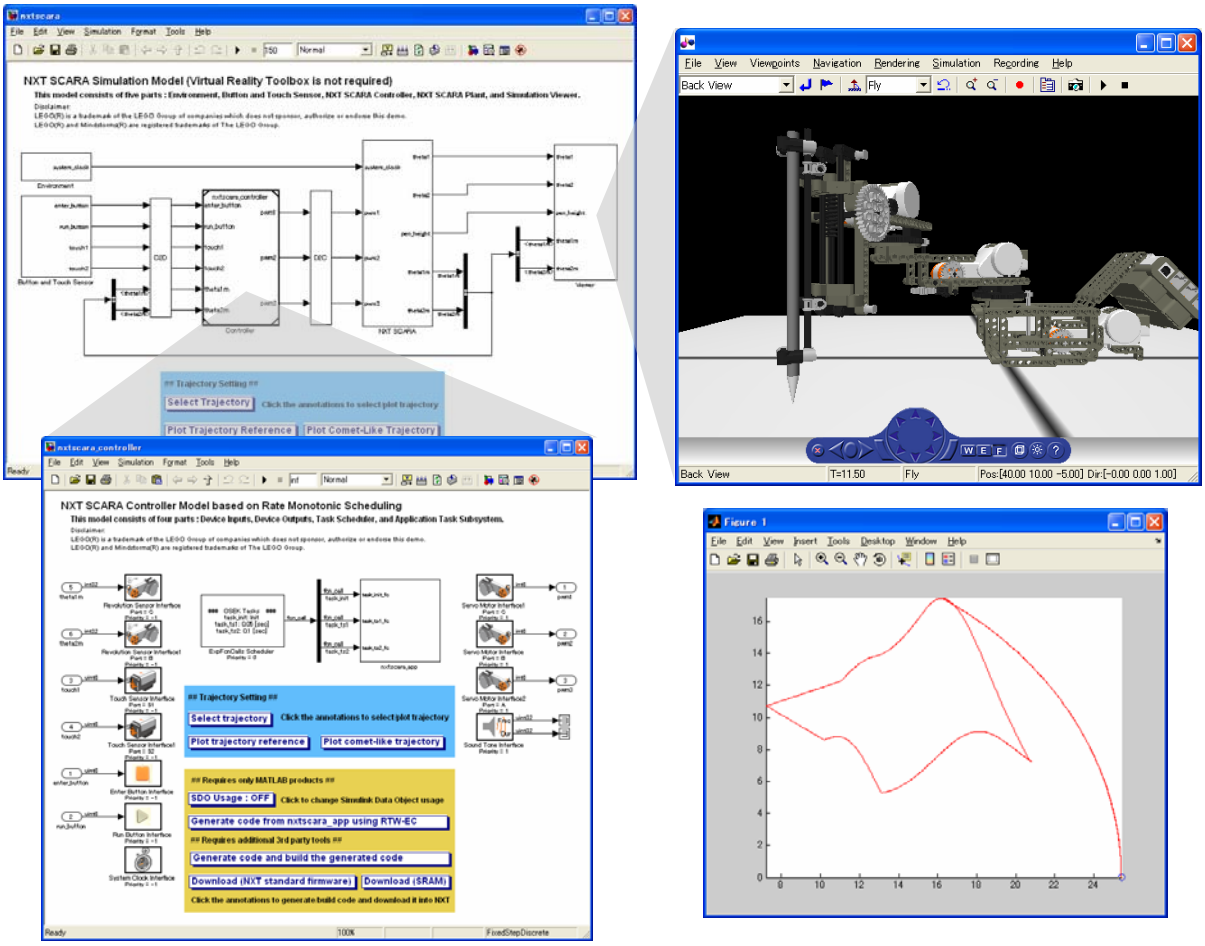
NXT SCARA のモデルベース開発

～LEGO Mindstorms NXT を用いた
2 リンク水平多関節ロボットの制御～



つくる情熱を、支える情熱。

CYBERNET

■ **著者（初版）**

サイバネットシステム株式会社
応用システム第 1 事業部　技術部　アドバンストサポート第 1 グループ
山本　順久　　E-mail : y_yama@cybernet.co.jp

■ **改訂履歴**

バージョン	年月	改訂内容	著者・編者
1.0	2008/09	初版	山本順久 y_yama@cybernet.co.jp

本資料の内容・記載 URL は予告無く変更される場合があります。

はじめに

NXT SCARA は LEGO Mindstorms NXT を用いて作成した 2 リンク水平多関節ロボットです。SCARA は Selective Compliant Assembly Robot Arm の頭文字をとった略称です。SCARA の詳細については下記 URL を参照してください。

http://en.wikipedia.org/wiki/SCARA

本資料は MATLAB / Simulink を用いた NXT SCARA の軌道追従制御プログラムのモデルベース開発について説明しています。主な内容は次の通りです。

- NXT SCARA のモデリング
- NXT SCARA の目標軌道作成
- NXT SCARA の制御器設計
- NXT SCARA モデルの解説
- シミュレーション結果および実験結果

前準備

NXT SCARAの組み立て方法については**NXT SCARA 組み立て手順書**をご覧ください。また、本資料ではモデルベース開発環境としてEmbedded Coder Robot NXTを使用しています。予めEmbedded Coder Robot NXT を下記URLからダウンロードし、**Embedded Coder Robot NXT 設定手順書**（Embedded Coder Robot NXT Instruction Jp.pdf）をご覧になって必要な設定と動作確認を行ってください。

http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=13399&objectType=file

本資料では下記バージョンのフリーソフトウェアを使用しています。

ソフトウェア	バージョン番号
Embedded Coder Robot NXT	3.14
nxtOSEK（旧名：LEJOS OSEK）	2.03
Cygwin	1.5.24
GNU ARM	4.0.2

使用 MATLAB 製品

製品名	バージョン番号	リリース番号
MATLAB®	7.5.0	R2007b
Simulink®	7.0	R2007b
Real-Time Workshop®	7.0	R2007b
Real-Time Workshop® Embedded Coder	5.0	R2007b
Virtual Reality Toolbox　(N1)	4.6	R2007b

(N1)： Virtual Reality Toolbox は 3D 表示有りモデル（nxtscara_vr.mdl）を使用する際に必要です。Virtual Reality Toolbox が無くても 3D 表示無しモデル（nxtscara.mdl）のシミュレーションおよびコード生成は可能です。

ファイルリスト

ファイル名	内容
cal_cp_ptp.m	CP 軌道および PTP 軌道計算用 M-関数
cal_eta.m	基本軌道計算用 M-関数
cal_ptp.m	PTP 軌道計算用 M-関数
cal_time_data.m	終了時間・ペン操作時間計算用 M-関数
cat_cp_ptp.m	CP 軌道・PTP 軌道結合用 M-関数
chk_limit.m	角度・角速度制限チェック用 M-関数
cp_circle.m	CP 軌道計算用 M-関数（円）
cp_ml_logo.m	CP 軌道計算用 M-関数（MATLAB ロゴ）
cp_smile.m	CP 軌道計算用 M-関数（スマイルマーク）
cp_spiral.m	CP 軌道計算用 M-関数（螺旋）
ml_logo.mat	MATLAB ロゴ用座標データ
nxtscara.mdl	NXT SCARA モデル（Virtual Reality Toolbox による 3D 表示無）
nxtscara_controller.mdl	NXT SCARA コントローラモデル
nxtscara_vr.mdl	NXT SCARA モデル（Virtual Reality Toolbox による 3D 表示有）
param_controller.m	制御器（コントローラ）パラメータ定義用 M-スクリプト
param_nxtscara.m	NXT SCARA パラメータ定義ファイル（param_***.m を実行）
param_plant.m	制御対象（プラント）パラメータ定義用 M-スクリプト
param_ref.m	目標座標および目標角度計算用 M-スクリプト
param_sim.m	シミュレーションパラメータ定義用 M-スクリプト
post_sdo_codegen.m	Simulink Data Object 適用コード生成 後処理用 M-スクリプト
pre_sdo_codegen.m	Simulink Data Object 適用コード生成 前処理用 M-スクリプト
theta2xy.m	角度・座標変換用 M-関数
vr_nxtscara.wrl	マップ参照＆NXT SCARA 用 VRML ファイル
vr_nxtscara_track.wrl	マップ用 VRML ファイル
xy2theta.m	座標・角度変換用 M-関数

