モータ制御開発支援システム
Ｓｉｍｔｒｏｌ-m

技 術 資 料

2006-04

株式会社 昭和電業社

目 次

１、製品概要

ＫＥＮＴＡＣ　Ｓｉｍｔｒｏｌ－ｍ（以下Ｓｉｍｔｒｏｌ-mと記す）は、ダイナミックシステムのプラットフォームです。
グラフィカルな環境と実装されたブロック・ライブラリ群により、制御信号処理及び時間依存システムの正確な設計・シミュレーション・実装・テストが可能です。

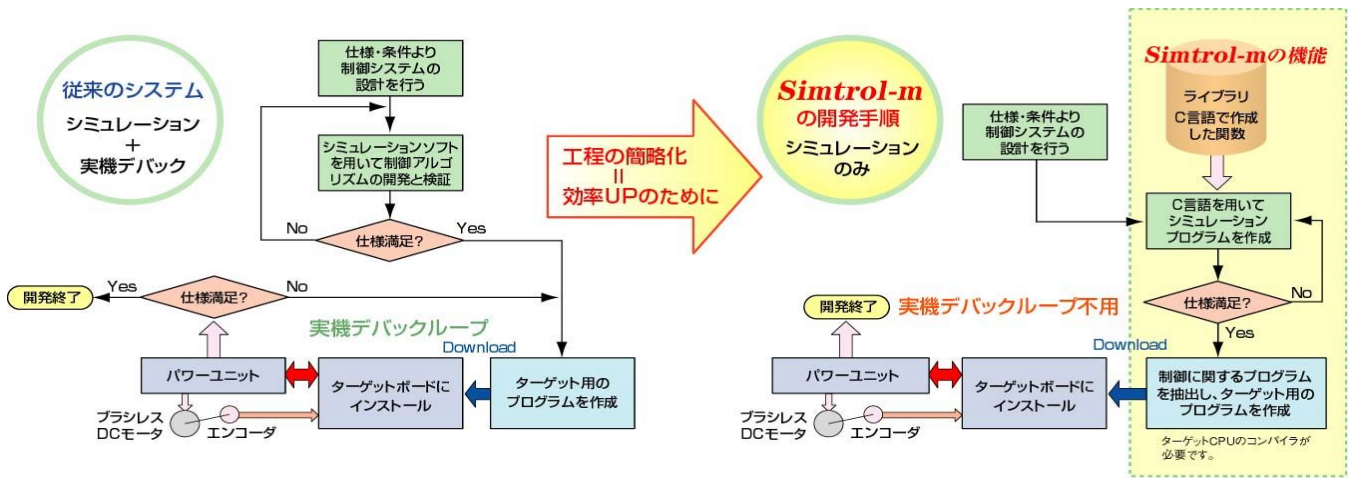
１．１　主な機能／特徴

・モータ制御に必要なブロックがライブラリ化されて用意されています。
・各ブロックには、それぞれ固有のパラメータを設定できます。
・ライブラリに搭載されたブロックを使用して、制御ブロックダイアグラムを構築し、シミュレーションが実行できます。
・ブロックダイアグラム上の必要と思われる箇所にモニタ装置を挿入し、シミュレーションの結果を表示させることができます。任意の箇所でのプロセス観測も可能です。
・シミュレーション結果が良好なら、コンパイルし、ブロックダイアグラムの内容を反映したＣ言語のソース・コードを生成します。ここで生成されるＣ言語はＡＮＳＩ準拠ですので、ほとんどのＣＰＵのＣ言語コンパイラで実行が可能です。
・ターゲット・ボードの制御には、ハードウェアの機能に依存する関数も必要です。この部分は基本的にお客様の製作となりますが、当社製のＫＥＮＴＡＣ-ＣＰＵボードに関するものは搭載済みです。

１．２　Ｓｉｍｔｒｏｌ-mを用いた場合の開発手順（例）

① モータの制御システムを設定する。
② Ｓｉｍｔｒｏｌ-mのグラフィカル環境及びブロック・ライブラリ等を利用し、制御用のブロックダイアグラムを作成。
③ 各ブロック要素の内部パラメータ・データを設定。
④ パソコン上でＳｉｍｔｒｏｌ-mによるシミュレーションを実施。
⑤ シミュレーション結果が満足なら、Ｓｉｍｔｒｏｌ-mでコンパイルを実施することで、ＡＮＳＩ準拠のＣ言語コードを抽出。
⑥ モータ制御ターゲット用のＣコンパイラでコンパイルを実施し、ターゲット・ボードに実装。
⑦ ターゲット・ボードでモータを駆動してみる。
⑧ 駆動結果をグラフ化し、観測してみる。
⑨ シミュレーション結果と比較し、良好ならば開発完了。

