
モータ制御開発及び組込み支援システム

S i m t r o l - m

技 術 資 料

2007-1 改版

株式会社 昭和電業社

目 次

1、製品概要	1
1. 1 主な機能／特徴	1
1. 2 S i m t r o l -mを用いた場合の開発手順（例）	1
1. 3 従来工程との比較（例）	1
2、機能概説	2
3、ライブラリ機能	2
4、操作概説	4
4. 1 エディタ（ブロック・ダイアグラムの作成）	4
4. 2 インタープリタ（シミュレーション）	7
4. 3 コンパイラ（C言語コード作成）	9
5、S i m t r o l -mによるブラシレスDCモータの駆動例	12
6、ブロック内演算内容	17
6. 1 BLDCM ブロック内演算内容	17
6. 2 BLDCM13600 ブロック演算内容	17
6. 3 DCCTRL ブロック演算内容	18

1、製品概要

KENTAC Simtrol-m（シムトロール・エム：以下Simtrol-mと記す）は、マイコンによるモータ制御システム／組込みシステム向け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、ブロック・ダイアグラムによる制御設計・シミュレーション・C言語ソース・コードの作成が実行できます。CPU向けのコンパイラ・システムとの連携により、モジュールの実装及びテストも可能です。

特にモータの制御を得意としております。

＊ “Simtrol-m” は株式会社 昭和電業社の登録商標です。

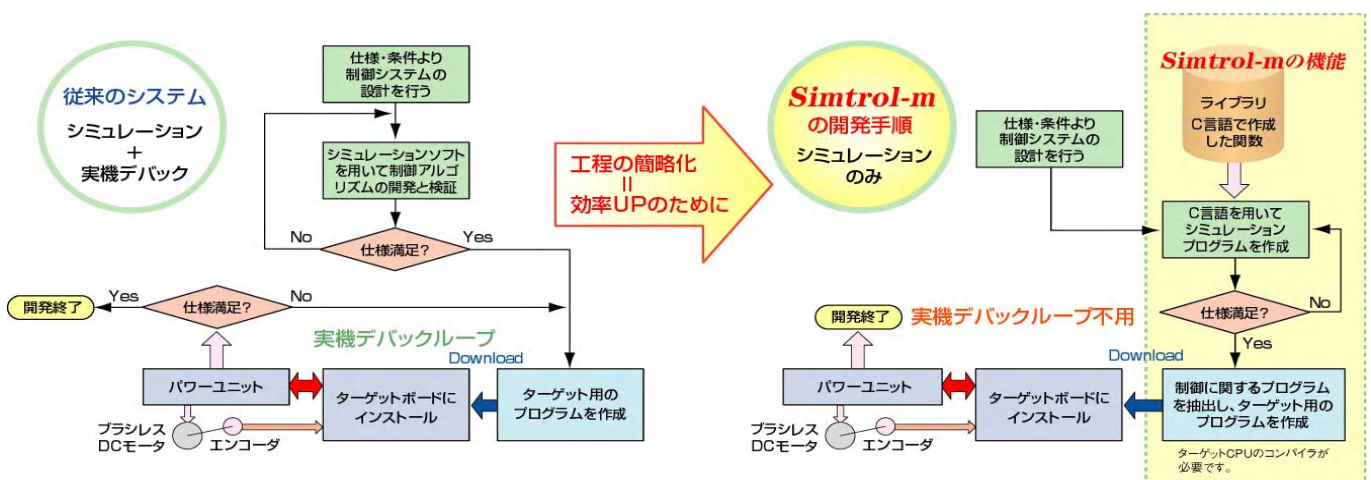
1. 1 主な機能／特徴

- ・モータ制御／組込みシステム制御に必要なブロックがライブラリ化され、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な箇所にモニタ装置（関数）を挿入し、シミュレーションの結果をグラフィカルに表示できます。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する。
- 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のコンパイル機能により、ブロック・ダイアグラムに対応した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・ボードでモータを制御／駆動してみる。
- ⑧ 駆動結果をオシロ・スコープなどでグラフ化し、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・ダイアグラムを作成し、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、

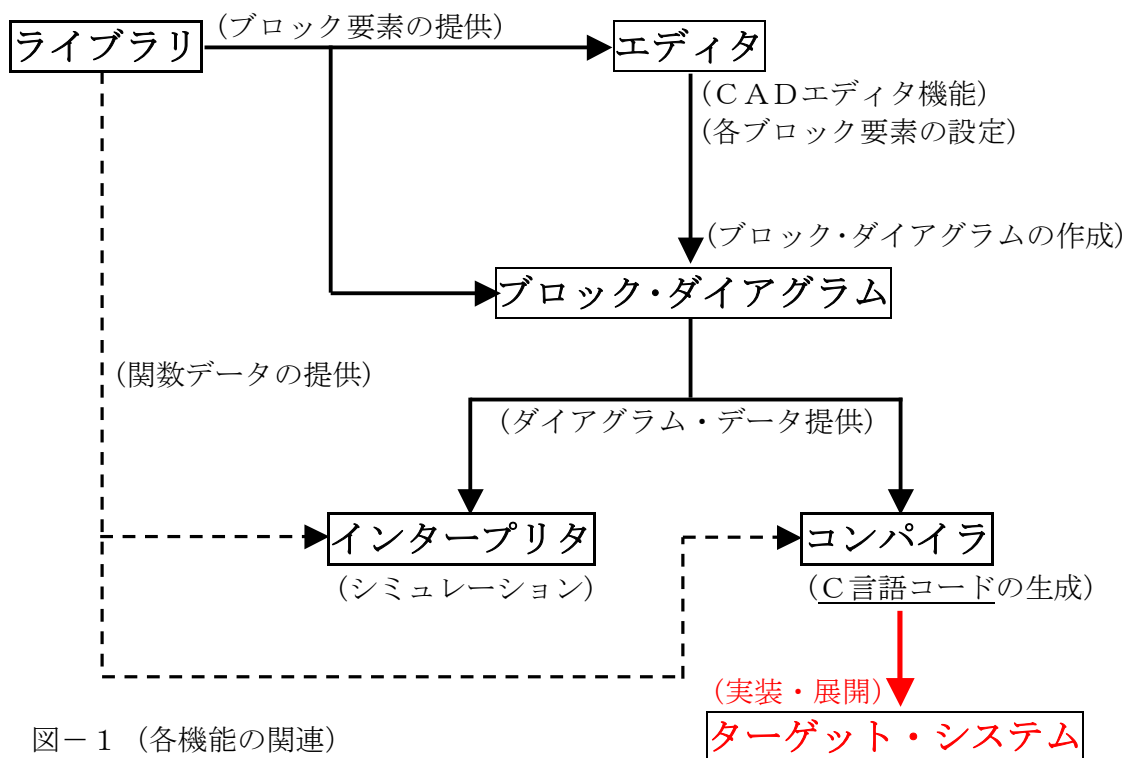
- ①ブロック・ダイアグラムを作成・編集する画面機能を、「エディタ」、
 - ②シミュレーションを設定・実行・表示する機能を、「インタープリタ」、
 - ③C言語ソース・コードを生成・表示する機能を、「コンパイラ」、
- と呼びます。以下、この呼称を使用します。