目次

1　制御系モデルベース開発

制御系モデルベース開発全般について概説します。

1.1　モデルベース開発とは

モデルベース開発（Model Based Design / Development、MBD と略されます）とは、シミュレーション可能なモデルを用いるソフトウェア開発手法です。制御系 MBD では、制御器および制御対象、またはその一部をモデルで表現し、机上シミュレーション／リアルタイムシミュレーションにより制御アルゴリズムの開発・検証を行います。リアルタイムシミュレーションとは、制御系の一部を実機、その他をリアルタイムシミュレータ上で動作するモデル生成コードとし、実時間での動作検証を行うシミュレーション技術のことです。

さらに、RTW-EC 等の C コード生成ツールを用いて、制御器モデルから実際の制御器（マイコン等）に組み込む制御用 C プログラムを作成することができます。図 1-1 は MATLAB 製品を用いた制御系 MBD の概念図です。

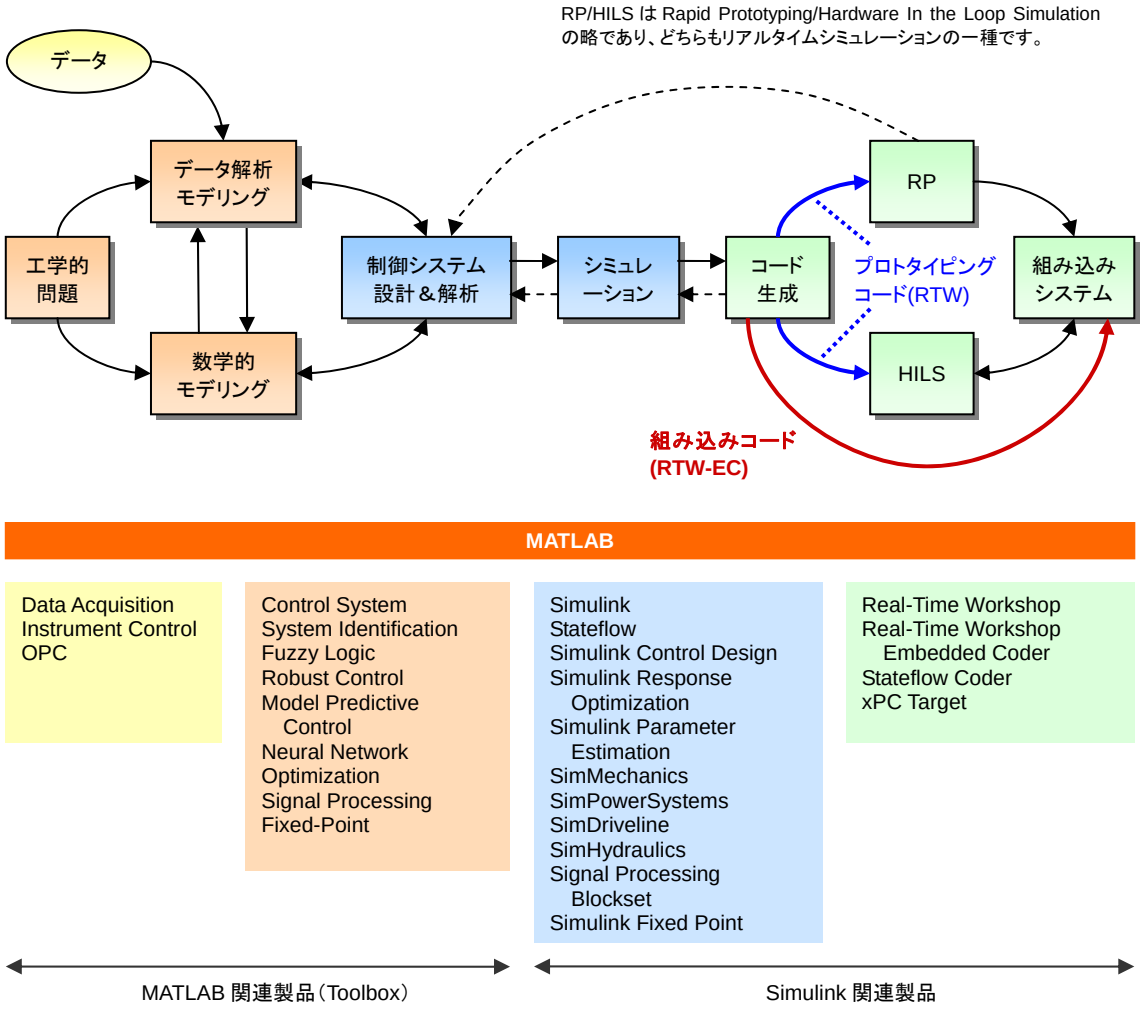


図 1-1　MATLAB 製品を用いた制御系 MBD

1.2　モデルベース開発プロセス

MBD による制御ソフトウェアの開発プロセスは、図 1-2 に示す V プロセスを用いて説明されます。V プロセスはソフトウェア開発を要求分析、各種設計、コーディングの工程に分け、各工程に検査（テスト）を対応させた V 字型の開発プロセスです。MBD では、V プロセスの左側の工程でモデリングを行い、制御仕様完成度の早期向上を図ります。また、作成したモデルを検証工程で利用することにより、コード品質・検証効率向上を図ります。

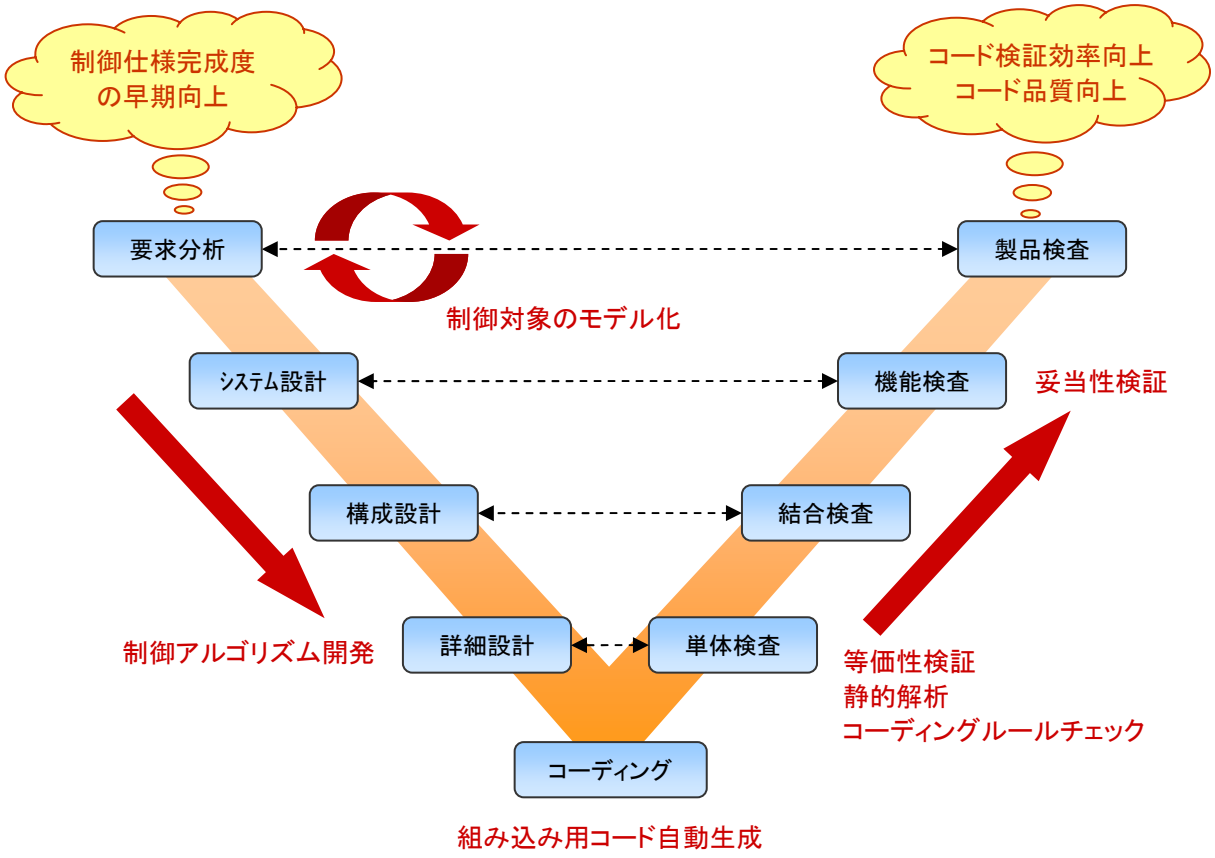


図 1-2　制御系 MBD の V プロセス

1.3 モデルベース開発のメリット

MBD には、以下のようなメリットがあります。

- 机上シミュレーションによる仕様ミスの早期検証
- リアルタイムシミュレーションによる試作工数削減・フェイルセーフ検証
- モデル検証機能によるテストの効率化
- モデル仕様の共通認識によるコミュニケーション改善
- 自動コード生成による人的コーディング工数・エラー削減