１．３　従来工程との比較（例）

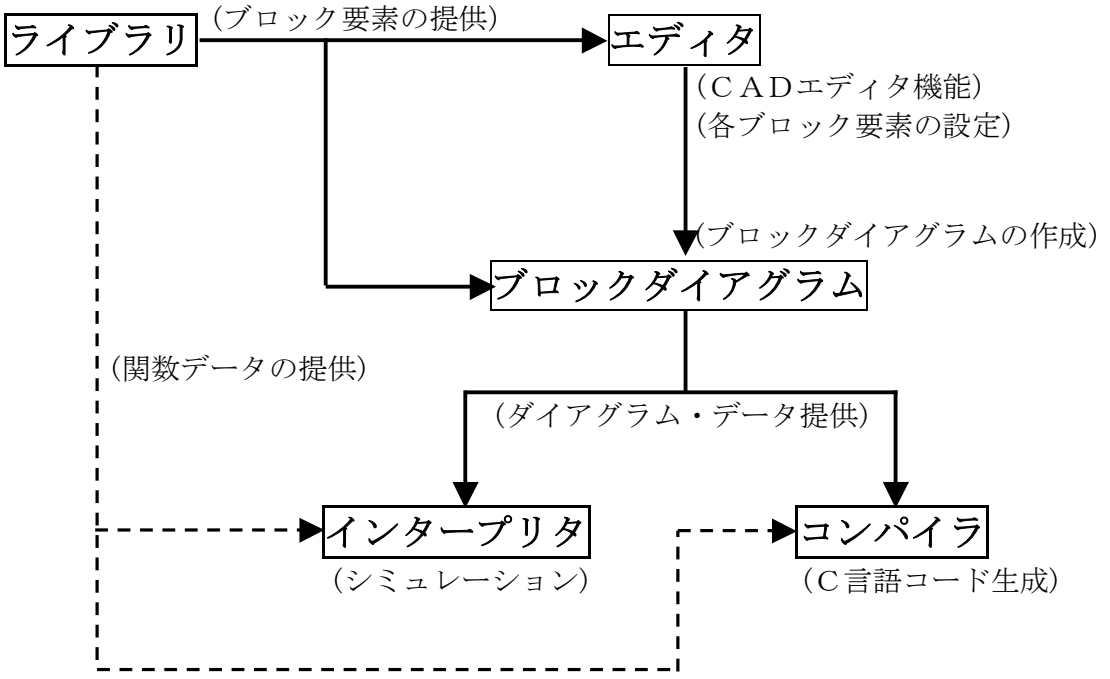


２、機能概説

　Ｓｉｍｔｒｏｌ－ｍは、ＰＣの画面上にターゲット・システムのブロックダイグラムを作成し、シミュレーションを実行し、かつ、対応するソース・コードをＣ言語で作成します。
　Ｓｉｍｔｒｏｌ―ｍでは、
①ブロックダイアグラムを作成・編集する画面機能を、「エディタ」、
②シミュレーションを設定・実行・表示する機能を、「インタープリタ」、
③Ｃ言語ソース・コードを生成・表示する機能を、「コンパイラ」、
　と呼びます。以下、この呼称を使用します。
　また、ブロックダイアグラムを作成するために必要なブロックは、専用のライブラリに格納／管理されています。エディタでは、必要なブロックを任意にライブラリから呼び出して、ブロックダイアグラムを構成します。
以下、この機能を、「ライブラリ」と呼びます。

以上の機能間の関連をまとめると、下図(図－１)のようになります。

Ｓｉｍｔｒｏｌ－ｍの各機能と関連



図－１（各機能の関連）

３、ライブラリ機能

　２００６年４月現在、Ｓｉｍｔｒｏｌ－ｍが搭載している、主なライブラリ関数を示します。

①ベクトル制御用関数（モータにおける座標系変換処理を行う関数）
３相３線→ｄｑ座標系変換、３相２線→ｄｑ座標系変換、ｄｑ座標系→３相３線変換、
ｄｑ座標系→３相２線変換。
αβ座標系→ｄｑ座標系変換、ｄｑ軸間非干渉化関数、ｄｑ座標系→αβ座標系変換。
αβ座標系→３相３線変換、３相３線→αβ変換など。

②プロセス制御用関数
ＰＩＤ制御器、ＰＩ-Ｄ制御器、Ｉ-ＰＤ制御器、ＰＩ制御器、Ｐ制御器、Ｉ制御器、Ｄ制御器、アンチリセットワインドアップＰＩＤ制御器など。

③演算関数（アナログ演算）
加算器、減算器、リミット、４入力加減算器、増幅器(乗算器)、マトリックス演算(２行２列)、
逆マトリックス演算(２行２列)、マトリックス演算(２行３列)、マトリックス演算(３行２列)など。

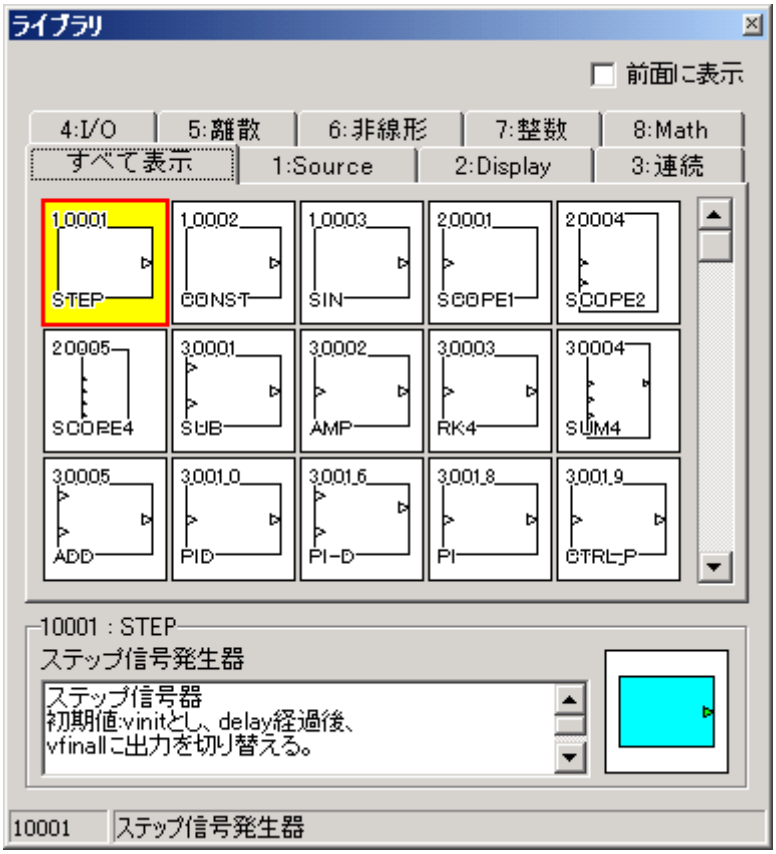
④モータ用関数（等価システム関数）
ブラシ付ＤＣモータ、ブラシレスＤＣモータなど。

⑤信号発生器
ステップ信号発生器、定数信号発生器、正弦波発生器、方形波発生器、三角波発生器、
ノイズ信号発生器、ランプ発生器、平衡３相信号発生器、任意波形発生器など

⑥ディスプレイ関数（シミュレーション結果のモニタ装置：シミュレーション時のみ使用）
１入力オシロスコープ、２入力オシロスコープ、４入力オシロスコープ。
バーグラフ(１入力)。数値表示(１入力)。

⑦ＰＩＯ関数（ハードウェアに依存する機能。当社製ＣＰＵボードに関するもの）
ＫＥＮＴＡＣ１３６００用　A/Dコンバータ、D/Aコンバータ、PWM発生器、エンコーダ。
ＫＥＮＴＡＣ１３５００用　A/Dコンバータ、D/Aコンバータ、PWM発生器、エンコーダ。
ＫＥＮＴＡＣ８００Ｈ８Ｓ用A/Dコンバータ、D/Aコンバータ、PWM発生器、エンコーダ。

上記の各関数が各々のブロックと対応つけられて、ライブラリ内に格納されています。ライブラリの表示画面を下図(図－２)に示します。この画面は直接、ブロックダイアグラムの作成に利用できます。その他、必要なときに呼び出すことが可能です。