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



3、ライブラリ機能

2007年1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（モータ制御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。
 $\alpha\beta$ 座標系→dq座標系変換、dq軸間非干渉化関数、dq座標系→ $\alpha\beta$ 座標系変換。
 $\alpha\beta$ 座標系→3相3線変換、3相3線→ $\alpha\beta$ 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）

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)。

⑦P I O関数（ハードウェアに依存する機能。当社製C P U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。

⑧その他

倒立振子。切替スイッチ。タイマなど。

上記の各関数が各々のブロックと対応付けられて、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することができます。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例)

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、エディタ画面上の任意の箇所に貼り付けます。

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回転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Simulink-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。

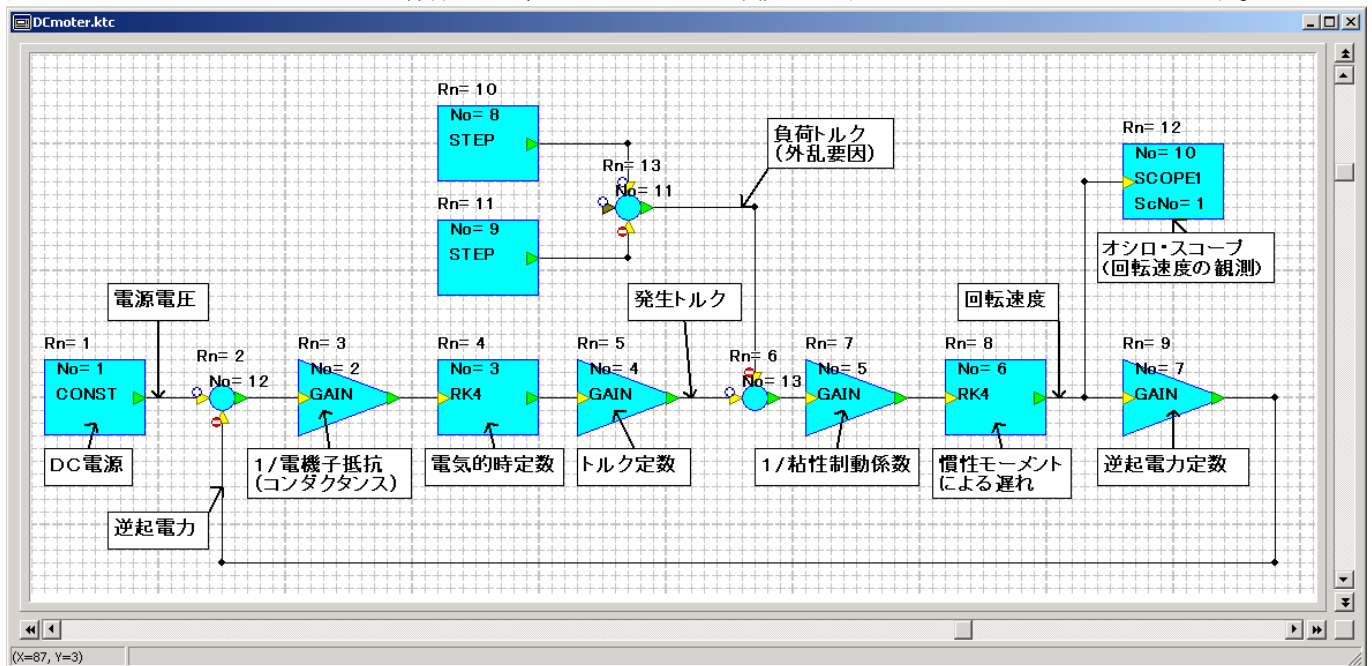


図-3 (DCモータの等価ブロック・ダイアグラム：コメント付)

図-3のブロック・ダイアグラムでは、DC電源からの入力電圧を定数(CONST:No. 1ブロック)とし、モータの電機子抵抗の逆数(GAIN:No. 2ブロック)／電氣的時定数(RK4:No. 3ブロック)／トルク定数(GAIN:No. 4ブロック)から、モータが発生するトルク(No. 4ブロックの出力)を求めます。

一方、外乱要因であるSTEP(No. 8ブロック)／STEP(No. 9ブロック)はモータの負荷トルクを表わします。ここでは負荷がパルス状に与えられた状態(No. 11ブロックの出力)とします。

モータが発生したトルク(No. 4ブロックの出力)から負荷トルク分(No. 11ブロックの出力)を減算(No. 13ブロック)すれば、モータ自体の回転に寄与するトルク(No. 13ブロックの出力)を求めることができます。

次に、軸受け等の機械的な粘性制動係数(GAIN:No. 5ブロック)及び、ロータの慣性モーメントの影響(RK4:No. 6ブロック)を考慮し、モータの回転速度(No. 6ブロックの出力)を求め、オシロスコープ(SCOPE1:No. 10ブロック)で表示します。

最後に、DCモータの回転速度に比例して電機子に生じる、逆起電力分の電圧(GAIN:No. 7ブロックの出力)が、電源電圧とは逆向きに作用(No. 12ブロック)しています。