2　NXT SCARA の機構

NXT SCARA の構造およびセンサ・アクチュエータの特徴について説明します。

2.1　構造

NXT SCARA の構造を図 2-1 に示します。

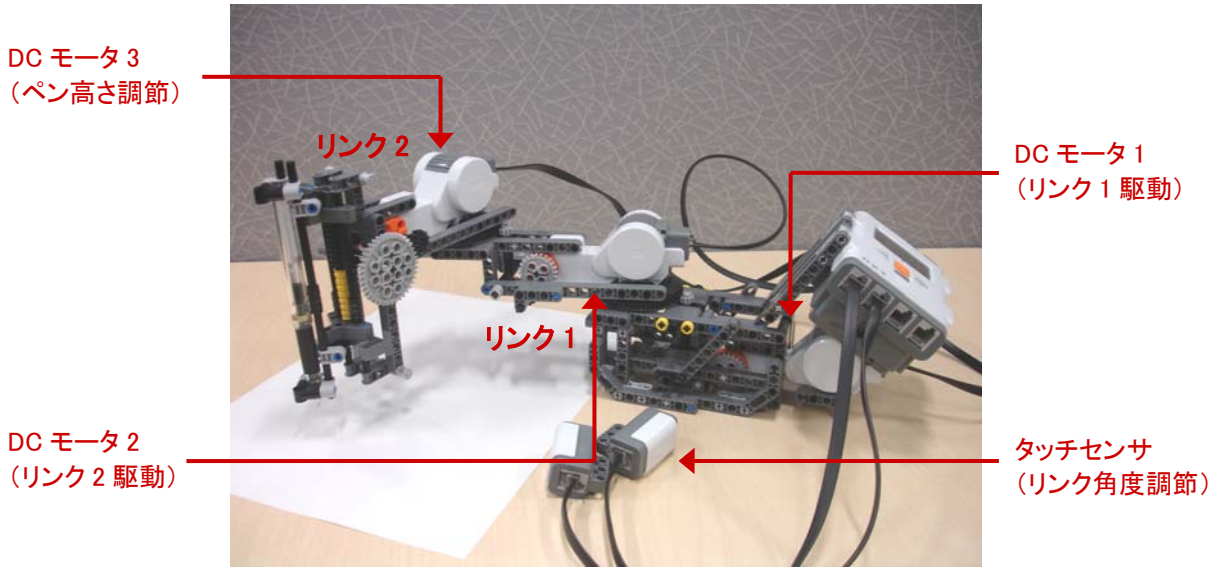


図 2-1　NXT SCARA

2.2　センサ・アクチュエータ

NXT SCARA で使用されているセンサ・アクチュエータの特性を表 2-1 および表 2-2 に示します。

表 2-1　センサの特性

センサ	出力値	単位	データタイプ	最大サンプル数 [1/sec]
ロータリエンコーダ	モータ回転角度	deg	int32	1000
タッチセンサ	タッチ有無		int8	1000

表 2-2　アクチュエータの特性

アクチュエータ	入力値	単位	データタイプ	最大サンプル数 [1/sec]
DC モータ	PWM	%	int8	500

参考文献[1]に DC モータの各種特性が紹介されています。一般に、センサ・アクチュエータには個体差があります。

2.3 ギヤトレイン・バックラッシュ

NXT SCARA ではモータ回転をリンク回転に変換・減速するために、ギヤトレイン（歯車列）とターンテーブルを関節部に用いています。図 2-2 はリンク 1-2 間のギヤトレインの拡大図です。

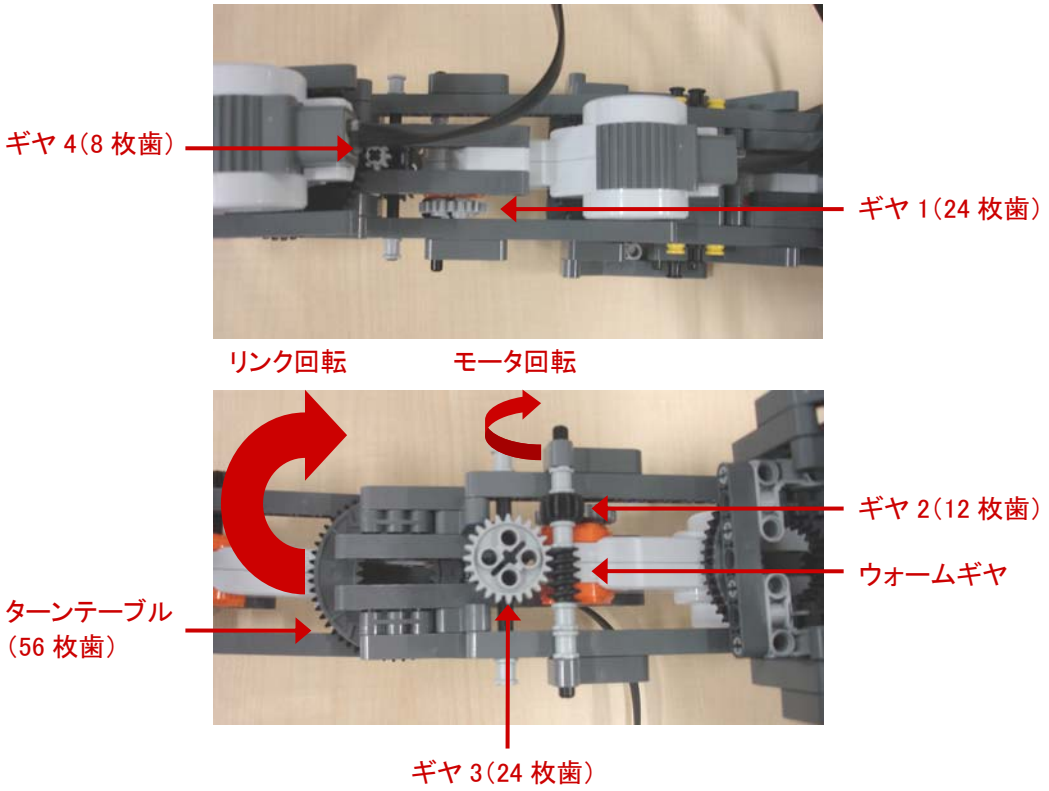


図 2-2　ギヤトレイン＆ターンテーブル

リンク回転角度 θ とモータ回転角度 θ_m 間のギヤ比 g は次式で求まります（$\theta = \theta_m / g$）。

$$worm\ gear\ ratio = \frac{gear3\,teeth}{1} = 24 \tag{2.1}$$

$$g = \frac{turn\,table\,teeth}{gear4\,teeth} \times worm\ gear\ ratio \times \frac{gear2\,teeth}{gear1\,teeth} = \frac{56}{8} \times 24 \times \frac{12}{24} = 84 \tag{2.2}$$

ギヤ間にはバックラッシュ（噛み合うギヤの間の隙間／遊び）があります。バックラッシュがあるとモータ逆回転時に空回り状態が発生するため、位置決め精度に悪影響を及ぼします。ギヤを噛み合った状態を保つためには、このバックラッシュを補償する必要があります。なお、ギヤを噛み合った状態をエンゲージ状態、噛み合っていない（空回りしている）状態をディスエンゲージ状態と呼びます。図 2-3 はバックラッシュの概念図です。