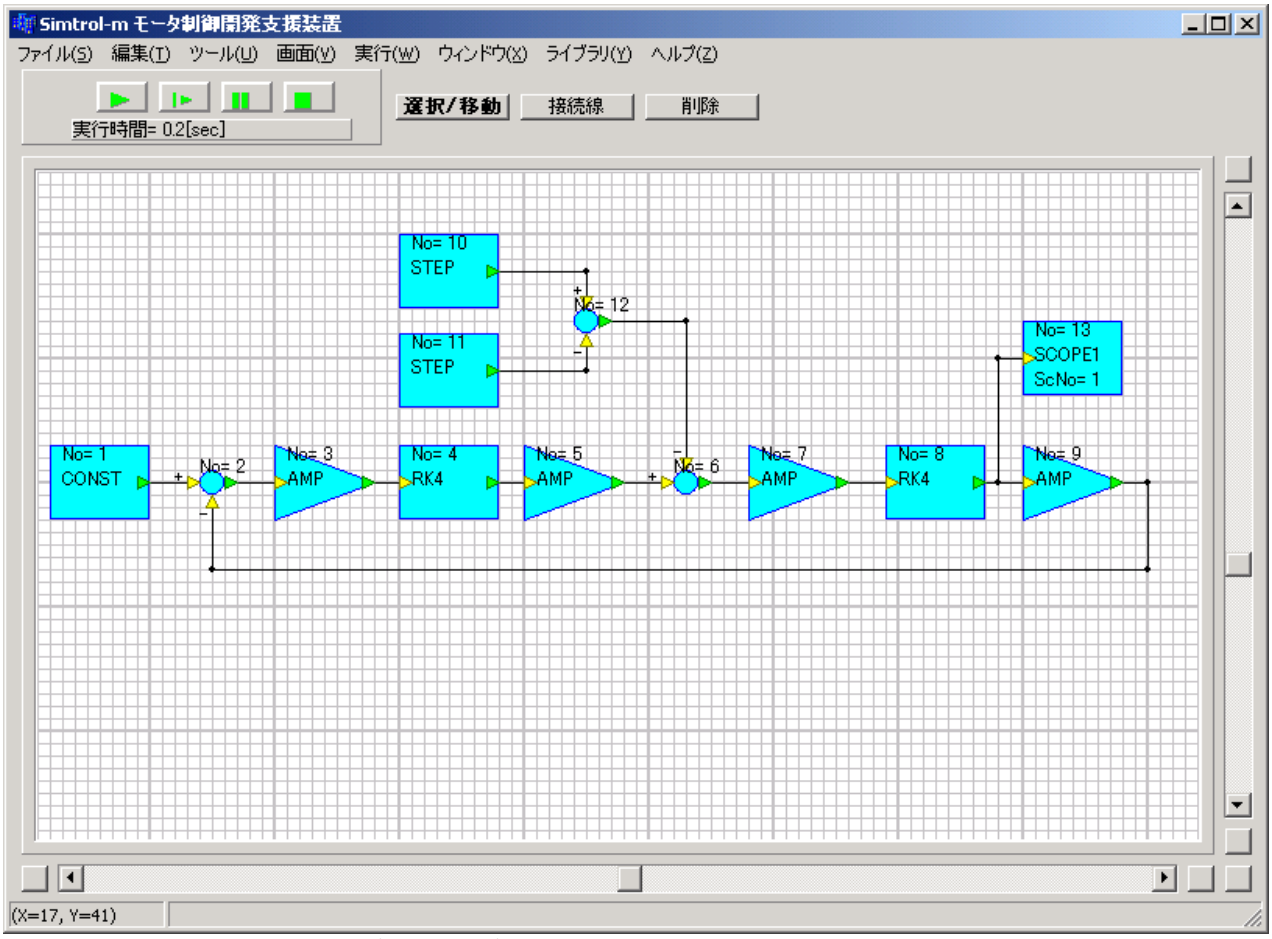
図－２（ライブラリの表示画面）

４、操作概説

４．１、エディタ（ブロックダイアグラムの作成）

最初にエディタを用いて、ブロックダイアグラムを作成します。
ライブラリ画面か、メニューバーの[ライブラリ]かのいずれかから、必要なブロックを選択し、エディタ画面上に貼り付けます。
任意のブロックの端子間を接続線でつなぎ、ブロックダイアグラムを作成していきます。
ブロック同士は接続線で接続されるとデータを入出力する関係となり、数学的にも結合されます。
必要なブロックの接続を終えると、ブロックダイアグラムは完成です。

今回はＤＣモータの、入力電圧及び負荷トルクに対する速度を求めるものとします。
まず、ＤＣモータとその駆動回路(ＤＣ電源)を等価なブロックダイアグラムで表現します。
図-３にＳｉｍｔｒｏｌ－ｍで作成した、ＤＣモータの等価ブロックダイアグラムを示します。



図－３（ＤＣモータの等価ブロックダイアグラム）

図-３のブロックダイアグラムでは、ＤＣ電源の電圧を定数(CONST:No1)とし、
モータの電機子抵抗(AMP:No3)／電機子インダクタンス(RK4:No4)／トルク定数(AMP:No5)から、
モータが発生するトルク(No5 出力)を求めます。一方、外乱要因である STEP(No10)／STEP(No11)は
モータの負荷を表わします。ここでは負荷がステップ状に変化した状態(No12 出力)です。
モータが発生したトルク(No5 出力) から負荷トルク分(No12 出力)を差し引けば(No6)、モータ自体の
回転に寄与するトルク(No6 出力)を求めることができます。
次に軸受け等の粘性制動係数(AMP:No7)及び、ロータの慣性モーメント(RK4:No8)を考慮し、
モータの回転速度(No8 出力)を求め、オシロスコープ(SCOPE1:No13)で表示します。
最後に、ＤＣモータの回転によって電機子に生じる、逆起電力分の電圧(AMP:No9 の出力)が、
電源とは逆向きに作用(No2)しています。
（但し、ＡＭＰ：増幅器（定数乗算器）、ＲＫ４：１次遅れ要素（ルンゲ・クッタ４次法）。）

なお、図－３の各ブロックのパラメータと、値の例を下表(表－１)に示します。

ブロック	種類	パラメータ	設定値	計算式
No.1	CONST	constant	12	Va
No.3	AMP	gain	1	1/Ra
No.4	RK4	τ	0.01	La/Ra
No.5	AMP	gain	0.05	KT
No.7	AMP	gain	100000	1/D
No.8	RK4	τ	5	J/D
No.9	AMP	gain	0.05	KE
No.10	STEP	delay	0.08	
No.10	STEP	vinit	0	
No.10	STEP	vfinal	0.15	TL
No.11	STEP	delay	0.14	
No.11	STEP	vinit	0	
No.11	STEP	vfinal	0.15	TL
No.13	SCOPE1	tmin	0	
No.13	SCOPE1	tmax	0.2	
No.13	SCOPE1	tgrid	0.05	
No.13	SCOPE1	ymin	0	
No.13	SCOPE1	ymax	250	
No.13	SCOPE1	ygrid	50	
No.13	SCOPE1	ScopeNo	1	
No.13	SCOPE1	buffsize	2000	

