

double delt :

(Read only) タイムステップ[s]

double * in :

(Read only) 入力値の配列。もし DLL ブロックが３入力の場合、in[0], in[1], in[2]でアクセスします。

double *out :

(Write only) 出力値の配列。計算が実行された後、出力はこの配列に書き込まれます。DLL ブロックが 4 出力の場合、out [0], out [1], out [2], out [3] でアクセスします。

int *pnError:

(Write only) 正常終了の場合は pnError を 0 に、エラーの場合は 1 に設定します。

例) *pnError = 0; //success*

char * szErrorMsg: (Unicode 版： wchar_t* szErrorMsg)

(Write only) エラーの場合にエラーメッセージをこの文字列に出力します。

例) strcpy(szErrorMsg, "input 2 cannot exceed 50V");

例) Unicode 版：_tcscpy(szErrorMsg, _T("input 2 cannot exceed 50V"));

void ** ptrUserData:

(Read, Write) ユーザ変数のポインタ。詳細は SimulationBegin 関数を参照してください。

int nThreadIndex:

(Read only) GetPsimValue および GetPsimText を呼出す時の必須パラメータです。

void * pAppPtr:

(Read only) GetPsimValue および GetPsimText を呼出す時の必須パラメータです。

1.2. 関数 SimulationBegin：

＜非-Unicode版＞

```
void SimulationBegin(
  const char *szId,
  int nInputCount,
  int nOutputCount,
  int nParameterCount,
  const char ** pszParameters,
  int *pnError,
  char * szErrorMsg,
  void ** ptrUserData,
  int nThreadIndex,
  void * pAppPtr
  )
```

＜Unicode版＞

```
void SimulationBeginW(
  const const wchar_t *szId,
  int nInputCount,
  int nOutputCount,
  int nParameterCount,
  const wchar_t ** pszParameters,
  int *pnError,
  wchar_t * szErrorMsg,
  void ** ptrUserData,
  int nThreadIndex,
  void * pAppPtr
  )
```

この関数はオプションです。 シミュレーションの開始時に一度だけ呼び出されます。 それは DLL ルーチンから情報を受け取り、DLL ルーチンがそれ自身の使用のためにメモリを割り当てることを可能にします。 以下でパラメータについて説明します。

const char *szId （Unicode 版: const wchar_t *szId）：
　(Read only) DLL block ブロック名
int nInputCount:
　(Read only) 入力ノード数
int nOutputCount:
　(Read only) 出力ノード数
int nParameterCount:

(Read only) オプションの入力ファイルパスを含むパラメータの数

const char ** pszParameters: (Unicode版: const wchar_t ** pszParameters)

(Read only) パラメータ値。Indexは0から開始します (pszParameters[0])。
オプションの入力ファイルが存在する場合、それが最後のパラメータになります。
(pszParameters[nParameterCount-1])。

int *pnError:

(Write only) 正常終了の場合はpnErrorを0に、エラーの場合は1に設定します。

例) *pnError = 0; //success*

char * szErrorMsg: (Unicode版: wchar_t* szErrorMsg)

(Write only) エラーがあればエラーメッセージをこの文字列にコピーします。

例) strcpy(szErrorMsg, "input 2 can not exceed 50V");

Unicode版:例) _tcscpy(szErrorMsg, _T("input 2 can not exceed 50V"));

void ** ptrUserData:

(Read, Write) ユーザー定義データのポインターです。 SimulationBeginでメモリを割り当て、SimulationEndで解放する必要があります。 これらはすべての呼び出しでSimulationStepに渡されます。DLLはSimulatin中に独自のデータ管理ができます。

注) このポインターは、関数REQUESTUSERDATAのユーザー定義ポインターとは異なります。 REQUESTUSERDATAからどのシミュレーション関数にもポインタを渡すことはできません。それらはパラメータのみを介して通信します。

int nThreadIndex:

(Read only) GetPsimValueおよびGetPsimTextを呼び出すための必須パラメータです。

void * pAppPtr:

(Read only) GetPsimValueおよびGetPsimTextを呼び出すための必須パラメータです。

1.3. 関数 SimulationEnd :

＜非-Unicode版＞

```
void SimulationEnd (
  const char *szId,
  void ** ptrUserData,
  int nThreadIndex,
  void * pAppPtr
  )
```

＜Unicode版＞

```
void SimulationEndW(
  const wchar_t *szId,
  void ** ptrUserData,
  int nThreadIndex,
  void * pAppPtr
```

)

この関数はオプションです。シミュレーション終了時に一度だけ呼び出されます。 その主な目的は、DLL が割り当てたメモリやリソースを解放できるようにすることです。 以下でパラメータの説明をします。

const char *szId : (Unicode版: const wchar_t *szId)