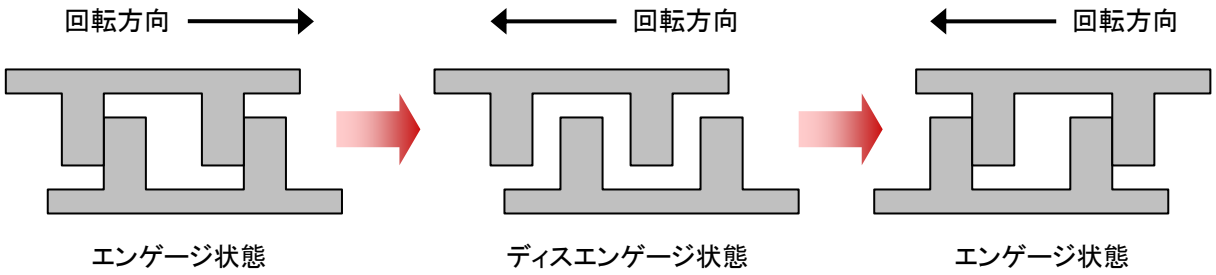


図 2-3　バックラッシュ（上側が駆動ギヤ）

ギヤトレインでは各ギヤのバックラッシュが加わるため、全体としてのバックラッシュが大きくなります。

2.4　リンク角度・角速度の制約

構造上、NXT SCARA のリンク角度・角速度には以下の制約があります。

最大リンク角度

リンク 1 の最大角度は約 90 [deg]、リンク 2 の最大角度は約 140 [deg]です。このため、アーム先端部の到達可能範囲に制約が生じます。

最大・最小リンク角速度

図 2-4 は各リンクを駆動するモータの PWM [%]と平均角速度 [deg/ms]を測定した結果です。リンク負荷の影響で、低 PWM ではモータが回転しない点にご注意下さい。

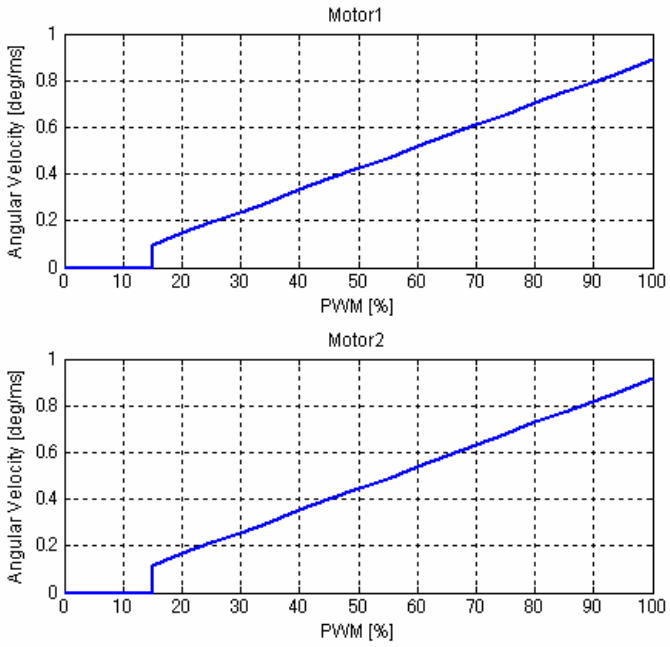


図 2-4　モータ PWM とモータ平均角速度の関係

図 2-4 の線形変化している部分を 1 次式としてフィッティングすると、次式を得ることができます。

$$pwm_i = gain_i \times \omega_i + offset_i \qquad (i = 1,2) \tag{2.3}$$

$$\begin{cases} gain_1 = 107.5056 \\ offset_1 = 4.403 \end{cases}, \quad \begin{cases} gain_2 = 106.748 \\ offset_2 = 2.3918 \end{cases}$$

ここで、pwm_i および ω_i は各モータの PWM [%]および平均角速度 [deg/ms]です。(2.3)式の pwm を 100 とすると、最大モータ角速度を求めることができます。同様に、pwm を 15 とすると最小モータ角速度を求めることができます。最大・最小モータ角速度を(2.2)式のギヤ比 g で割れば、最大・最小リンク角速度を求めることができます。

3　NXT SCARA のモデリング

NXT SCARA を 2 リンク水平多関節ロボットとしてモデリングし、アーム先端部の位置座標からリンク角度を求める逆運動学について説明します。本章の詳細については参考文献[2]および[3]をご覧下さい。

3.1　2 リンク水平多関節ロボット

NXT SCARA を図 3-1 に示す 2 リンク水平多関節ロボットとしてモデリングします。

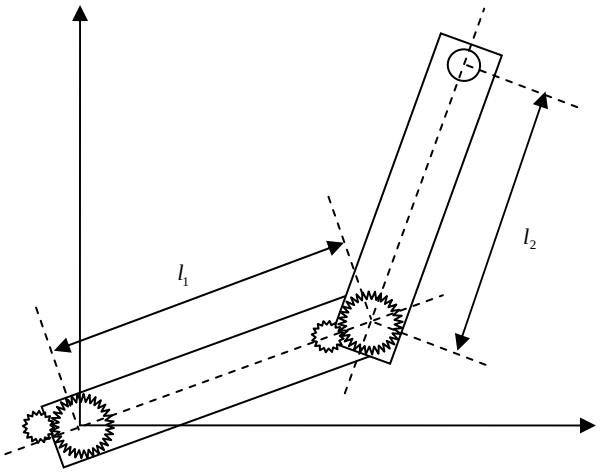
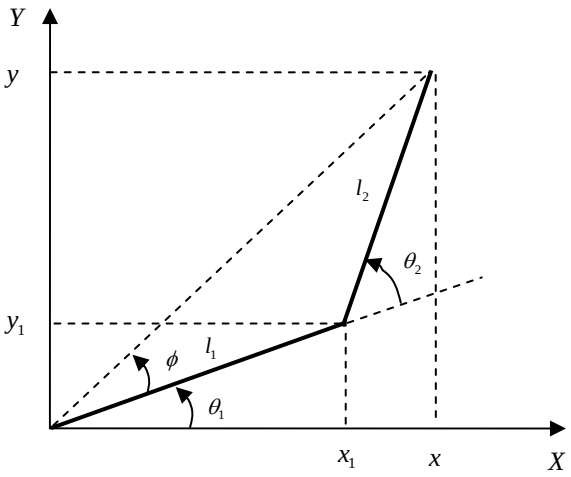


図 3-1　2 リンク水平多関節ロボット

図 3-2 に水平多関節ロボットの座標系を示します。



$\theta_{1,2}$ ： リンクの回転角度　　$\theta_{m_{1,2}}$: DC モータの回転角度　　$\theta_{1,2} = \theta_{m_{1,2}} / g_{1,2}$

図 3-2　水平多関節ロボットの座標系

NXT SCARA の物理パラメータは次の通りです。

$l_1 = 0.118$	$[m]$	:	リンク 1 長さ
$l_2 = 0.136$	$[m]$	:	リンク 2 長さ
$g_1 = 84$		:	リンク 1 駆動部のギヤ比
$g_2 = 84$		:	リンク 2 駆動部のギヤ比

3.2 逆運動学

逆運動学は、アーム先端部の位置や動作から関節の角度や動作を求める手順です。一般に、ロボットの動作制御においては先端部の動作を指令として与えることが多く、コントローラはその先端部への指令動作を達成させるための関節動作を求める必要があります。このため、既知の先端動作から未知の関節動作を求める逆運動学が重要といえます。

以下、2 リンク水平多関節ロボットのアーム先端位置(x, y)から関節角(θ_1, θ_2)を求める式を導出します。図 3-2 より、(x, y)と(θ_1, θ_2)の関係は次式で与えられます。

$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \tag{3.1}$$

$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \tag{3.2}$$

(3.1)式と(3.2)式からθ_2が求まります。

$$x^2 + y^2 = l_1^2 + l_2^2 + 2l_1 l_2 \cos\theta_2 \tag{3.3}$$

$$\theta_2 = \pm\cos^{-1}\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}\right) \tag{3.4}$$

また、幾何学的な関係から次式が成立します。

$$\theta_1 + \phi = \tan^{-1}\left(\frac{y}{x}\right) \tag{3.5}$$

$$\sin\phi = \frac{l_2 \sin\theta_2}{\sqrt{x^2 + y^2}} \tag{3.6}$$

(3.5)式と(3.6)式からϕおよびθ_1が求まります。

$$\phi = \pm\cos^{-1}\left(\frac{x^2 + y^2 + l_1^2 - l_2^2}{2l_1\sqrt{x^2 + y^2}}\right) \tag{3.7}$$

$$\theta_1 = \mp\cos^{-1}\left(\frac{x^2 + y^2 + l_1^2 - l_2^2}{2l_1\sqrt{x^2 + y^2}}\right) + \tan^{-1}\left(\frac{y}{x}\right) \tag{3.8}$$