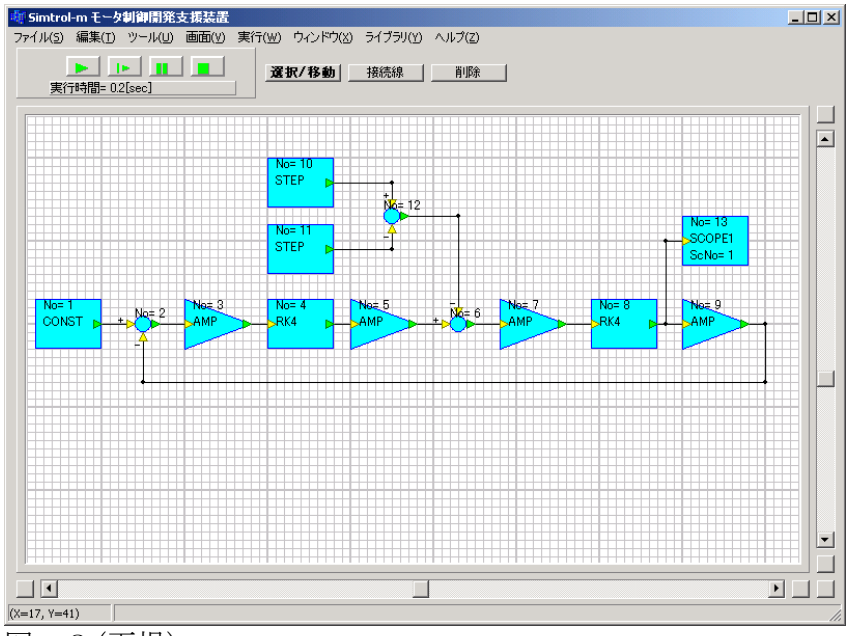
表-１(図-３の各パラメータ値一覧)

Va = 12.0 ：モータ供給電圧[V]
Ra = 1.0 ：モータ電機子抵抗[Ω]
La = 0.01 ：電機子インダクタンス[H]
KT = 0.05 ：トルク定数[Nm/A]
KE = 0.05 ：逆起電力定数[Vs/rad]
TL = 0.15 ：負荷トルク[Nm]
D = 0.00001 ：粘性制動係数[Nm/rad]
J = 0.00005 ：慣性モーメント[kgm2]
(表-１付記)

４．２　インタープリタ（シミュレーション）

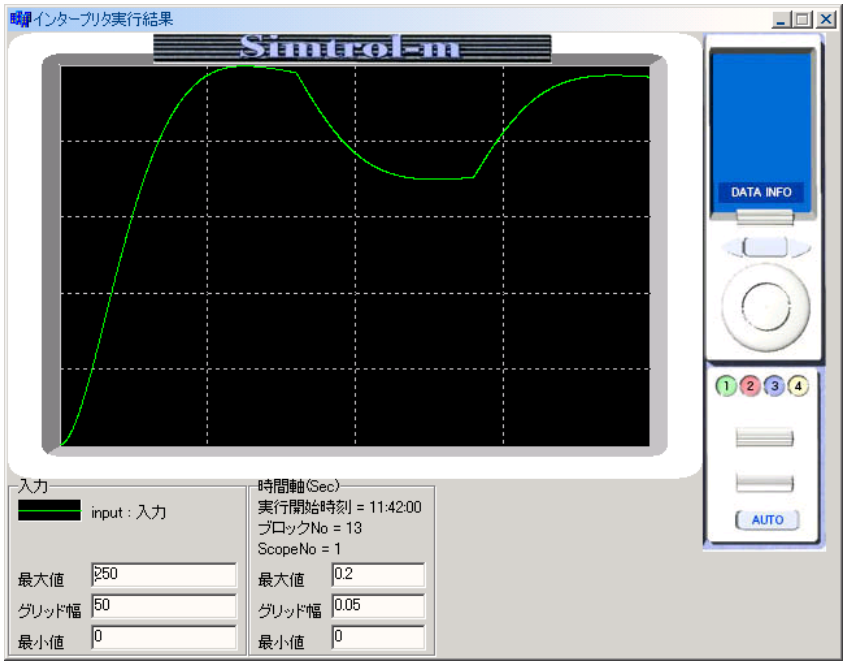
ブロックダイアグラムが作成できたら、インタープリタでシミュレーションを実行してみます。はじめに、メニューバーの[実行]－[インタープリタ設定]コマンドで、インタープリタの実行に必要な設定（実行時間などの設定）をします。

次に個々のブロックの内部パラメータ（定数、時間など）を設定します。各設定が終了したら、インタープリタを実行します。図－３のＤＣモータのダイアグラムでインタープリタを実行してみます。



図－３（再掲）

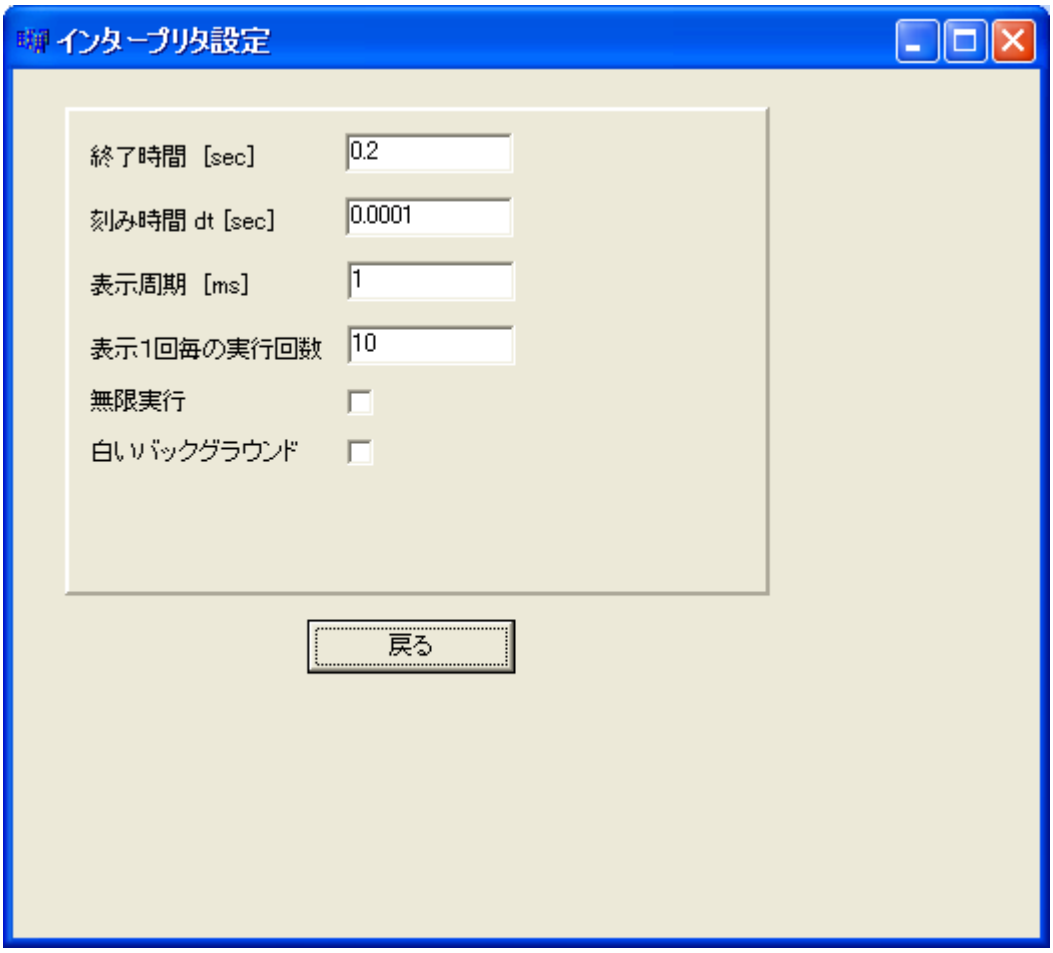
この例では、一定入力電圧と外乱（負荷トルク変動）に対する、ＤＣモータの回転速度の時間変化をシミュレーションすることができます。インタープリタの実行結果を下図（図－４）に示します。