(Read only) DLL blockブロック名

void ** ptrUserData:

(Read, Write) ユーザ定義データのポインタ。詳細は、SimulationBegin関数を参照してください。

int nThreadIndex:

(Read only) GetPsimValueおよびGetPsimTextを呼び出すための必須パラメータです。

void * pAppPtr:

(Read only) GetPsimValueおよびGetPsimTextを呼び出すための必須パラメータです。

1.4. 関数 GetPsimValue :

＜非-Unicode版＞

```
double GetPsimValueA(
  int nThreadIndex,
  void * pAppPtr,
  char * szName,
  char * szParamName,
  int & nStatus
  )
```

＜Unicode版＞

```
double GetPsimValueW(
  int nThreadIndex,
  void * pAppPtr,
  wchar_t * szName,
  wchar_t * szParamName,
  int & nStatus
  )
```

この関数は、シミュレーション中に回路情報を取得するために SimulationStep、SimulationBegin、または SimulationEnd で呼び出されます。以下でパラメータの説明をします。

int nThreadIndex:

これは各シミュレーション関数に存在する必須パラメータです。

void * pAppPtr:

これは各シミュレーション関数に存在する必須パラメータです。

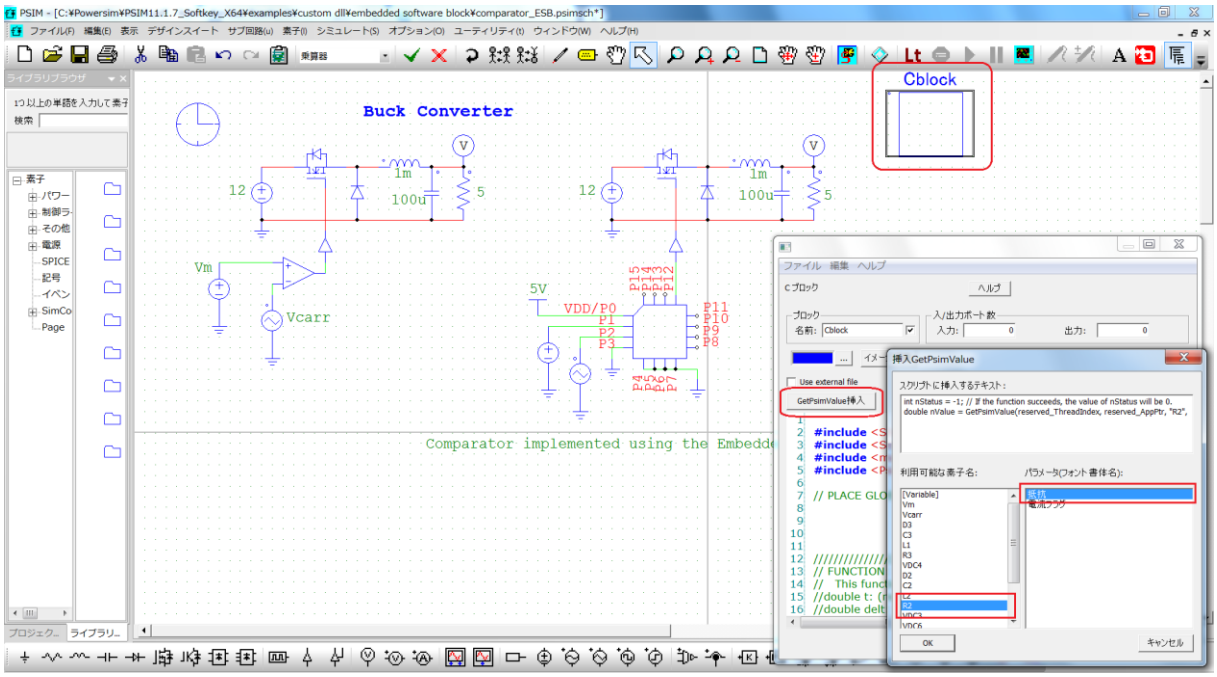
const char * szName: (Unicode版: const wchar_t * szName)

const char * szParamName: (Unicode版: const wchar_t * szParamName)

szNameとszParamNameの組み合わせによって、この関数によって取得されるパラメーター値が決まります。

定義・説明	szName	szParamName
Total Time	"PSIM"	"TOTALTIME"
Time Step	"PSIM"	"TIMESTEP"
Load Flag (0 or 1)	"PSIM"	"LOADFLAG"
Save Flag (0 or 1)	"PSIM"	"SAVEFLAG"
パラメータファイルから変数の値を取得します。例：'alpha'	"" (空文字)	変数名:"alpha"
回路要素の属性を取得します。 範例"comparator_ESB.psimsch"の'R2'-'Resistance':Cblock を配置し、[GetPsimValue]をクリックして素子名とパラメータを取得します。C ブロックは DLL ブロックと互換性があります。（下図）	素子名："R2"	パラメータ："Resistance"