(但し、GAIN：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 No.	種類	パラメータ	設定値	計算式	単位
1	CONST	constant	12	V_a	[V]
2	GAIN	gain	1	$1/R_a$	$[1/\Omega] ([S])$
3	RK4	τ	0.01	L_a/R_a	[sec]
4	GAIN	gain	0.05	K_T	$[N \cdot m/A]$
5	GAIN	gain	100000	$1/D$	$[rad/N \cdot m \cdot sec]$
6	RK4	τ	5	J/D	[sec]
7	GAIN	gain	0.05	K_E	$[V \cdot sec/rad]$
8	STEP	delay	0.08		[sec]
		vinit	0		$[N \cdot m]$
		vfinal	0.15	TL	$[N \cdot m]$
9	STEP	delay	0.14		[sec]
		vinit	0		$[N \cdot m]$
		vfinal	0.15	TL	$[N \cdot m]$
10	SCOPE1	tmin	0		[sec]
		tmax	0.2		[sec]
		tgrid	0.05		[sec]
		ymin	0		$[rad/sec]$
		ymax	250		$[rad/sec]$
		ygrid	50		$[rad/sec]$
		ScopeNo	1	(自動設定／変更不可)	
		buffsize	2001	(自動設定／変更不可)	

表－１ (図－３ 内各ブロックのパラメータ値一覧)

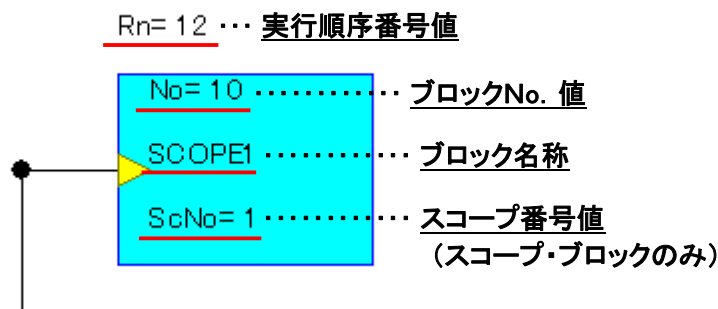
$V_a = 12.0$: モータ供給電圧[V]
 $R_a = 1.0$: モータ電機子抵抗[Ω]
 $L_a = 0.01$: 電機子インダクタンス[H]
 $\tau = 0.01$: 電氣的時定数[sec]
 $K_T = 0.05$: トルク定数[$N \cdot m/A$]
 $K_E = 0.05$: 逆起電力定数[$V \cdot sec/rad$]
 $TL = 0.15$: 負荷トルク[$N \cdot m$]
 $D = 0.00001$: 粘性制動係数[$N \cdot m \cdot sec/rad$]
 $J = 0.00005$: 慣性モーメント[$kg \cdot m^2$]
(表－１ 付記)

* SCOPE1 ブロックの ScopeNo／buffsize の項はユーザによる変更は不可です。
(システムによる自動設定です。)

* No. 8 (STEP) ブロックの出力から、No. 9 (STEP) ブロックの出力を
減算 (No. 11 ブロック) して、等価的にパルス状の信号を作成します。

4. 1. 1 ブロックの表示について。

エディタ画面上の各ブロックには、次のデータが表示されています。（図－4 参照）



図－4：ブロックの表示データ

① 実行順序番号値（R n = . . . ）

ブロックに対応する関数の、インタプリタ／コンパイラでの実行処理の順序を指定する番号です。この番号の昇順に関数が実行されます。コンパイラでソース・コードに関数が出力される順序も、この番号の順序に従います。

ユーザによる値の変更が可能です。

② ブロックNo. 値（N o = . . . ）

個々のブロックを指定する番号です。ブロックが生成される度に、新しい番号が昇順に割り振られ、ブロックが削除されるまで、変更されません。ブロックを識別するためのユニークな番号として用います。特にコンパイラ処理では、ブロックをプロパティ・データの配列と関連つけるために使用します。

ユーザによる値の変更はできません。

③ ブロック名称

ブロックの種類を表わします。表示される名称は任意に変更可能ですが、ブロックそのものの種類を変更することはできません。

④ スコープ番号値（S c N o = . . . ）

特にスコープ類のブロックに対してのみ、生成順に割り振られる番号です。インタプリタ処理において、表示用のスコープ類ブロックを特定するために用います。

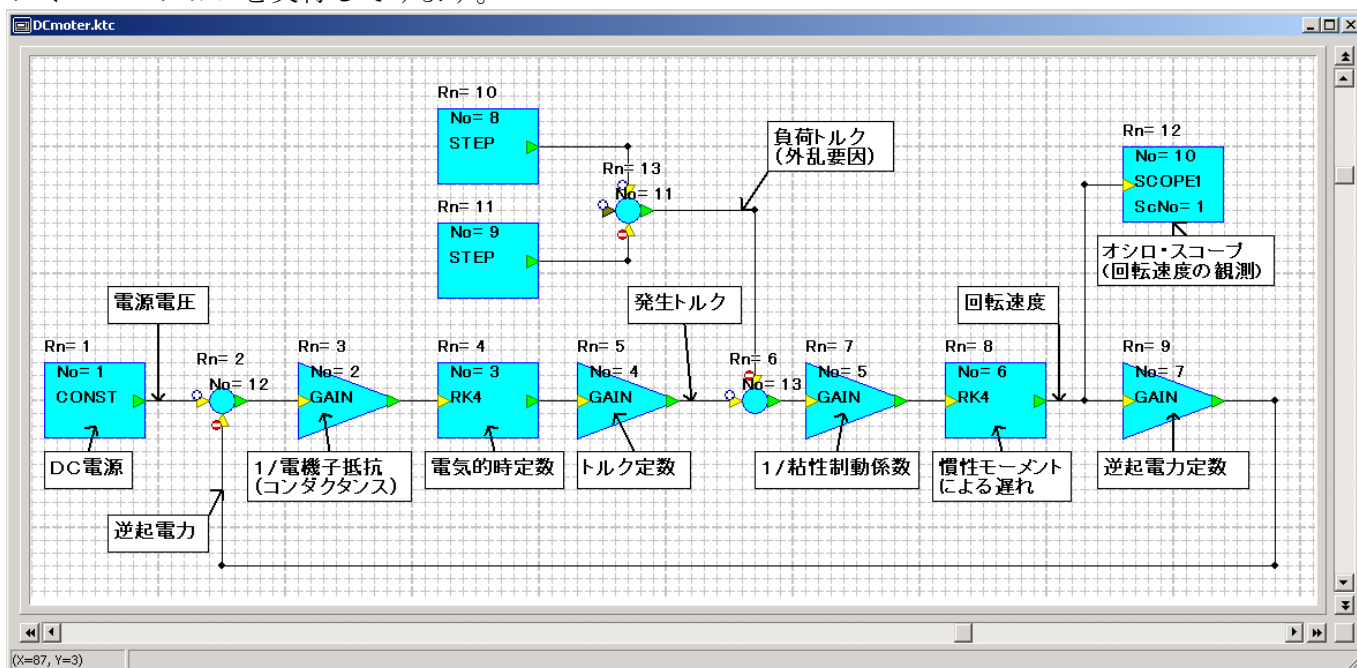
ユーザによる値の変更はできません。

上記の各情報は、画面表示／非表示の設定が可能です。

4. 2 インタープリタ (シミュレーション)