図－４（インタープリタ実行結果：例）

図－４は、0 秒にＤＣモータの電機子に電圧が印加されたときの、0 秒から 0.2 秒までのＤＣモータの回転速度の変化をシミュレーションしたものです。但し、この例では、0.08 秒～0.14 秒の間、一定値の負荷トルクが印加されているため、0.08 秒から回転に変化が生じています。

また、インタープリタの実行設定は、専用の画面を呼び出して行います。
インタープリタの実行設定用の画面を以下に示します。（図－５）



図－５（インタープリタ設定画面）

インタープリタの実行で設定可能な各パラメータの内容は以下の通りです。

① 終了時間
　シミュレーションの実行時間の設定。
② 刻み時間
　シミュレーション実行時の刻み時間を設定。
③ 表示周期
　シミュレーション画面(図－４など)での表示処理の周期を設定。
　（画面上の時間軸表示の周期ではありません。）
④ 表示一回毎の実行回数
　１表示周期中のインタープリタ処理の実行回数を設定。

４．３　コンパイラ（Ｃ言語コード作成）

良好なシミュレーション結果が得られたら、コンパイラでコンパイルしてみます。
前記図－３のブロックダイアグラムをコンパイルすると、以下のようなＣ言語のソース・コードが得られます。

実際にＤＣモータを制御・駆動するには、ＤＣモータの駆動を制御するＣＰＵボード(ターゲット・ボード)用のＣ言語コンパイラで、このリストをコンパイルします。得られたモジュールをボード上に転送して実行すれば、前記インタープリタでのシミュレーション結果と同じ結果が得られます。
モジュールのＲＯＭ化等も可能です。

但し、他のＣＰＵ上での利用を前提としているソース・コードであるため、プリプロセッサやヘッダ・ファイルなど、コンパイラ環境ごとに異なる部分は含みません。その点はソース・コードへの補償が必要です。

なお、ソース・コードには簡単な内容説明のコメントが付加されます。（付加/否の選択可）
また、ソース・コードは任意のファイル名で保存することが可能です。

Ｓｉｍｔｒｏｌ－ｍでの、図－３のコンパイル結果（Ｃ言語ソース・コード）(1/2)

```
/****************************************/
/* Simtrol-m by SHOWA DENGYOSHA CO.,LTD. */
/****************************************/

/**************/
/* 変数の宣言 */
/**************/
/* 線変数の宣言 */
float x1 ; float x2 ; float x3 ; float x4 ; float x5 ;
float x6 ; float x7 ; float x8 ; float x9 ; float x10 ;
float x11 ; float x12 ;
/* スコープ類の変数宣言 */
int *counter13 ;
int dataSize13 ;
float data13[2000] ;
/* プロパティ・データ領域の宣言 */
float pro1[1], pro3[1], pro4[1], pro5[1], pro7[1] ;
float pro8[1], pro9[1], pro10[3], pro11[3], pro13[8] ;
/* 処理用変数の宣言 */
int err ;    /* エラー処理用変数 */

/****************/
/* 変数の初期化 */
/****************/
x1 = 0 ; x2 = 0 ; x3 = 0 ; x4 = 0 ; x5 = 0 ;
x6 = 0 ; x7 = 0 ; x8 = 0 ; x9 = 0 ; x10 = 0 ;
x11 = 0 ; x12 = 0 ;
counter13 = 0 ;
dataSize13 = 2000 ;

/************************/
/* プロパティー値の設定 */
/************************/

/* CONST : 定数信号器 */
pro1[0] = 12 ;        /* constant : 定数 */

/* AMP : 増幅器(定数乗算器) */
pro3[0] = 1 ;         /* gain : 増幅率 */
```