◆範例"comparator_ESB.psimsch"



int & nStatus:

関数が正常に実行されると、nStatus は 0 に設定され、関数は戻り値を返します。nStatus がゼロではない場合、関数にエラーがあり、戻り値は 0 を返します。

Return value: (double)

戻り値。nStatus が 0 の場合は正常値。(nStatus が 0 以外は戻り値は 0 を返します。)

この関数はヘッダファイル"psim.h"が必要です。 以下で nStatus の記述例を示します。

```
int nStatus = -1; // If the function succeeds, the value of nStatus will be 0.
double fTotalTime = GetPsimValueW(nThreadIndex, AppPtr, _T("PSIM"), _T("TOTALTIME"),
nStatus);
if(nStatus == 0)
{
    //Success
    //...
}
else
{
    //Error...
    //fTotalTime is invalid.
    //...
}
```

1.5. 関数 GetPsimText ：

＜非-Unicode版＞

```
int GetPsimTextA(
    int nThreadIndex,
    void * pAppPtr,
    char * szName,
    char * szParamName,
    char *szReturn,
    int *pnReturnLen,
    int & nStatus
    )
```

＜Unicode版＞

```
int GetPsimTextW(
    int nThreadIndex,
    void * pAppPtr,
    wchar_t * szName,
    wchar_t * szParamName,
    wchar_t *szReturn,
    int *pnReturnLen,
    int & nStatus
    )
```

この関数は、シミュレーション中に回路情報を取得するために SimulationStep、SimulationBegin、または SimulationEnd で呼び出されます。以下でパラメータの説明をします。

int nThreadIndex:

これは各シミュレーション関数に存在する必須パラメータです。

void * pAppPtr:

これは各シミュレーション関数に存在する必須パラメータです。

const char * szName: (Unicode版: const wchar_t * szName)

const char * szParamName: (Unicode版: const wchar_t * szParamName)

szNameとszParamNameの組み合わせによって、この関数によって取得されるパラメーター値が決まります。

定義・説明	szName	szParamName
PSIM Folder	"PSIM"	"PSIMFOLDER"
Three part PSIM version (10.0.5)	"PSIM"	"PSIMVERSION"
Full path and file name of schematic file.	"PSIM"	"SCHPATH"
Full path and file name of simview file.	"PSIM"	"SMVPATH"
Schematic file's Path without the file name.	"PSIM"	"SCHFOLDER"
Simview file's Path without the file name.	"PSIM"	"SMVFOLDER"
Schematic file name without the folder path.	"PSIM"	"SCHNAME"
Simview file name without the folder path.	"PSIM"	"SMVNAME"
To retrieve the value of a string variable from a parameter file. For example variable 'TableFilePath'.	(Empty string) ""	Variable name: "TableFilePath"

char *szReturn: (Unicode版: wchar_t *szReturn)

nStatusが０の場合の返される文字列。

int *pnReturnLen:

返される文字列の長さ。

int & nStatus:

関数が正常に実行されると、nStatusは0に設定され、関数は戻り値を返します。関数の実行後にnStatusがゼロではない場合、関数にエラーがあり、戻り値は０を返します。

1.6. 関数 REQUESTUSERDATA :

＜非-Unicode版＞

```
void REQUESTUSERDATA(
        int nRequestReason,
        int nRequestCode,
```

```
        int nRequestParam,
        void ** ptrUserData,
        int * pnParam1,
        int * pnParam2,
        char * szParam1,
        char * szParam2
        )
```

＜Unicode版＞

```
void REQUESTUSERDATAW(
        int nRequestReason,
        int nRequestCode,
        int nRequestParam,
        void ** ptrUserData,
        int * pnParam1,
        int * pnParam2,
        wchar_t * szParam1,
        wchar_t * szParam2
        )
```