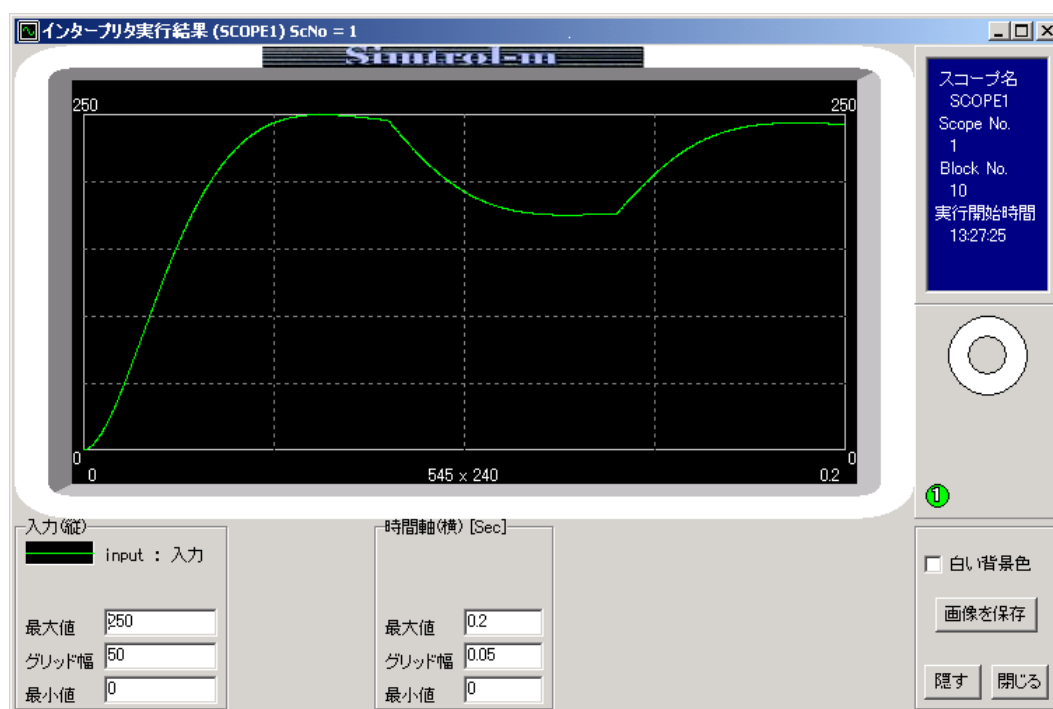
ブロック・ダイアグラムが作成できたら、インタープリタでシミュレーションを実行してみます。はじめに、メニューバーの[実行]―[インタープリタ設定]コマンドで、インタープリタの実行に必要な設定（実行時間などの設定）を実施します。

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によるシミュレーションを実行してみます。



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－5）に示します。



図－5 (インタープリタ実行結果：例)

前頁の図－５は、ブロック・ダイアグラムで等価的に表わしたＤＣモータの電機子に、０秒から直流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。

なお、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－６）

図－６（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[sec]
シミュレーションの実行時間の設定。
- ② 刻み時間 dt[sec]
シミュレーション実行時の処理刻み時間を設定。
- ③ 表示周期[ms]
シミュレーション画面(図－５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((株)ルネサス・テクノロジー製「HEW」など)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、基本的に同じ動作結果が得られます。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) (1/3)

```
////////////////////////////////////
// Simtrol-m by SHOWA DENGYOSHA CO.,LTD. //
////////////////////////////////////
//
// 作成元ファイル名 : [DCmoter.ktc]
//
// 作成日時 : [2006/09/21 14:06]
//

////////////////////////////////////
// 変数の宣言 //
////////////////////////////////////

// 線変数の宣言
float x0;
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 counter10;
int dataSize10;
float data10[2000];

// プロパティ・データ領域の宣言
float pro1[1], pro12[3], pro2[1], pro3[1], pro4[1];
float pro13[3], pro5[1], pro6[1], pro7[1], pro8[3];
float pro9[3], pro10[8], pro11[3];

// 処理用変数の宣言
int err; // エラー処理用変数
```

```

////////////////////////////////////
// 変数の初期化 //
////////////////////////////////////

x0 = 0;
x1 = 0;  x2 = 0;  x3 = 0;  x4 = 0;  x5 = 0;
x6 = 0;  x7 = 0;  x8 = 0;  x9 = 0;  x10 = 0;
x11 = 0; x12 = 0;
counter10 = 0;
dataSize10 = 2001;

////////////////////////////////////
// プロパティ値の設定 //
////////////////////////////////////

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

//2/3 入力加減演算
pro12[0] = -1;          // input1 : 入力 1(上) (加算:0 減算:1 表示なし:-1)
pro12[1] = 0;           // input1 : 入力 2(左) (加算:0 減算:1 表示なし:-1)
pro12[2] = 1;           // input1 : 入力 3(下) (加算:0 減算:1 表示なし:-1)

//GAIN : 増幅器(定数乗算器)
pro2[0] = 1;            // gain : 増幅率

//RK4 : 1次遅れ要素(ルンゲ・クッタ4次法)
pro3[0] = 0.01;         // tau : 時定数[sec]

//GAIN : 増幅器(定数乗算器)
pro4[0] = 0.05;         // gain : 増幅率

//2/3 入力加減演算
pro13[0] = 1;           // input1 : 入力 1(上) (加算:0 減算:1 表示なし:-1)
pro13[1] = 0;           // input1 : 入力 2(左) (加算:0 減算:1 表示なし:-1)
pro13[2] = -1;          // input1 : 入力 3(下) (加算:0 減算:1 表示なし:-1)

//GAIN : 増幅器(定数乗算器)
pro5[0] = 100000;       // gain : 増幅率

//RK4 : 1次遅れ要素(ルンゲ・クッタ4次法)
pro6[0] = 5;            // tau : 時定数[sec]

//GAIN : 増幅器(定数乗算器)
pro7[0] = 0.05;         // gain : 増幅率