(θ_1, θ_2)として 2 通りの解が得られますが、これは図 3-3 のように先端位置に対して 2 通りの姿勢が存在することに対応しています。

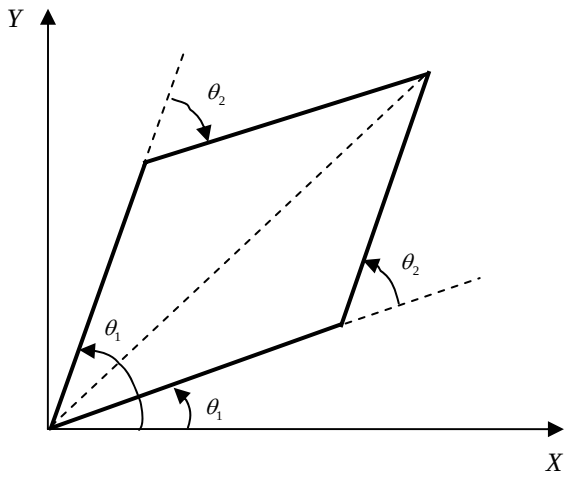


図 3-3　水平多関節ロボットの姿勢

以下、本資料ではϕの値が負となる姿勢を採用します。

逆余弦関数$\theta = \cos^{-1}(\cdots)$は$\theta$の値が 0 [rad]もしくは$\pm\pi$ [rad]に近いと数値精度が悪化するため、(3.4)式および(3.8)式を次のように変形します。

$$\theta_1 = \text{atan 2}(y, x) - \text{atan 2}\left(\sqrt{4l_1^2\left(x^2 + y^2\right) - \left(x^2 + y^2 + l_1^2 - l_2^2\right)^2}, x^2 + y^2 + l_1^2 - l_2^2\right) \tag{3.9}$$

$$\theta_2 = \text{atan 2}\left(\sqrt{4l_1^2 l_2^2 - \left(x^2 + y^2 - l_1^2 - l_2^2\right)^2}, x^2 + y^2 - l_1^2 - l_2^2\right) \tag{3.10}$$

ここで、$\text{atan 2}(\cdots)$は 4 象限逆正接関数です。

4　NXT SCARA の目標軌道作成

制御器の参照値となる、アーム先端部の目標軌道の作成手順について説明します。本章の詳細については参考文献[2]および[3]をご覧下さい。

4.1　軌道関数の作成手順

目標軌道を実現するには、軌道を表す時間の関数を作成する必要があります。目標軌道の作成方法には次の 2 通りの方法があります。

1. 位置座標を直接与える（直交座標系であれば $x(t)$，$y(t)$）。
2. 軌道を基本軌道に分割し、基本軌道関数 $\eta(t)$ を接続して全体の軌道を作成する。

基本軌道関数としては、直線や円弧等が選ばれます。

- 直線軌道：直線上の線の長さを表す時間関数を $\eta(t)$ とする。
- 円弧軌道：中心・半径を固定し、角度を表す時間関数を $\eta(t)$ とする（$\eta(t)=\varphi(t)$）。

$\eta(t)$ が設計できれば、それから位置座標 $x(t)$，$y(t)$ が定まり、それを逆運動力学により $\theta_1(t)$，$\theta_2(t)$ に変換することができます。図 4-1 は基本軌道関数を用いた目標軌道の作成手順です。

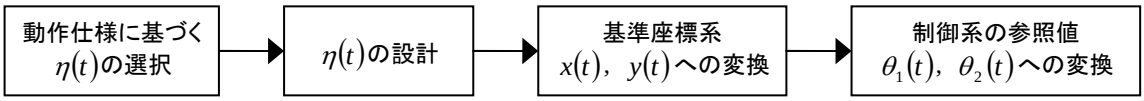


図 4-1　基本軌道関数を用いた目標軌道の作成手順

4.2　5-1-5 次多項式を用いた基本軌道関数の設計

速度波形を用いた基本軌道関数 $\eta(t)$ の設計方法について説明します。速度波形としては、台形型の加減速を改良した 5-1-5 次多項式型の加減速を採用します（図 4-2 参照）。

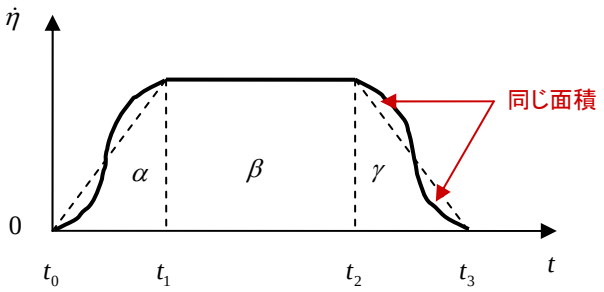


図 4-2　5-1-5 次多項式による速度波形

全移動距離は面積α,β,γの和となり、加速時間と減速時間を等しくとると$\alpha=\gamma$となります。所望の移動距離から加減速に要する距離$\alpha+\gamma$を引いた結果β'により、次の 2 つの動作に分類することができます。

- Large Motion ($\beta'>0$)
  等速区間が存在する動作。最大速度または加速時間・減速時間を再設定する必要は特にない。
- Small Motion ($\beta'\le 0$)
  等速区間が存在しない動作。移動距離の 1/2 を加速時間・減速時間となるように、最大速度または加速時間・減速時間を再設定する。

次に、図 4-2 の各区間における軌道関数の作成方法について説明します。

加速区間の軌道関数

軌道（位置）関数、速度関数、加速度関数を次式のように設定します。

$$\eta(t)=a_5t^5+a_4t^4+a_3t^3+a_2t^2+a_1t+a_0 \tag{4.1}$$
$$\dot{\eta}(t)=5a_5t^4+4a_4t^3+3a_3t^2+2a_2t+a_1 \tag{4.2}$$
$$\ddot{\eta}(t)=20a_5t^3+12a_4t^2+6a_3t+2a_2 \tag{4.3}$$

開始時刻をt_0、終了時刻をt_1とし、その位置、速度、加速度をη_0，$\dot{\eta}_0$，$\ddot{\eta}_0$，η_1，$\dot{\eta}_1$，$\ddot{\eta}_1$とすると、(4.4)式を得ます。係数行列が正則であれば、(4.5)式により係数a_5，a_4，a_3，a_2，a_1，a_0を求めることができます。

$$\begin{bmatrix}\eta_0\\ \dot{\eta}_0\\ \ddot{\eta}_0\\ \eta_1\\ \dot{\eta}_1\\ \ddot{\eta}_1\end{bmatrix}=\begin{bmatrix}t_0^5 & t_0^4 & t_0^3 & t_0^2 & t_0 & 1\\ 5t_0^4 & 4t_0^3 & 3t_0^2 & 2t_0 & 1 & 0\\ 20t_0^3 & 12t_0^2 & 6t_0 & 2 & 0 & 0\\ t_1^5 & t_1^4 & t_1^3 & t_1^2 & t_1 & 1\\ 5t_1^4 & 4t_1^3 & 3t_1^2 & 2t_1 & 1 & 0\\ 20t_1^3 & 12t_1^2 & 6t_1 & 2 & 0 & 0\end{bmatrix}\begin{bmatrix}a_5\\ a_4\\ a_3\\ a_2\\ a_1\\ a_0\end{bmatrix} \tag{4.4}$$

$$\begin{bmatrix}a_5\\ a_4\\ a_3\\ a_2\\ a_1\\ a_0\end{bmatrix}=\begin{bmatrix}t_0^5 & t_0^4 & t_0^3 & t_0^2 & t_0 & 1\\ 5t_0^4 & 4t_0^3 & 3t_0^2 & 2t_0 & 1 & 0\\ 20t_0^3 & 12t_0^2 & 6t_0 & 2 & 0 & 0\\ t_1^5 & t_1^4 & t_1^3 & t_1^2 & t_1 & 1\\ 5t_1^4 & 4t_1^3 & 3t_1^2 & 2t_1 & 1 & 0\\ 20t_1^3 & 12t_1^2 & 6t_1 & 2 & 0 & 0\end{bmatrix}^{-1}\begin{bmatrix}\eta_0\\ \dot{\eta}_0\\ \ddot{\eta}_0\\ \eta_1\\ \dot{\eta}_1\\ \ddot{\eta}_1\end{bmatrix} \tag{4.5}$$