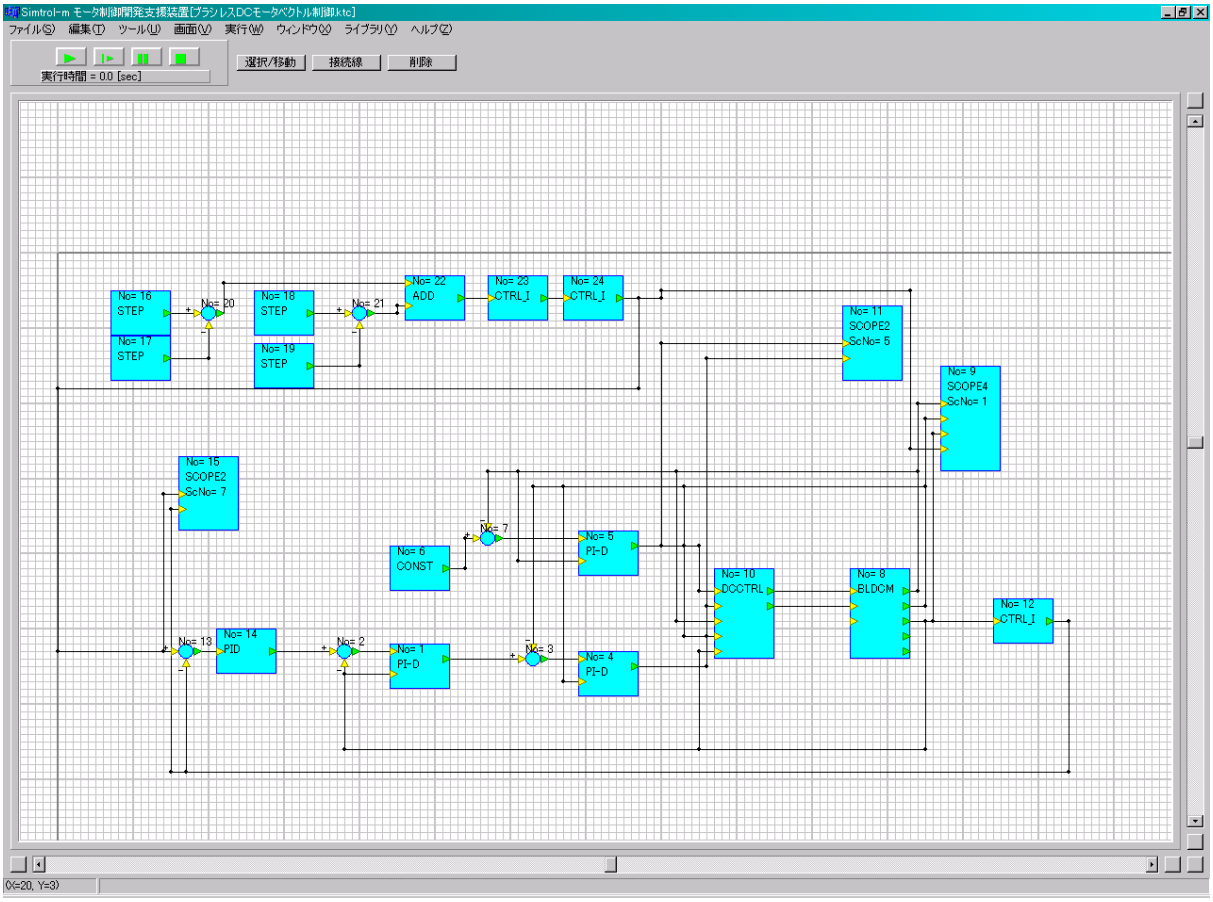
(次ページへ続く)

```
/* RK4 : １次遅れ要素(ルンゲ・クッタ４次法) */
pro4[0] = 0.01 ;      /* tau :  時定数[sec] */

/* AMP : 増幅器(定数乗算器) */
pro5[0] = 0.05 ;      /* gain : 増幅率 */

/* AMP : 増幅器(定数乗算器) */
pro7[0] = 100000 ;    /* gain : 増幅率 */

/* RK4 : １次遅れ要素(ルンゲ・クッタ４次法) */
pro8[0] = 5 ;         /* tau :  時定数[sec] */

/* AMP : 増幅器(定数乗算器) */
pro9[0] = 0.05 ;      /* gain : 増幅率 */

/* STEP : ステップ信号発生器 */
pro10[0] = 0.08 ;     /* delay : 切り替え時間[sec] */
pro10[1] = 0 ;        /* vinit : 初期値 */
pro10[2] = 0.15 ;     /* vfinal : 切り替え後の値 */

/* STEP : ステップ信号発生器 */
pro11[0] = 0.14 ;     /* delay : 切り替え時間[sec] */
pro11[1] = 0 ;        /* vinit : 初期値 */
pro11[2] = 0.15 ;     /* vfinal : 切り替え後の値 */

/* SCOPE1 : オシロスコープ(１入力) */
pro13[0] = 0 ;        /* tmin : 時間軸最小値[sec] */
pro13[1] = 0.2 ;      /* tmax : 時間軸最大値[sec] */
pro13[2] = 0.05 ;     /* tgrid : 時間軸目盛り刻み[sec] */
pro13[3] = 0 ;        /* ymin : y 軸最小値 */
pro13[4] = 250 ;      /* ymax : y 軸最大値 */
pro13[5] = 50 ;       /* ygrid : y 軸目盛り刻み */
pro13[6] = 1 ;        /* ScopeNo : スコープ番号 */
pro13[7] = 2000 ;     /* buffsize : バッファサイズ */

err = 0 ;    /* エラー変数の初期化 */

/************/
/* 実行処理 */
/************/
fn_const( pro1 , &x1 , &err ) ;          /* CONST : 定数信号器 */
fn_sub( x1 , x9 , &x2 , &err ) ;         /* 減算 */
fn_amp( x2 , pro3 , &x3 , &err ) ;       /* AMP : 増幅器(定数乗算器) */
fn_rk4( x3 , pro4 , &x4 , &err ) ;       /* RK4 : １次遅れ要素(ルンゲ・クッタ４次法) */
fn_amp( x4 , pro5 , &x5 , &err ) ;       /* AMP : 増幅器(定数乗算器) */
fn_sub( x5 , x12 , &x6 , &err ) ;        /* 減算 */
fn_amp( x6 , pro7 , &x7 , &err ) ;       /* AMP : 増幅器(定数乗算器) */
fn_rk4( x7 , pro8 , &x8 , &err ) ;       /* RK4 : １次遅れ要素(ルンゲ・クッタ４次法) */
fn_amp( x8 , pro9 , &x9 , &err ) ;       /* AMP : 増幅器(定数乗算器) */
fn_step( pro10 , &x10 , &err ) ;         /* STEP : ステップ信号発生器 */
fn_step( pro11 , &x11 , &err ) ;         /* STEP : ステップ信号発生器 */
fn_sub( x10 , x11 , &x12 , &err ) ;      /* 減算 */
fn_scope1( x8 , pro13 , &counter13 , dataSize13 , data13 , &err ) ;  /* SCOPE1 : オシロスコープ(１入力) */
```

５、Ｓｉｍｔｒｏｌ-mによるブラシレスＤＣモータの駆動例

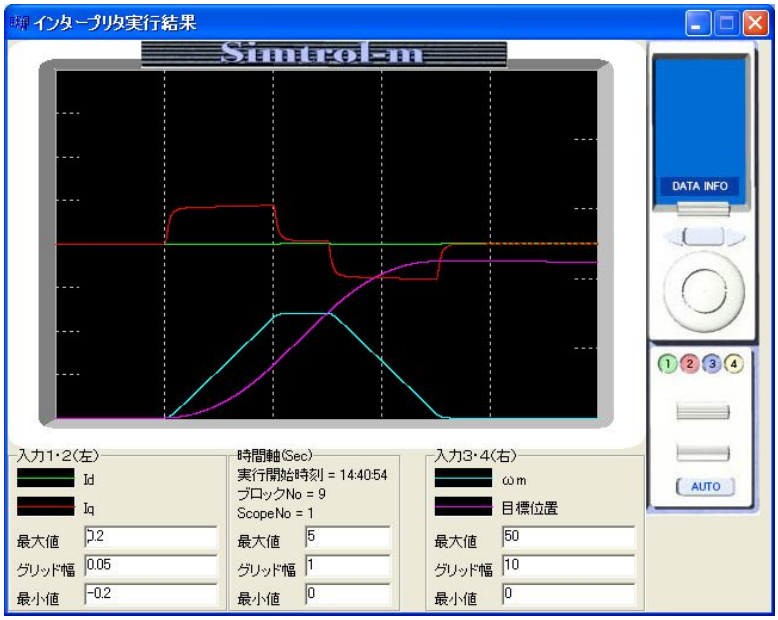
下図(図－6)に Simtrol-mで作成した、ブラシレス DC モータの位置決め制御のダイアグラムの例を示します。モータのd軸／q軸の電圧を制御することで、ブラシレス DC モータ(BLDCM:No9)の回転数を制御します。