```

```
//STEP : ステップ信号発生器
pro8[0] = 0.08;      // delay : 切り替え時間[sec]
pro8[1] = 0;        // vinit : 初期値
pro8[2] = 0.15;     // vfinal : 切り替え後の値

//STEP : ステップ信号発生器
pro9[0] = 0.14;     // delay : 切り替え時間[sec]
pro9[1] = 0;        // vinit : 初期値
pro9[2] = 0.15;     // vfinal : 切り替え後の値

//SCOPE1 : オシロスコープ(1入力)
pro10[0] = 0;       // tmin : 時間軸最小値[sec]
pro10[1] = 0.2;     // tmax : 時間軸最大値[sec]
pro10[2] = 0.05;    // tgrid : 時間軸目盛り刻み[sec]
pro10[3] = 0;       // ymin : y 軸最小値
pro10[4] = 250;     // ymax : y 軸最大値
pro10[5] = 50;      // ygrid : y 軸目盛り刻み
pro10[6] = 1;       // ScopeNo : スコープ番号
pro10[7] = 2000;    // bufsize : バッファサイズ

//2/3 入力加減演算
pro11[0] = 0;       // input1 : 入力1(上)(+(加算):0、-(減算):1)
pro11[1] = 0;       // input2 : 入力2(左)(+(加算):0、-(減算):1)
pro11[2] = 1;       // input3 : 入力3(下)(+(加算):0、-(減算):1)

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_sum3( x0, x1, x9, pro12, &x2, &err ); // 2/3 入力加減演算
fn_amp( x2, pro2, &x3, &err );          // GAIN : 増幅器(定数乗算器)
fn_rk4( x3, pro3, &x4, &err );          // RK4 : 1次遅れ要素(ルンゲ・クッタ4次法)
fn_amp( x4, pro4, &x5, &err );          // GAIN : 増幅器(定数乗算器)
fn_sum3( x12, x5, x0, pro13, &x6, &err ); // 2/3 入力加減演算
fn_amp( x6, pro5, &x7, &err );          // GAIN : 増幅器(定数乗算器)
fn_rk4( x7, pro6, &x8, &err );          // RK4 : 1次遅れ要素(ルンゲ・クッタ4次法)
fn_amp( x8, pro7, &x9, &err );          // GAIN : 増幅器(定数乗算器)
fn_step( pro8, &x10, &err );           // STEP : ステップ信号発生器
fn_step( pro9, &x11, &err );           // STEP : ステップ信号発生器
fn_scope1( x8, pro10, &counter10, dataSize10, data10, &err ); // SCOPE1 : オシロスコープ
(1入力)
fn_sum3( x10, x0, x11, pro11, &x12, &err ); // 2/3 入力加減演算
```

5、Simtrol-mによるブラシレスDCモータの駆動例

下図(図-7)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: No. 2)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:DCCTRL:No. 4)を実行しています。

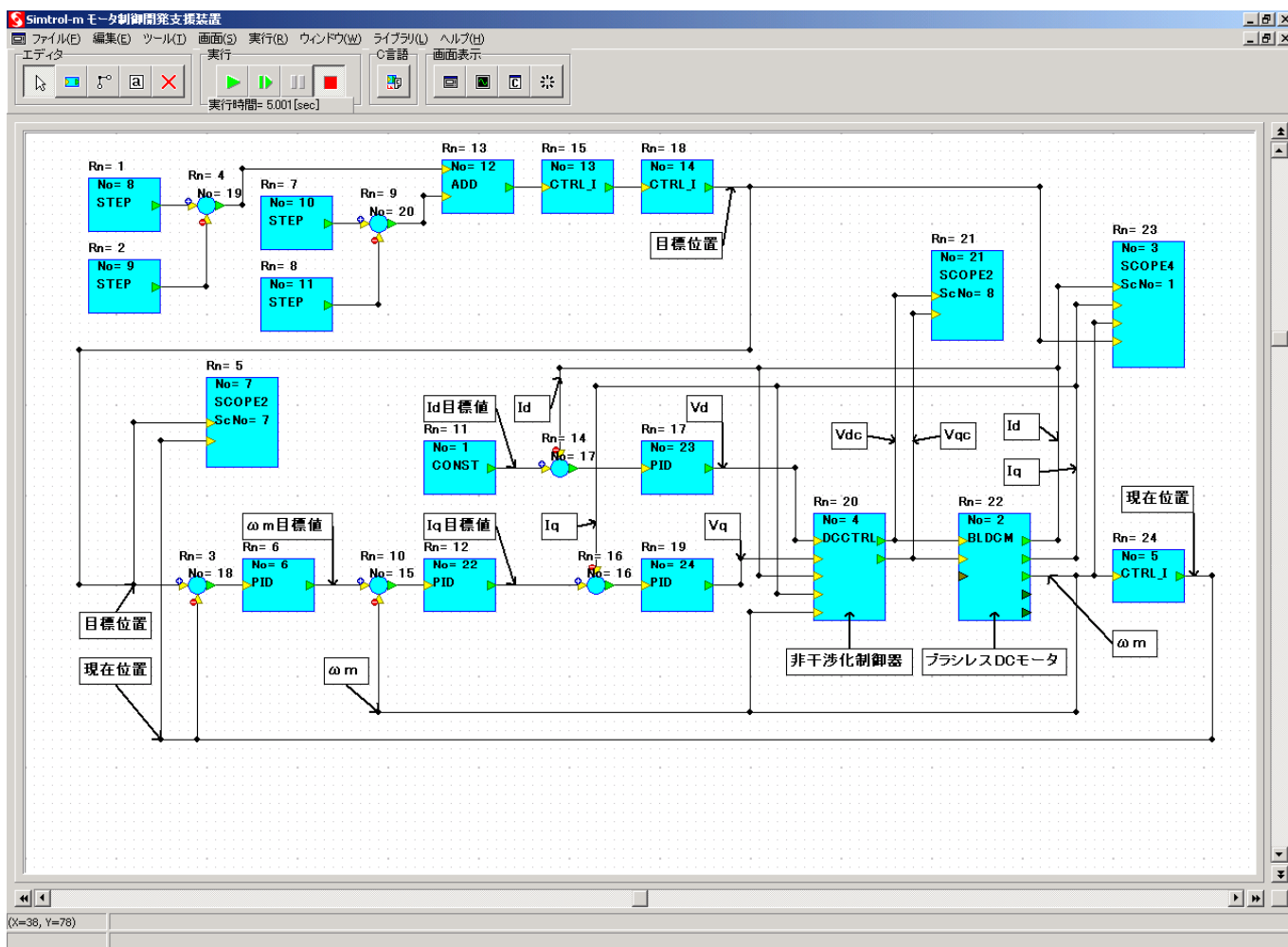


図-7 (ブラシレスDCモータの位置決め制御のブロック・ダイアグラムの例)