等速区間の軌道関数

開始時刻をt_1、終了時刻をt_2とし、その位置、速度、加速度をη_1，$\dot{\eta}_1$，$\ddot{\eta}_1$，η_2，$\dot{\eta}_2$，$\ddot{\eta}_2$とします。軌道関数、速度関数、加速度関数は次のようになります。

$$\eta(t) = \dot{\eta}_1(t - t_1) + \eta_1 \tag{4.6}$$

$$\dot{\eta}(t) = \dot{\eta}_1 = \dot{\eta}_2 \tag{4.7}$$

$$\ddot{\eta}(t) = \ddot{\eta}_1 = \ddot{\eta}_2 = 0 \tag{4.8}$$

減速区間の軌道関数

減速区間の軌道関数は加速区間と同じ方法で求めることができます。

NXT SCARA の基本軌道関数の計算は cal_eta.m で行われています。

4.3 CP 動作と PTP 動作

ロボットの動作には CP（Continuous Path）動作と PTP（Pose To Pose）動作があります。両動作の特徴は表 4-1 の通りです。

表 4-1　CP 動作と PTP 動作

	CP	PTP
動作	開始と終了時の位置・姿勢に加えて、途中経路も重視する動作	開始と終了時の位置・姿勢のみを重視し、途中経路は問わない動作
例	ロボットによる塗布・研磨作業、工作機械の 3 次元表面加工	ロボットによる組立作業、HDD の位置決め制御
軌道関数	所望の軌道を作成する	最も移動時間がかかる駆動軸に基本軌道関数を適用し、その他の軸の軌道関数は全運動軸の運動が同時に開始・終了するように速度を設定する

NXT SCARA の場合、ペンによる描画が CP 動作、描画を伴わないペン位置の移動が PTP 動作となります。NXT SCARA の CP 軌道、すなわち描画する図形は次の 4 種類が用意されており、それぞれ cp_***.m で定義されています。

- 円　：　cp_circle.m
- 螺旋　：　cp_spiral.m
- スマイルマーク　：　cp_smile.m
- MATLAB ロゴ　：　cp_ml_logo.m

PTP軌道の計算はcal_ptp.mで行われおり、CP軌道とPTP軌道を結合して全軌道を求めています。NXT SCARAの軌道データの計算フローについては**6.3 軌道データ計算**を参照してください。

5　NXT SCARA の制御器設計

NXT SCARA の軌道追従制御器の設計を行います。

5.1　制御系の特徴

NXT SCARA の制御系としての特徴は次の通りです。

入出力

アクチュエータへの入力は各 DC モータの PWM デューティ値、センサからの出力は各モータの回転角度です。モータ角度を減速比で割ればリンク角度を得ることができます。

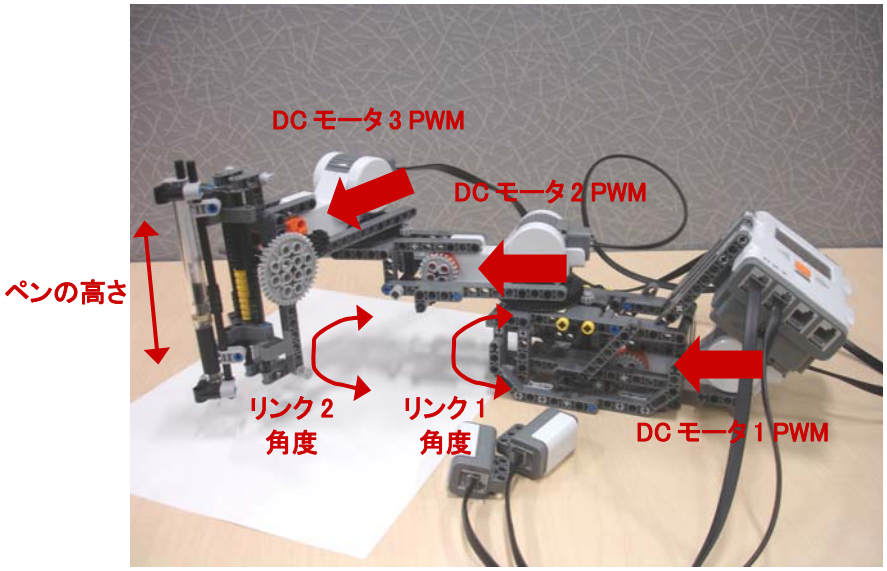


図 5-1　NXT SCARA の入出力

バックラッシュ

2.3 **ギヤトレイン・バックラッシュ**で説明した通り、リンク駆動部にはギヤトレインから生じるバックラッシュが存在します。バックラッシュがあるとモータ逆回転時に空回り状態が発生するため、位置決め精度に悪影響を及ぼします。ギヤが噛み合った状態を保つためには、バックラッシュを制御器で補償する必要があります。

5.2 制御器設計

図 5-2 は一般的なモーションコントロールシステムの構成図です。

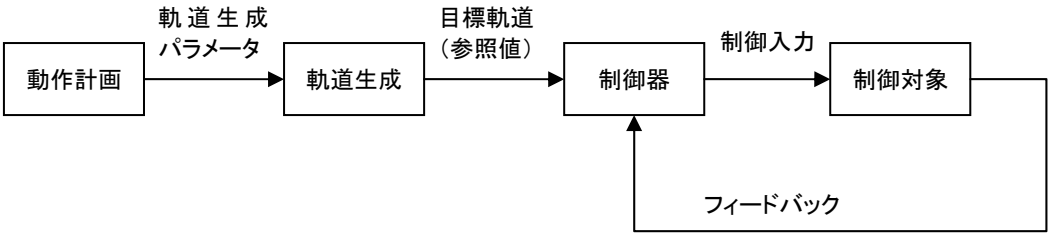


図 5-2　モーションコントロールシステム

NXT SCARA では、動作計画および軌道生成は M-ファイルを用いて行われ、制御器および制御対象のシミュレーションには Simulink モデルを使用しています。

NXT SCARA の各リンクの制御には(2.3)式に基づいた比例制御を用います。図 5-3 は NXT SCARA の軌道追従制御器のブロック線図です(実際にはリンク角度を制限するための処理も入っていますが、この図では省略しています)。

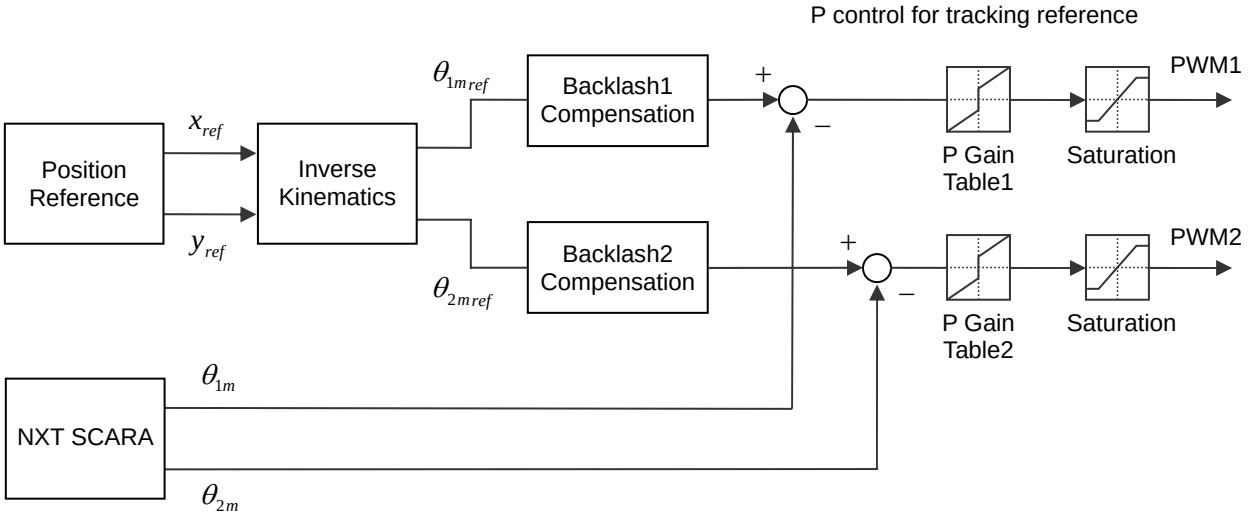


図 5-3　NXT SCARA の軌道追従制御器のブロック線図

6　NXT SCARA モデル

NXT SCARA モデルの概要およびパラメータ定義ファイルについて説明します。

6.1　モデル概要

nxtscara.mdl および nxtscara_vr.mdl は NXT SCARA の制御系全体を表したモデルです。両モデルは基本的に同じであり、Virtual Reality Toolbox による 3D 表示を含むかどうかのみが異なります。