この関数はオプションです。PSIM とのユーザーインターフェイスを処理します。これは、General DLL ブロック素子が作成された、または、そのプロパティがプロパティボックスで変更されたときに PSIM によって呼び出されます。以下でパラメータの説明をします。

int nRequestReason：

この関数が呼び出されたときのユーザーアクションを記述します。 可能なユーザーアクション値は次のとおりです。

ユーザーアクション値	ユーザーアクション説明
ACTION_DLL_SELECTED	回路図に新しいDLLブロックが配置され、このDLLが選択されます。
ACTION_ELEMENT_LOAD	DLLブロックを含む回路図ファイルが、読み込まれているかコピー/ペースト操作として貼り付けられています。
ACTION_ELEMENT_SAVE	DLLブロックを含む回路図ファイルを、保存するかバッファします（コピー/貼り付け操作）。
ACTION_INPUTFILE_CHANGED	新しい入力ファイルが選択されます。
ACTION_PARAMETERS_CHANGED	プロパティダイアログボックスでパラメータを変更します。
ACTION_ELEMENT_DELETE	素子(要素)が削除されているか、ファイルが閉じます。

例：ユーザーがライブラリから回路図にDLLブロックを配置すると、このブロックにDLLファイル名が定義された後、関数REQUESTUSERDATAが呼び出され、パラメータnRequestReasonがACTION_DLL_SELECTEDに設定されます。

int nRequestCode :

DLLから要求されている情報について記述します。可能な指定値は次のとおりです。

REQUEST_BEGIN:

新しいリクエストが開始されます。 pnParam1、pnParam2、szParam1、および szParam2の値は、nRequestReasonの値によって異なります。

REQUEST_END:

リクエストを完了します。

pnParam1: pnParam2: szParam1: szParam2: は無視されます。

REQUEST_PARAM_COUNT:

プロパティボックスの右側に表示するパラメータ数を0～10の範囲で指定します。

pnParam1(Read, Write): パラメータの数

pnParam2(Read, Write): 入力データファイルが必要な場合は1に設定し、それ以外の場合は0に設定します。

szParam1(Read, Write): 入力データファイル用のファイルオープンダイアログフィルタ。 ファイルを開くダイアログのフィルタフォーマットに準拠している必要があります。 以下は適切にフォーマットされたフィルタ文字列です :"All Files|*.*| Image files|*.jpg;*.bmp;*.gif| Text Files|*.txt||"。 フィルタのデフォルト値は "All Files|*.*||"です。

szParam2(Read only): 要素に関連付けられた文字列ID。

例：次のコードは、この要求に対する可能な応答です。 3つのパラメータと拡張子が「aaa」の入力ファイルを要求します。

```
*pnParam1 = 3; // 3 parameters
*pnParam2 = 1; // Input Data File is required
strcpy(szParam1, "MyProgram Files|*.aaa|All Files|*.*||"); //File Filter
```

REQUEST_DATAFILE_INFO:

入力データファイル情報の要求です。

pnParam1(Read, Write): 1に設定すると、チェックボックスがプロパティボックスに表示されます。 チェックボックスをオンにすると、ファイルパスが回路図に表示されます。

pnParam2: 無視されます。

szParam1(Read, Write): プロパティボックスに表示されるラベル。 長さは最大20文字です。 デフォルト値は "Input Data File"です。

szParam2(Read Only): 選択したファイルのフルパスとファイル名。

例：次のコードは、ラベルをデフォルト値の "Input Data File"から "MyProgram data File"に変更し、データファイルのチェックボックスを要求します。

```
strcpy(szParam1, "MyProgram data File");
*pnParam1 = 1; // Show Display check box
```

REQUEST_PARAM_INFO:

各パラメーター情報の要求です。

nRequestParam(Read only): ゼロベースのパラメータインデックス

pnParam1: 1に設定すると、チェックボックスがプロパティボックスに表示されます。 このチェックボックスをオンにすると、パラメータ値が回路図に表示されます。

pnParam2 : 無視されます。

szParam1(Read, Write): プロパティボックスに表示されるラベル。 長さは最大20文字です。

szParam2(Read, Write): パラメータ値 長さは最大50文字です。

例：次のコードは、3つのパラメータのラベルを設定し、Parameter1が３～10の整数であることを確認します。

```
strcpy(szParam1, "MyProgram data File");
*pnParam1 = 1; // Show Display check box
switch(nRequestParam)
{ //Three Parameters expected
    case 0:
    strcpy(szParam1, "Moving Avg. Samples");
    *pnParam1 = 1; // Show Display check box
    //verify the value
    nVal = atoi(szParam2);
    if( nVal < 3 )
    {
        nVal = 3;
    }
    if( nVal > 10 )
    {
        nVal = 10;
    }
    itoa( nVal, szParam2, 10);
    break;

    case 1:
    strcpy(szParam1, "Multiplier");
    *pnParam1 = 1; // Show Display check box
    if( strlen(szParam2) == 0 )
    {
        strcpy(szParam2, "3.14");
```

```
            }
            break;

            case 2:
            strcpy(szParam1, "Title");
            *pnParam1 = 0; // Do not Show Display check box
            break;

            default://Only expecting three parameters
            break;
        }
```