下図(図-8)は、インタプリタによる図-7のシミュレーション実行結果の一例です。



図-8 (図-7のインタプリタ実行結果 (SCOPE4:No.6ブロック))

図-8に示す SCOPE4 ブロック(No. 3)の入力1・入力 2 は、図-7のブラシレス DC モータ BLDCM ブロック(No. 2)のd軸電流値 I_d [A]／q軸電流値 I_q [A]、同じく入力3・入力 4 は BLDCM の回転角速度である ω_m [rad/sec]及び、位置決め制御の目標位置(角度)の値 [rad] (No. 14 ブロックの出力)です。それぞれスコープ表示されています。

図-7では、正負の単位パルス信号を直列に接続し、二度積分することで、位置決め目標値(目標位置)を設定します。(No. 8～No. 14 及び、No. 19、No. 20 の各ブロックによる。)

一方で、No. 2 の BLDCM ブロックから得られたモータの角速度 ω_m 値が I 制御要素(CTRL_I:No. 5)に入力されています。I 制御要素の出力(角速度の積分値)により、モータのロータ角度の実測値が得られ、これを位置決め制御の実測値として使用します。

以下、カスケード制御方式により、q軸電圧値 V_q [V]を求め、別に求めたd軸電圧値 V_d [V]とともに非干渉化制御用ブロック(DCCTRL:No. 4)への入力とします。具体的な方式は次の通りです。

まず、ロータの目標位置と現在位置の偏差(No. 18)を求めます。その偏差を0にするように操作するには必然的に角速度の値を操作することになるので、PID制御(No. 6)の操作量は角速度となります。それを角速度 ω_m の目標値とします。

次に BLDCM ブロック(No. 2)からの角速度の実測値 ω_m を基に偏差(No. 15)を求め、それを基に PID 制御(No. 22)を実行します。角速度を制御する場合、実際にはq軸電流 I_q を制御することになりますので、同様に PID 制御(No. 22)の操作量出力をq軸電流値 I_q の目標値とします。

また BLDCM ブロック(No. 5)からのq軸電流 I_q の実測値を基に、偏差(No. 16)を求め、PID 制御(No. 24)します。q軸電流値を制御する場合、実際にはq軸電圧 V_q を制御することになるので、PID 制御(No. 24)の操作量(出力)はq軸電圧値 V_q となります。

以上の手順で位置情報からq軸電圧値 V_q を求め、DCCTRL(No. 4)ブロックに入力します。

一方、d軸電流値 I_d の変移からd軸電圧値 V_d を求めます。

d軸電流値は基本的に0:一定(CONST:No. 1)と仮定できますので、その値をd軸電流の目標値として、BLDCM ブロック(No. 2)からd軸電流の実測値を求め、d軸電流の偏差(No. 17)を求めます。

d軸電流 I_d の偏差と実測値を PID 制御(No. 23)することで、操作量出力としてd軸電圧 V_d を求めることができます。その後、 V_d を DCCTRL(No. 4)ブロックに代入し、 V_q/V_d 相互間の非干渉化制御を実施します。

以上の関係を下図(図-9)に図示します。

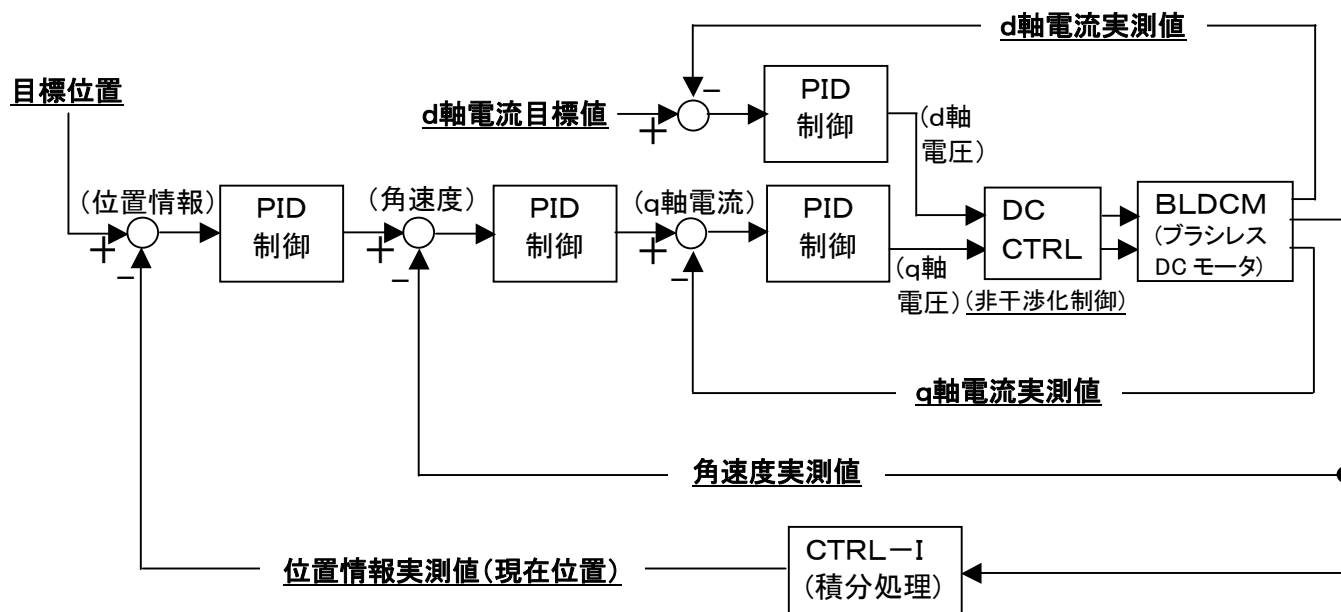


図-9 (図-7の制御内容概略図)