また、d軸／q軸の電圧による制御ではd軸／q軸での速度起電力による干渉が生じます。その影響を除くため、モータに供給されるd軸電圧／q軸電圧を、回転速度／d軸電流／q軸電流で補償する制御(非干渉化制御)を実行しています



図－6 （ブラシレスDCモータの位置決め制御のブロックダイアグラムの例）

下図(図－7)は、インタープリタによる、図－6のシミュレーション実行結果の一例です。

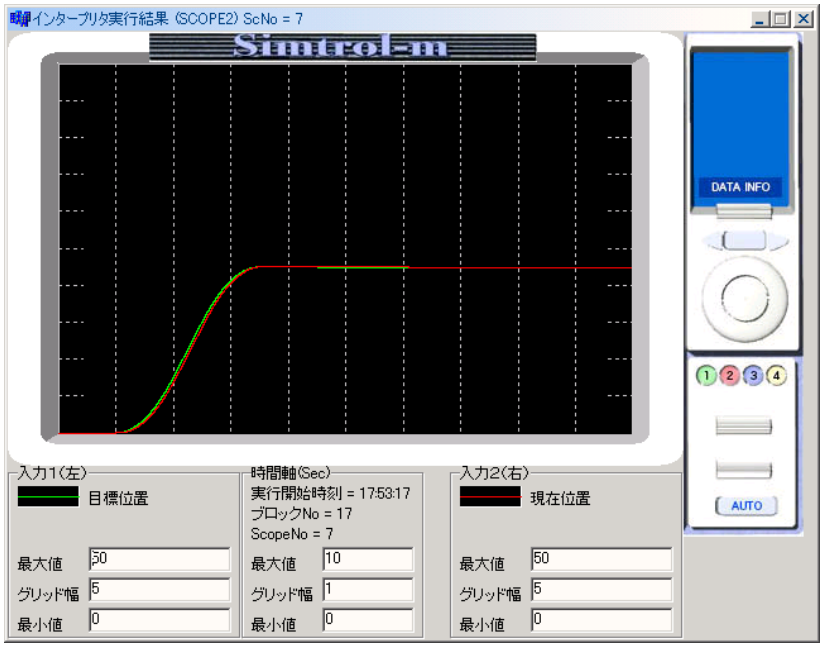


図－7:図－6のインタープリタ実行結果 （SCOPE4:No9 ブロック）

図－7で、SCOPE4(No9)ブロックへの入力1・入力 2 はブラシレス DC モータ BLDCM(No8)のd軸電流 Id q軸電流 Iq、同じく入力3・入力 4 は BLDCM の回転角速度 ωm 及び、制御の目標位置(角度)です。

図－6では得られたモータの角速度 ωm 値を I 制御要素(CTRL_I:No12)に入力しています。I 制御要素の出力(積分値)は現在の位置情報(モータのロータ角度)です。図-1 の SCOPE2(No15)ブロックでは、ωm 値から得られた位置(角度)の実測値と、前述の制御目標値とが入力されており、各々を表示できます。

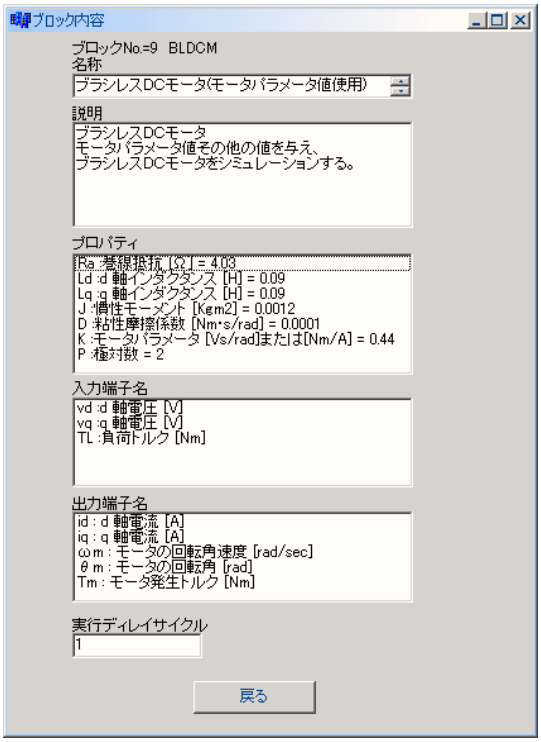
図－6の SCOPE2(No15)ブロックでのシミュレーション結果の一例を、下図(図-7)に示します。



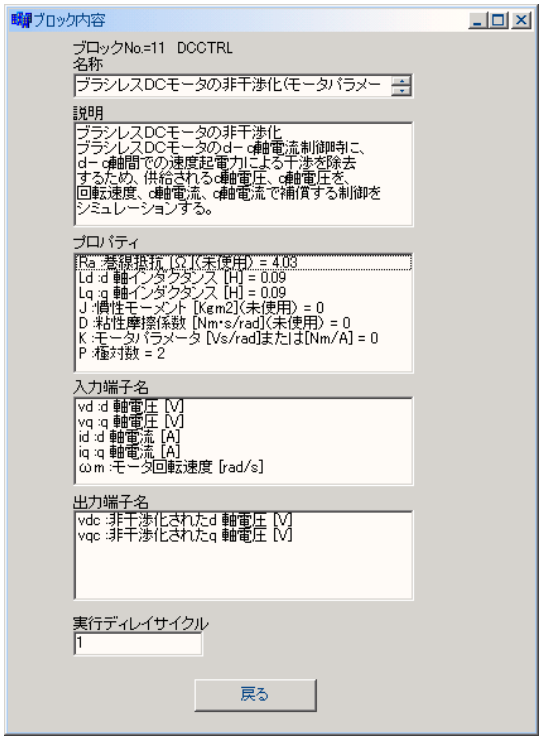
図－8:図－6のインタープリタ実行結果 (SCOPE2:No15 ブロック)