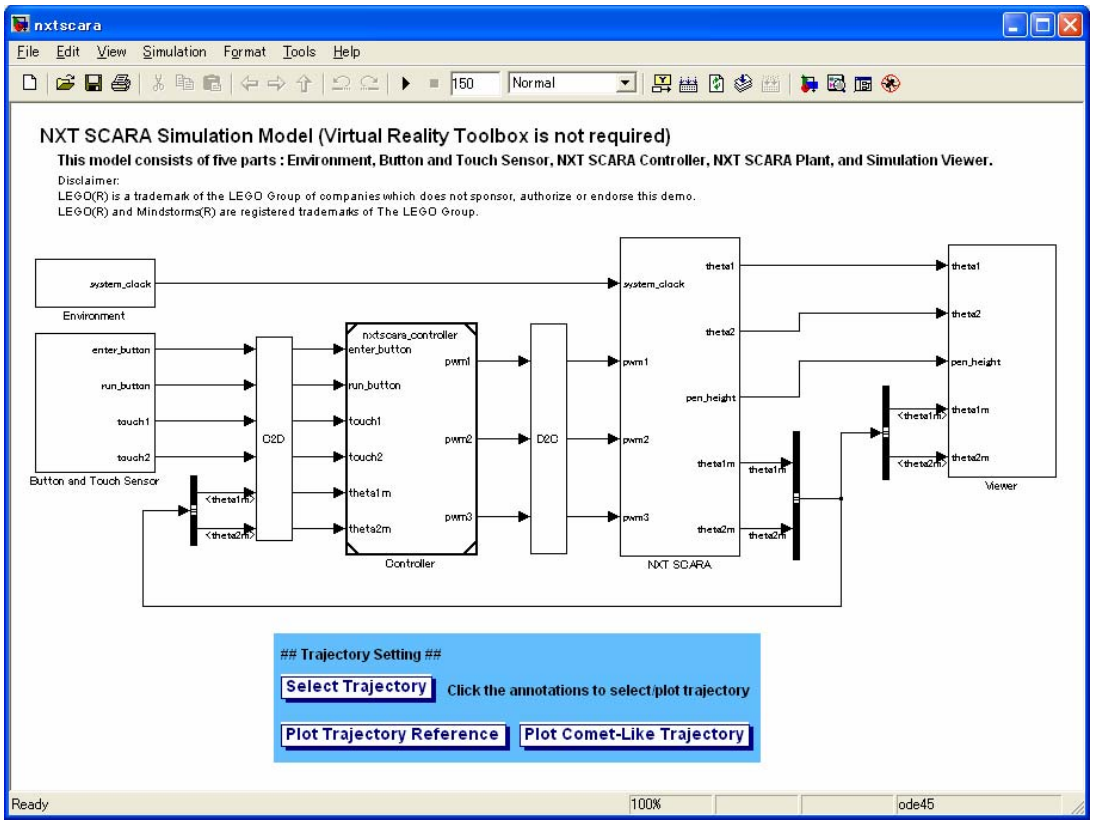


図 6-1　nxtscara.mdl

nxtscara.mdl および nxtscara_vr.mdl の主な構成要素は次の通りです。

Environment

タイマを作成しています。

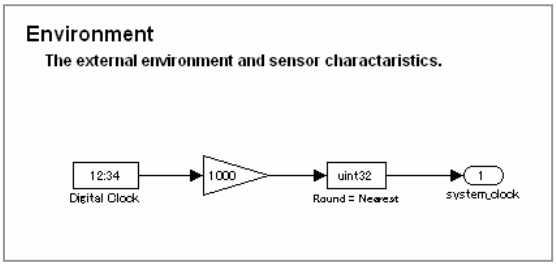


図 6-2　Environment サブシステム

Button and Touch Sensor

NXT SCARA に対するボタン・センサ信号発生器です。Signal Builder ブロックを使用して NXT 本体の Enter ボタンや Run ボタン、タッチセンサの ON/OFF を入力できるようになっています。

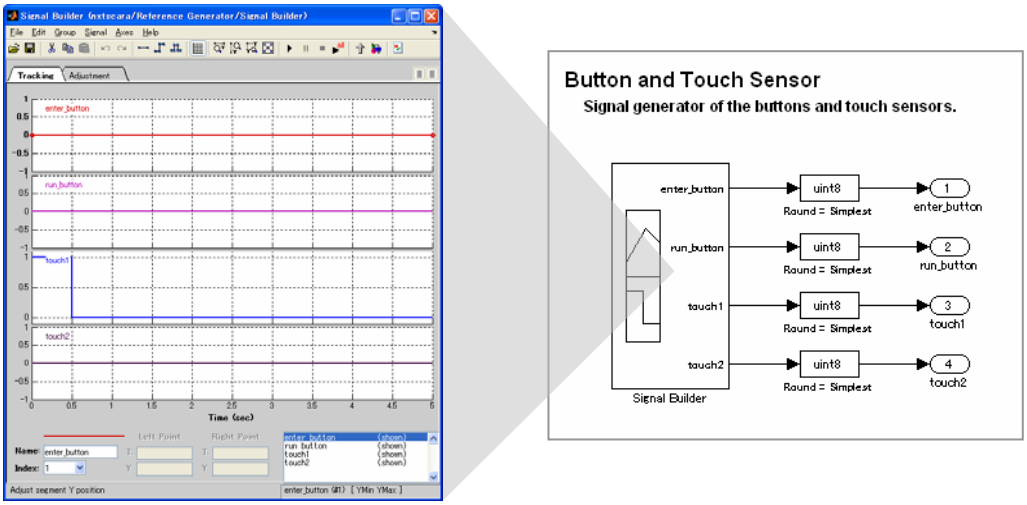


図 6-3　Button and Touch Sensor サブシステム

Controller

NXT SCARA の制御器（コントローラ）です。モデルリファレンス機能を使用して nxtscara_controller.mdlを参照しています。同モデルの詳細については**8 コントローラモデル**を参照してください。

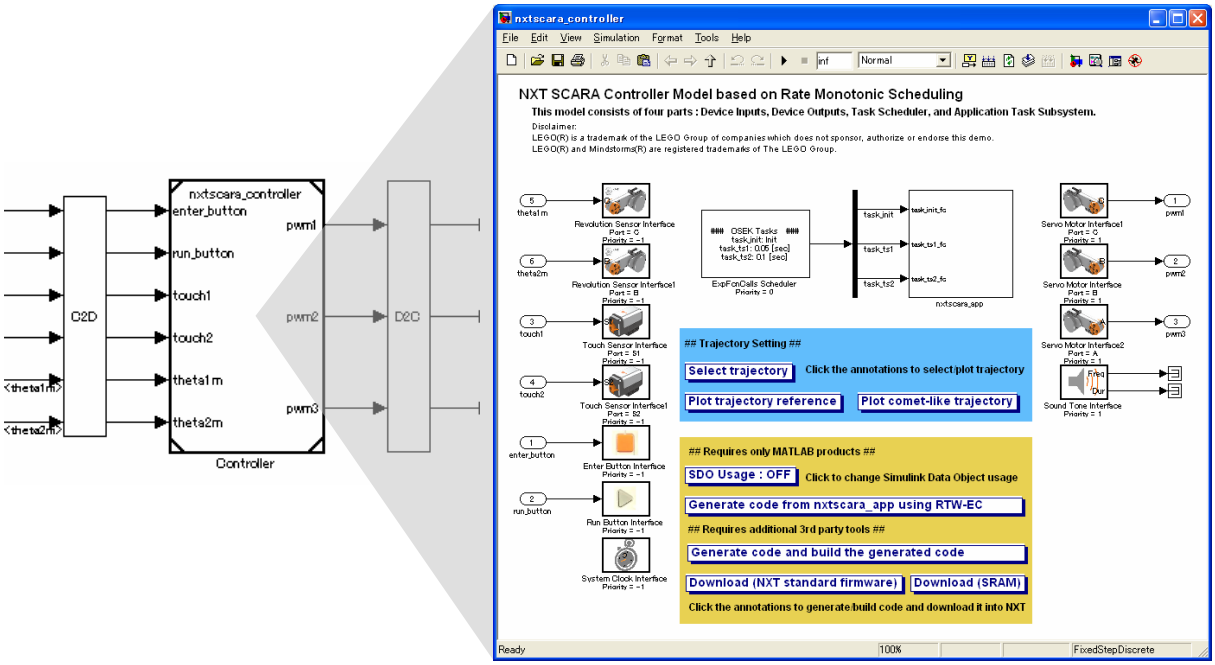


図 6-4　Controller ブロック（nxtscara_controller.mdl)

Controller ブロックは離散時間（ベースサンプル時間：TS = 1 [ms]）、NXT SCARA サブシステムは連続時間（サンプル時間：0 [ms]）で動作します。連続系と離散系が混在しているため、両者の間に Rate Transition ブロックを挿入して連続・離散変換を行っています。