＊ カスケード制御方式

フィードバック制御において、一つの制御装置(一次ループの調整計)の出力信号により、他の制御系(二次ループの調整計)の目標値を変化させて行なう制御方式です。二次ループに入る外乱が抑制され、一次ループ側からも制御対象の特性が扱いやすくなり、制御が容易になります。

図-7の SCOPE2 (No. 11) ブロックでは、 ω_m 値から得られたモータの現在位置 (角度) が入力2に、前述の位置制御目標値が入力1にそれぞれ入力されており、スコープ表示できます。

図-7の SCOPE2 (No. 11) ブロックでのシミュレーション結果の一例を、下図(図-10)に示します。

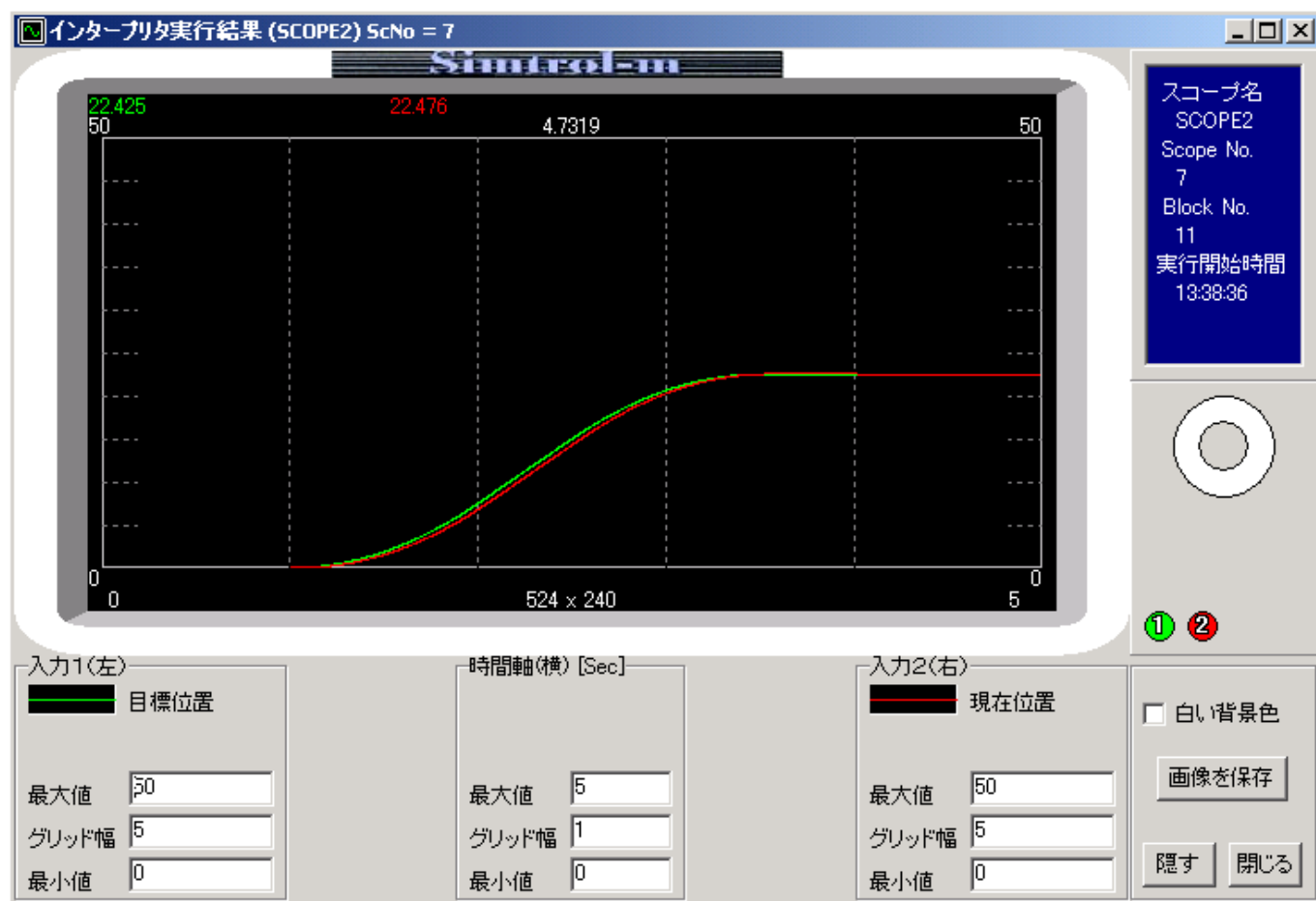


図-10 (図-7のインタープリタ実行結果 (SCOPE2:No.11ブロック))

図-10では、目標位置(目標値)は緑色の線で、現在位置(実測値)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(No.2)ブロック及び、d-q軸間非干渉化:DCCTRL(No.4)ブロックのパラメータ設定を下図(図-11)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。(プロパティの部分には、BLDCM/DCCTRLともに同じ値を設定して下さい。)

ブロック内容

ブロックNo.=2 BLDCM
ブロック図での表示
BLDCM 初期化

名称
ブラシレスDCモータ(モータパラメータ値使用)

説明
ブラシレスDCモータ
モータパラメータ値その他の値を与え、
ブラシレスDCモータをシミュレーションする。

プロパティ
Ra:巻線抵抗 [Ω] = 4.03
Ld:d軸インダクタンス [H] = 0.09
Lq:q軸インダクタンス [H] = 0.09
J:慣性モーメント [kgm²] = 0.0012
D:粘性摩擦係数 [Nm·s/rad] = 0.0001
K:モータパラメータ [Vs/rad]または[Nm/A] = 0.44 初期化

入力端子名
vd:d軸電圧 [V]
vq:q軸電圧 [V]
TL:負向トルク [Nm] 接続先から取得 初期化

出力端子名
id:d軸電流 [A]
iq:q軸電流 [A]
ωm:モータの回転角速度 [rad/sec]
θm:モータの回転角 [rad]
Tm:モータ発生トルク [Nm] 初期化

実行ディレイサイクル
1

OK キャンセル

図-11(a):BLDCM ブロックの設定内容

ブロック内容

ブロックNo.=4 DCCTRL
ブロック図での表示
DCCTRL 初期化

名称
ブラシレスDCモータの非干渉化(モータパラメータ値使用)