図－8では、目標値は緑色の線で、実測値(現在位置)は赤色の線で示されています。
目標値に対し、実測値がほぼ完全に追従しています。

この場合のブラシレス DC モータ:BLDCM(No8)ブロック及び、dq軸間非干渉化:DCCTRL(No10)ブロックのパラメータ設定を下図(図-9)に示します。駆動する条件やモータが変われば、順次、設定をかえていきます。



図－9(a):BLDCM ブロックの設定内容



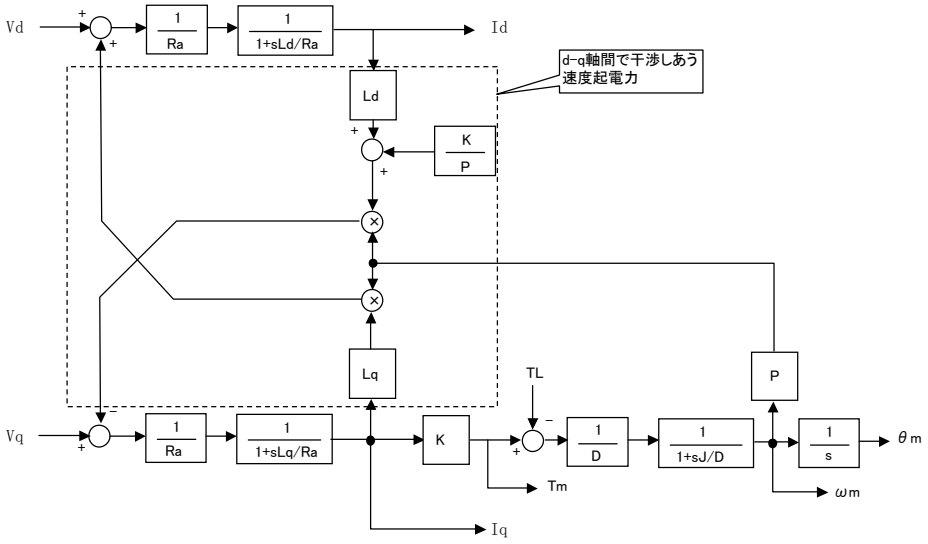
図－9(b):DCCTRL ブロックの設定内容

６、ブロック内演算内容

ブロック内部の演算処理の一例として、図－６で紹介した、ブラシレスＤＣモータ：ＢＬＤＣＭブロック及び、ｄ－ｑ軸間非干渉化：ＤＣＣＴＲＬブロックの内部演算内容を紹介します。

６．１、ＢＬＤＣＭブロック内演算内容

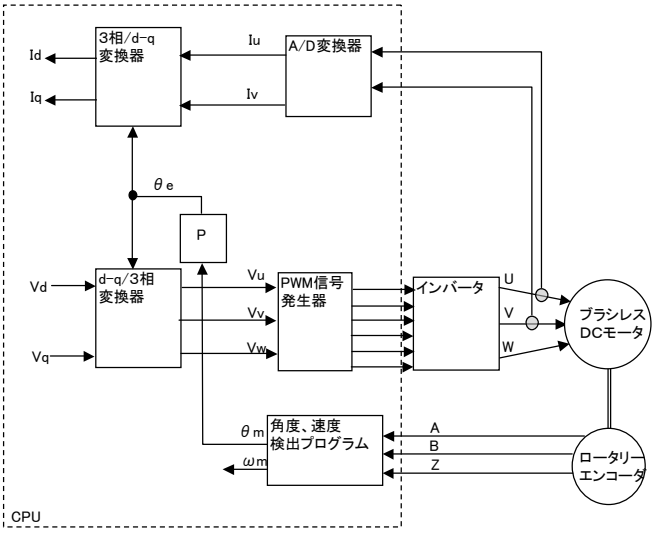
ＢＬＤＣＭブロックはブラシレスＤＣモータをシミュレーションするブロックです。
このブロックで扱うブラシレスＤＣモータは３相交流モータですが、d-q座標系（直交回転座標系）に座標変換を行うことで等価的に直流量として扱うことが出来ます。　（参考文献参照）
設定されたモータパラメータからブラシレスＤＣモータを直流量でシミュレーションします。
モータの電圧/電流のｄ軸/ｑ軸成分、VdとVq、IdとIqはそれぞれ直交しております。
Idはトルクを発生しない電流成分、Iqはトルクを発生する電流成分です。d-q軸間にはお互いに干渉しあう速度起電力があります。



図－１０：ブラシレスＤＣモータの直流量シミュレーション

６．２　ＢＬＤＣＭ１３６００ブロック演算内容

BLDCM13600はルネサス製SH7085搭載の32bitマイコンボード(型式 KENTAC 13600)専用のブロックです。
シミュレーション時はBLDCMと同様にブラシレスＤＣモータを直流量でシミュレーションします。
コントロール時はd-q軸電圧をU・V・Wの各デューティ比に変換してＰWM出力し、
発生したU-V電流をd-q電流に変換します。



図－１１：BLDCM13600ブロック演算内容

６．３　DCCTRLブロック演算内容

ブラシレスＤＣモータのｄ－ｑ軸電流制御を行うときに，ｄ－ｑ軸間で干渉しあう速度起電力があります。この速度起電力はｄ軸電流，ｑ軸電流に影響を及ぼすため，速度起電力を求めて，干渉する成分を除去することで干渉を防止できます。（参考文献参照）この関数は，供給されるｄ軸電圧，ｑ軸電圧を回転速度，ｄ軸電流、ｑ軸電流で補償するものです。

計算式

$\omega e = \omega m \cdot P$　（ωｍ：モータ回転数/Ｐ：極対数）

補償後のｄ軸電圧 $Vdc = Vd - \omega e \cdot Lq \cdot Iq$

補償後のｑ軸電圧 $Vqc = Vq + \omega e \cdot (Ld \cdot Id + K / P)$　（K：モータパラメータ（Ｎ・ｍ/Ａ））

＊ 参考文献　杉本英彦 編著 「ACサーボモータシステムの理論と設計の実際」　総合電子出版社 刊

＊　Simtrol-m等の価格／具体的な製品仕様については、別紙カタログをご参照頂くか、当社までお問い合わせ下さい。

＊ 本資料の内容は 2006 年4月現在のものであり、今後改良のため、予告なく変更する場合があります。ご了承下さい。

株式会社 **昭 和 電 業 社**

〒299-0111　千葉県市原市姉崎 745-2

TEL : 0436-61-4616

HP : http://www.k-sd.co.jp/

Mail : kentac@k-sd.co.jp