REQUEST_IN_OUT_NODES:

入力ノードと出力ノードの数の取得と設定です。

pnParam1(Read, Write): 入力ノードの数

pnParam2(Read, Write): 出力ノードの数

szParam1: 無視されます

szParam2: 無視されます

これらの値はいつでも変更でき、変更は回路図に直ちに反映されます。

例：次のコードは、入力ノード数を1に、出力ノード数を2に設定した例です。

```
*pnParam1 = 1; // one input node
*pnParam2 = 2; // two output nodes
```

REQUEST_INPUT_NODE_INFO:

各入力ノードに関する情報の要求です。

nRequestParam: ゼロベースのノードインデックス

pnParam1: 無視されます

pnParam2: 無視されます

szParam1(Read, Write): ノードラベル。長さは最大20文字です。

szParam2: 無視されます

例：次のコードは、最初の入力ノードのラベルを設定します。

```
switch(nRequestParam)
{
//One input node
    case 0:
    strcpy(szParam1, "in1");
    break;

    default:
    break;
```

```
        }
```

REQUEST_OUTPUT_NODE_INFO

各出力ノードに関する情報の要求です。

nRequestParam: ゼロベースのノードインデックス

pnParam1: 無視されます

pnParam2: 無視されます

szParam1(Read, Write): ノードラベル。長さは最大20文字です。

szParam2: 無視されます

出力ノードごとに、ラベルをいつでも変更できます。

例：次のコードは、2つの出力ノードにラベルを設定します。

```
        switch(nRequestParam)
        { //two output nodes
            case 0:
            strcpy(szParam1, "o1");
            break;

            case 1:
            strcpy(szParam1, "o2");
            break;

            default:
            break;
        }
```

int nRequestParam :

この値はパラメータnRequestCodeに依存します。

void ** ptrUserData:

これはユーザー定義データへのポインターです。 すべての関数呼び出しに含まれており、ユーザーは自分のデータを管理できます。 メモリはユーザーによって割り当てられ、解放されます。 メモリはいつでも割り当て、再割り当て、または解放することができます。ただし、メモリを割り当てるための論理的な選択は次の

REQUESTUSERDATA(ACTION_DLL_SELECTED, REQUEST_BEGIN,…)

および

REQUESTUSERDATA(ACTION_ELEMENT_LOAD, REQUEST_BEGIN, …) です。

データが入力ファイルからロードされている場合は、次の時点でメモリを割り当てることもできます。

REQUESTUSERDATA(ACTION_INPUTFILE_CHANGED,REQUEST_BEGIN,…)

メモリは一般的に次で解放されます。

REQUESTUSERDATA(ACTION_ELEMENT_DELETE, REQUEST_BEGIN,…)

例：次のコードはこの値の使い方を示しています。

```
Struct MyStruct
{
int myInteger;
...
}
//Allocate memory
*ptrUserData = new MyStruct();
...
// Use your internal data
MyStruct * pData = (MyStruct)(*ptrUserData);
PData-> myInteger = 10;
...
// Free memory
if( *ptrUserData != NULL )
{
delete (MyStruct)(*ptrUserData);
*ptrUserData = NULL;
}
```

int * pnParam1, int * pnParam2 , char * szParam1, char * szParam2:

これらの引数は、nRequestReason、nRequestCode、およびnRequestParamの値によって異なります。

2. 呼び出しシーケンス

PSIMと連携するDLLを作成するには、PSIMがDLLと相互作用する方法を理解することが不可欠です。機能REQUESTUSERDATAのパラメータnRequestReasonは、ユーザアクションを記述します。 以下は、nRequestReasonの値に基づいて編成されたREQUESTUSERDATA関数呼び出しの説明です。

ACTION_DLL_SELECTED

ユーザーはPSIM回路図に一般的なDLLブロックを配置し、画像をダブルクリックしてプロパティボックスを表示させ、「DLLファイル」を定義します。
この時点で、PSIMはDLLをロードして関数RUNSIMUSERを探します。 この関数が存在しない場合、DLLは拒否されます。 RUNSIMUSER関数は一般的なDLLブロックには必須です。
次に、PSIMはDLL内で関数REQUESTUSERDATAを探します。 DLLがこの関数をエクスポートする場合、PSIMはDLLに関する情報を集めるためにそれを数回呼び出します。
これらの各呼び出しの説明は次のとおりです。

REQUESTUSERDATA(ACTION_DLL_SELECTED, REQUEST_BEGIN, 0, ptrUserData,
pnParam1, pnParam2, szParam1, szParam2):

pnParam1: 無視されます
pnParam2: 無視されます
szParam1(Read only): DLLのフルパスとファイル名。
szParam2(Write only): デフォルトの入力データのフルパスとファイル名。
これはptrUserDataにメモリを割り当てるのに適した場所です。 例えば、

```
*ptrUserData = new Internal_DLL_RuntimeData();
```