説明
ブラシレスDCモータの非干渉化
ブラシレスDCモータのd-q軸電流制御時に、
d-q軸間での速度起電力による干渉を除去
するため、供給されるd軸電圧、q軸電圧を、

プロパティ
Ra:巻線抵抗 [Ω](未使用) = 4.03
Ld:d軸インダクタンス [H] = 0.09
Lq:q軸インダクタンス [H] = 0.09
J:慣性モーメント [kgm²](未使用) = 0.0012
D:粘性摩擦係数 [Nm·s/rad](未使用) = 0.0001
K:モータパラメータ [Vs/rad]または[Nm/A] = 0.44 初期化

入力端子名
vd:d軸電圧 [V]
vq:q軸電圧 [V]
id:d軸電流 [A]
iq:q軸電流 [A]
ωm:モータ回転速度 [rad/s] 接続先から取得 初期化

出力端子名
vdc:非干渉化されたd軸電圧 [V]
vqc:非干渉化されたq軸電圧 [V] 初期化

実行ディレイサイクル
1

OK キャンセル

図-11(b):DCCTRL ブロックの設定内容

配列行	データ	値	単位
0	Ra (巻線抵抗)	4.03	[Ω]
1	Ld (d軸インダクタンス)	0.09	[H]
2	Lq (q軸インダクタンス)	0.09	[H]
3	J (慣性モーメント)	0.0012	[kg・m ²]
4	D (粘性摩擦係数)	0.0001	[N・m・sec/rad]
5	K (モータパラメータ)	0.44	[V・sec/rad] または [N・m/A]
6	P (極対数)	2	—

図-11 (a) 付表:BLDCM ブロックのプロパティ・データ例

配列行	データ	値	単位
0	Ra (巻線抵抗) (未使用)	4.03	[Ω]
1	Ld (d軸インダクタンス)	0.09	[H]
2	Lq (q軸インダクタンス)	0.09	[H]
3	J (慣性モーメント) (未使用)	0.0012	[kg・m ²]
4	D (粘性摩擦係数) (未使用)	0.0001	[N・m・sec/rad]
5	K (モータパラメータ)	0.44	[V・sec/rad] または [N・m/A]
6	P (極対数)	2	—

図-11 (b) 付表:DCCTRLブロックのプロパティ・データ例

6、ブロック内演算内容

Simulinkのブロック内部における演算処理の一例として、図-7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、原理／構造は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、等価的に直流量で扱うことが出来ます。即ち、通常のDCモータのような取り扱いが可能となります。（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、 V_d と V_q 、 I_d と I_q はそれぞれ直交しております。 I_d はトルクを発生しない電流成分、 I_q 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。

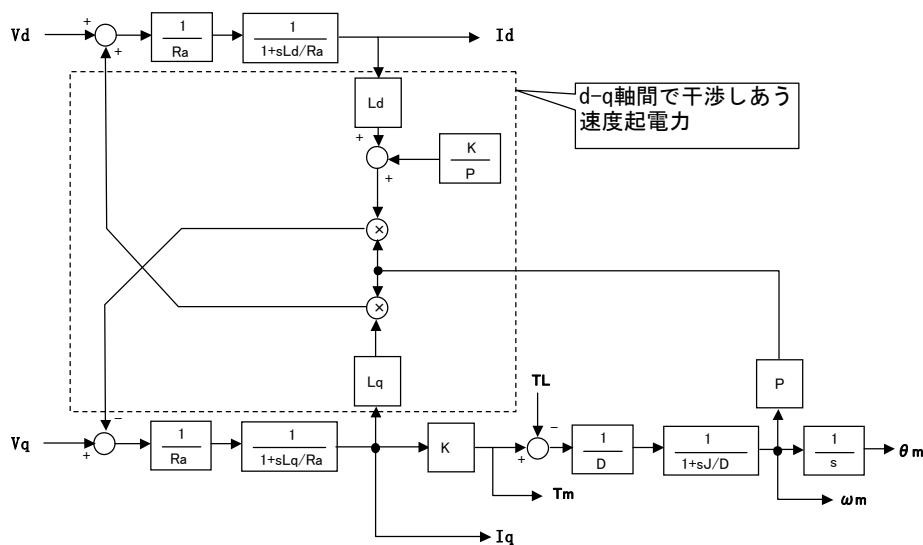


図-12：ブラシレスDCモータの直流量シミュレーション

6.2 BLDCM13600ブロックの演算内容

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。シミュレーション実行時はBLDCMブロック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。

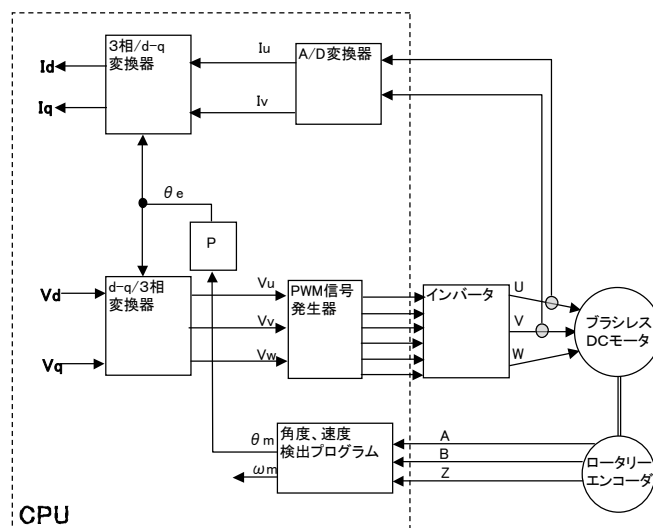


図-13：BLDCM13600ブロック演算内容（コントロール時）
（コントロール時：関数をマイコン・ボード内に組み込んだ場合）

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

$$\omega_e = \omega_m \cdot P \quad (\omega_m: \text{モータ回転数}, P: \text{極対数})$$

$$\text{補償後のd軸電圧 } \mathbf{V}_{dc} = V_d - \omega_e \cdot L_q \cdot I_q$$

$$\text{補償後のq軸電圧 } \mathbf{V}_{qc} = V_q + \omega_e \cdot (L_d \cdot I_d + K / P) \quad (K: \text{モータパラメータ (N} \cdot \text{m/A)})$$

* 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2007年1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