入力データファイルのデフォルトファイルが必要な場合は、そのフルパスとファイル名をszParam2にコピーします。 ファイルは存在しなければなりません、そうでなければ無視されます。 また、次回のREQUEST_PARAM_COUNTの呼び出しでは、入力データファイルを要求する必要があります。 この時点でファイルパスが設定されていると、このシーケンスが終了した後、
ACTION_INPUTFILE_CHANGEDの新しいシーケンスが実行されます。

REQUESTUSERDATA(ACTION_DLL_SELECTED, REQUEST_PARAM_COUNT, 0, ptrUserData,
pnParam1, pnParam2, szParam1, szParam2):

詳細については、nRequestCodeのセクションを参照してください。

REQUESTUSERDATA(ACTION_DLL_SELECTED, REQUEST_DATAFILE_INFO, 0, ptrUserData,
pnParam1, pnParam2, szParam1, szParam2):

詳細については、nRequestCodeのセクションを参照してください。

REQUESTUSERDATA(ACTION_DLL_SELECTED, REQUEST_PARAM_INFO, nRequestParam,
ptrUserData, pnParam1, pnParam2, szParam1, szParam2):

この関数は、「nRequestParam」の値が０から「パラメータ数－１」（前の呼び出しで設定されたパラメータ数）まで複数回呼び出されます。

各パラメータについて、ラベルまたは値をいつでも変更できます。
詳細については、nRequestCodeのセクションを参照してください。

REQUESTUSERDATA(ACTION_DLL_SELECTED, REQUEST_IN_OUT_NODES, 0, ptrUserData, pnParam1, pnParam2, szParam1, szParam2):
この関数は、入力ノードと出力ノードの数を取得および設定するために呼び出されます。 ノード数はいつでも変更できます。 場合によっては、ノード数は入力データファイルに依存します。その場合は、この呼び出しをここで無視してACTION_INPUTFILE_CHANGEDで処理することができます。
詳細については、nRequestCode のセクションを参照してください。

REQUESTUSERDATA(ACTION_DLL_SELECTED, REQUEST_INPUT_NODE_INFO, nRequestParam, ptrUserData, pnParam1, pnParam2, szParam1, szParam2):
詳細については、nRequestCodeとREQUEST_INPUT_NODE_INFOのセクションを参照してください。この関数は、nRequestParamの値を0から「入力ノード数 - 1」まで複数回呼び出されます。

REQUESTUSERDATA(ACTION_DLL_SELECTED, EQUEST_OUTPUT_NODE_INFO, nRequestParam, ptrUserData, pnParam1, pnParam2, szParam1, szParam2):
この関数は、nRequestParam の値を 0 から「出力ノード数 - 1」まで複数回呼び出されます。詳細については、nRequestCode のセクションを参照してください。

REQUESTUSERDATA(ACTION_DLL_SELECTED, REQUEST_END, 0, ptrUserData, pnParam1, pnParam2, szParam1, szParam2):
詳細については、nRequestCode のセクションを参照してください。

ACTION_ELEMENT_LOAD

この一連の呼び出しが実行される状況は 2 つあります。 1 つは、回路図ファイルがロードされていて、そのファイル内に一般的な DLL ブロックが存在する場合です。もう 1 つは、General DLL ブロックが貼り付けられているときのコピー/ペースト中です。

REQUESTUSERDATA(ACTION_ELEMENT_LOAD, REQUEST_BEGIN, 0, ptrUserData, pnParam1, pnParam2, szParam1, szParam2):
pnParam1(Read only): szParam1 の有効バイト数
pnParam2: 無視されます
szParam1(Read only): 保存時に DLL によって提供されるバイナリデータ。 詳細は ACTION_ELEMENT_SAVE を参照してください。
szParam2(Read, Write): 選択された入力ファイルパス ここでファイルパスを変更することができます。
これは ptrUserData にメモリを割り当てるのに適した場所です。 例えば、

```
*ptrUserData = new Internal_DLL_RuntimeData();
```

これは、各 DLL ブロックに対する PSIM から DLL への最初の呼び出しです。 DLL ブロックが新規の場合（[エレメント]メニューから選択して回路図ウィンドウに配置した場合）、ACTION_ELEMENT_LOAD を使用した一連の呼び出しは実行されません。

代わりに、ACTION_DLL_SELECTED を指定した一連の呼び出しが実行されます。

このシーケンスの残りを無視することもできます。その場合は、前のセッションで選択されてファイルに保存された値、またはコピーされた値（コピー/ペーストの場合）が使用されます。

次の呼び出しはこのシーケンスの残りの部分です。 詳細については、nRequestCode セクションを参照してください。

```
REQUESTUSERDATA(ACTION_ELEMENT_LOAD, REQUEST_PARAM_COUNT, 0, ptrUserData,
  pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_ELEMENT_LOAD, REQUEST_DATAFILE_INFO, 0, ptrUserData,
  pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_ELEMENT_LOAD, REQUEST_PARAM_INFO, nRequestParam,
  ptrUserData, pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_ELEMENT_LOAD, REQUEST_IN_OUT_NODES, 0, ptrUserData,
  pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_ELEMENT_LOAD, REQUEST_INPUT_NODE_INFO,
  nRequestParam, ptrUserData, pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_ELEMENT_LOAD, REQUEST_OUTPUT_NODE_INFO,
  nRequestParam, ptrUserData, pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_ELEMENT_LOAD, REQUEST_END, 0, ptrUserData, pnParam1,
  pnParam2, szParam1, szParam2);
```

ACTION_ELEMENT_SAVE

このシーケンスは、要素をファイルに保存するとき、またはコピー/貼り付け操作の一部として要素をコピーするときに呼び出されます。

DLL 固有の情報を保存するために、最大 100 バイトのサイズのバッファーが用意されています。 これは、DLL のバージョン、または表示されているパラメータの 1 つではない追加のパラメータです。 このバッファは、シーケンス ACTION_ELEMENT_LOAD の間に DLL に返されます。

```
REQUESTUSERDATA(ACTION_ELEMENT_SAVE, REQUEST_BEGIN, 0, ptrUserData,
  pnParam1, pnParam2, szParam1, szParam2):
    pnParam1(Write only): szParam1 の有効バイト数
    pnParam2:無視されます
    szParam1(Write only): .sch ファイルに保存するバイナリデータをコピーする
          (DLL バージョン、ファイルバージョンなど) (最大 100 バイト)
```

szParam2(Read only): 選択された入力ファイルパス

たとえば、次のコードは 25 バイトのユーザー定義バイナリデータをコピーします。

```
//copy 25 bytes of binary data.
memcpy(szParam1, mydata, 25);
*pnParam1 = 25;
```

```
REQUESTUSERDATA(ACTION_ELEMENT_SAVE, REQUEST_END, 0, ptrUserData, pnParam1,
  pnParam2, szParam1, szParam2);
```

ACTION_INPUTFILE_CHANGED

このシーケンスは、DLL ブロックのプロパティボックスで新しい入力データファイルが選択されるたびに実行されます。

```
REQUESTUSERDATA(ACTION_INPUTFILE_CHANGED, REQUEST_BEGIN, 0, ptrUserData,
  pnParam1, pnParam2, szParam1, szParam2):
```

pnParam1(Write only): pnParam1 を 0 に設定すると、このファイルを拒否することができます。 ファイルが拒否された場合は、ただちに REQUEST_END が呼び出され、入力ファイルパスの値が以前の値に戻されます。

pnParam2：無視されます。

szParam1(Read, Write): 選択した入力ファイルのフルパスとファイル名。このパラメータは読み書き可能です。パスとファイル名を変更することができます。

szParam2: 無視されます。

ファイルが拒否されなかった場合、以降の呼び出しはこのシーケンスの残りの部分です。

```
REQUESTUSERDATA(ACTION_INPUTFILE_CHANGED, REQUEST_PARAM_COUNT, 0,
  ptrUserData, pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_INPUTFILE_CHANGED, REQUEST_DATAFILE_INFO, 0,
  ptrUserData, pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_INPUTFILE_CHANGED, REQUEST_PARAM_INFO,
  nRequestParam, ptrUserData, pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_INPUTFILE_CHANGED, REQUEST_IN_OUT_NODES, 0,
  ptrUserData, pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_INPUTFILE_CHANGED, REQUEST_INPUT_NODE_INFO,
  nRequestParam, ptrUserData, pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_INPUTFILE_CHANGED, REQUEST_OUTPUT_NODE_INFO,
  nRequestParam, ptrUserData, pnParam1, pnParam2, szParam1, szParam2);
REQUESTUSERDATA(ACTION_INPUTFILE_CHANGED, REQUEST_END, 0, ptrUserData,
  pnParam1, pnParam2, szParam1, szParam2);
```

ACTION_PARAMETERS_CHANGED

このシーケンスは、プロパティボックス内のデータが変更されるたびに実行されます。

REQUESTUSERDATA(ACTION_PARAMETERS_CHANGED, REQUEST_BEGIN, 0, ptrUserData, pnParam1, pnParam2, szParam1, szParam2):

pnParam1: 無視されます。

pnParam2: 無視されます。

zzParam1: 無視されます。

szParam2: 無視されます。

REQUESTUSERDATA(ACTION_PARAMETERS_CHANGED, REQUEST_PARAM_INFO, nRequestParam, ptrUserData, pnParam1, pnParam2, szParam1, szParam2):

この関数は、nRequestParam を 0 から 'number of parameters - 1'に設定して複数回呼び出されます。

nRequestParam: ゼロベースのパラメータインデックス

pnParam1: 無視されます。

pnParam2: 無視されます。

szParam1(Read Only): パラメータラベル

szParam2(Read, Write): パラメータ値。値は変更できますが、拒否することはできず、この機能でメッセージボックスをポップアップ表示することはできません。

REQUESTUSERDATA(ACTION_PARAMETERS_CHANGED, REQUEST_END, 0, ptrUserData, pnParam1, pnParam2, szParam1, szParam2):

ACTION_ELEMENT_DELETE

このシーケンスは、要素が削除されているとき、または回路図ファイルが閉じられているときに実行されます。作成されたメモリリソースはすべて解放する必要があります。

REQUESTUSERDATA(ACTION_ELEMENT_DELETE, REQUEST_BEGIN, 0, ptrUserData, pnParam1, pnParam2, szParam1, szParam2):

pnParam1: 無視されます。

pnParam2: 無視されます。

zzParam1: 無視されます。

szParam2: 無視されます。

これは、割り当てられたメモリを解放するための適切な場所です。 例えば、

```
delete (Internal_DLL_Block_RuntimeData *)(*ptrUserData);
*ptrUserData = NULL;
```

REQUESTUSERDATA(ACTION_ELEMENT_DELETE, REQUEST_END, 0, ptrUserData, pnParam1, pnParam2, szParam1, szParam2):

3. Examples(範例)

一般的なDLLブロックの使用方法を説明するために3つの例が、以下のように提供されています。（PSIMメニュー：「ファイルー範例を開くーcustom dll」にあります。）

3.1. TestBlock (Files: general_dll_block_test1.sch, TestBlock.dll):

この例では、入力データファイルから入出力ノード数とノードラベルを読み込みます。DLLブロックが回路図ファイル（ACTION_ELEMENT_LOAD）から読み込まれるか、入力ファイルが変更される（ACTION_INPUTFILE_CHANGED）と、DLLはファイルを読み込み、その内部データ用にメモリを再割り当てします。

Genaral DLLブロックには10個のパラメータがあります。 ACTION_DLL_SELECTED：REQUEST_PARAM_COUNTを処理して新しい要素が作成されたときのパラメータ数を設定します。 それはREQUEST_PARAM_COUNTへの他のすべての呼び出しを無視します、それで要素の寿命を通してそして複数のシミュレーションセッションを通してパラメータの数の元の値を保持します。

3.2. TestBlock2 (Files: general_dll_block_test2.sch, TestBlock2.dll):

この例では、汎用DLLブロックに1つの入力ノードと2つの出力ノードがあります。 入力データファイルと3つのパラメータ（整数、浮動小数点値、文字列）があります。 文字列パラメータは、出力ファイル名として使用されます。 シミュレーション中に、DLLは入力データファイルを開いて読み取ります。 入力データファイルの値とパラメータを使用して計算を実行します。 DLLは結果を出力データファイルに書き込みます。

3.3. TestBlock3 (Files: general_dll_block_test3.sch, TestBlock3.dll):

この例では、DLLはユーザーインターフェイス関数REQUESTUSERDATAをエクスポートしません。 要素のプロパティボックスでは、ユーザーは入力ノードと出力ノードの数を入力できます。 5つのパラメータと入力データファイルもあります。 この例では、DLLがPSIMからパラメータ値と入力データファイルパスを取得する方法を示します。 ただし、これらのパラメータと入力データファイルはDLL内では使用されず、空白のままになります。 この例では、汎用DLLブロックに少なくとも1つの入力ノードと1つの出力ノードがあると想定しています。

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. (本資料に記載された製品をユーザ装置に組み込む際には、バックアップやフェイルセーフ機能をシステム的に設置してください。)
7. (弊社は、人命に関わる装置として特別に開発したものは用意しておりません。)
8. 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。

Copyright 2019 by Myway Corporation
All rights reserved. No part of this manual may be photocopied or reproduced in any form or by any means without the written permission of Myway Corporation.

発行:Myway プラス株式会社
〒220-0022
横浜市西区新花咲町 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