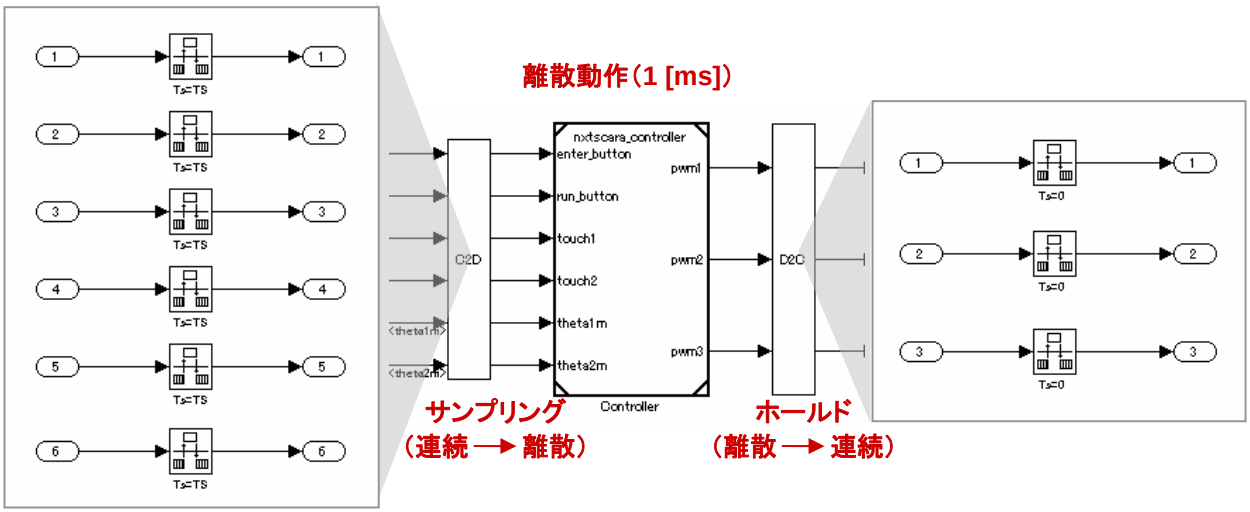


図 6-5　入出力信号の連続・離散変換

NXT SCARA

NXT SCARAのプラントモデルです。詳細については**7 プラントモデル**を参照してください。

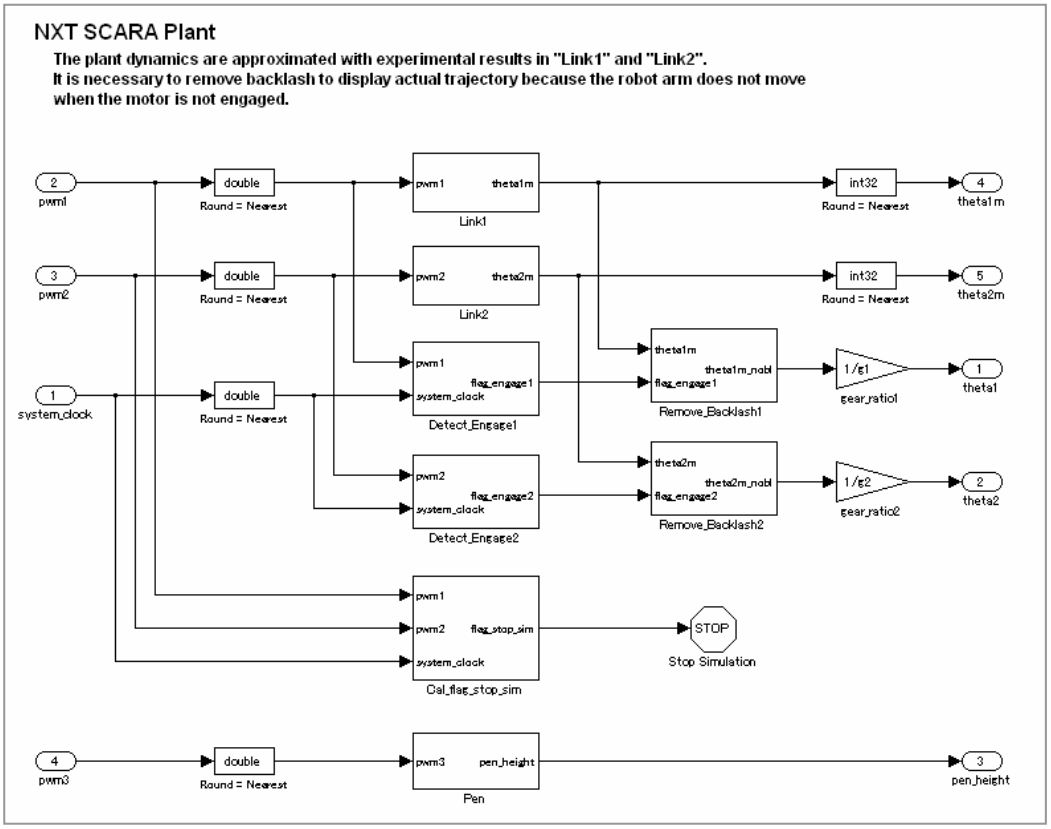


図 6-6　NXT SCARA サブシステム

Viewer

シミュレーション結果表示部です。nxtscara.mdl では XY Graph ブロックによる軌道表示、nxtscara_vr.mdl では Virtual Reality Toolbox による 3D 表示を行うことができます。

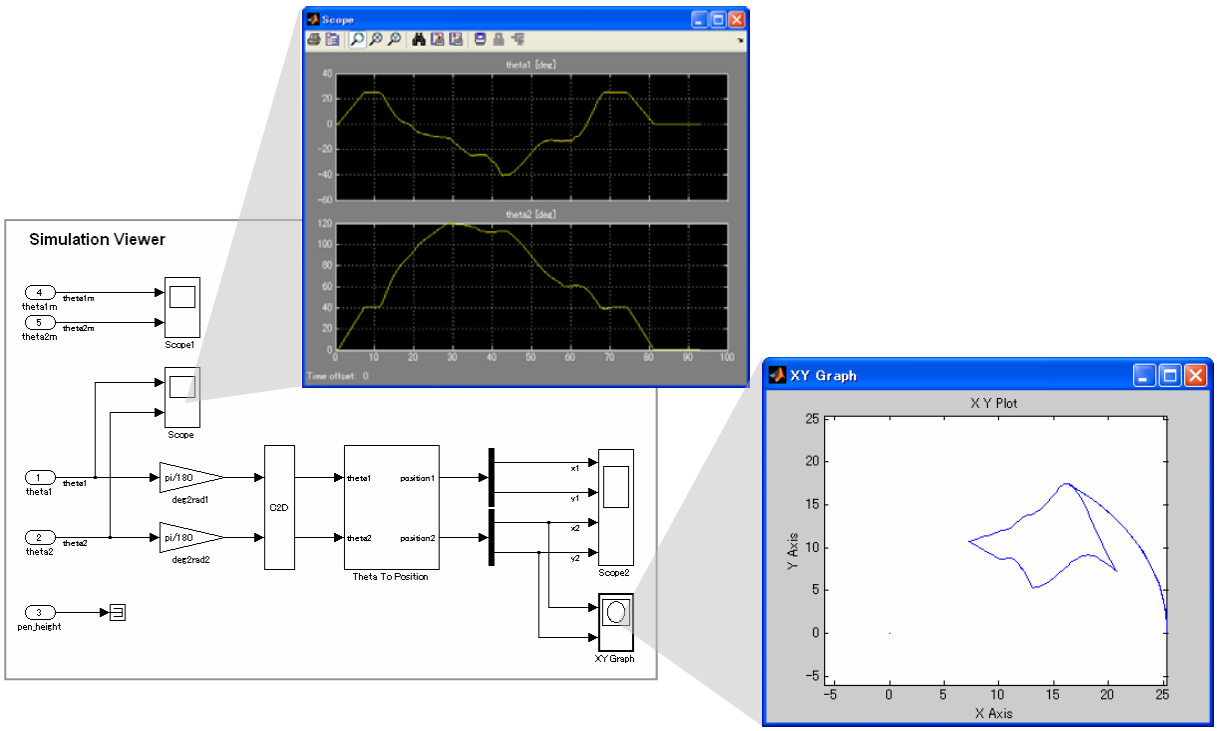


図 6-7　Viewer サブシステム (nxtscara.mdl)

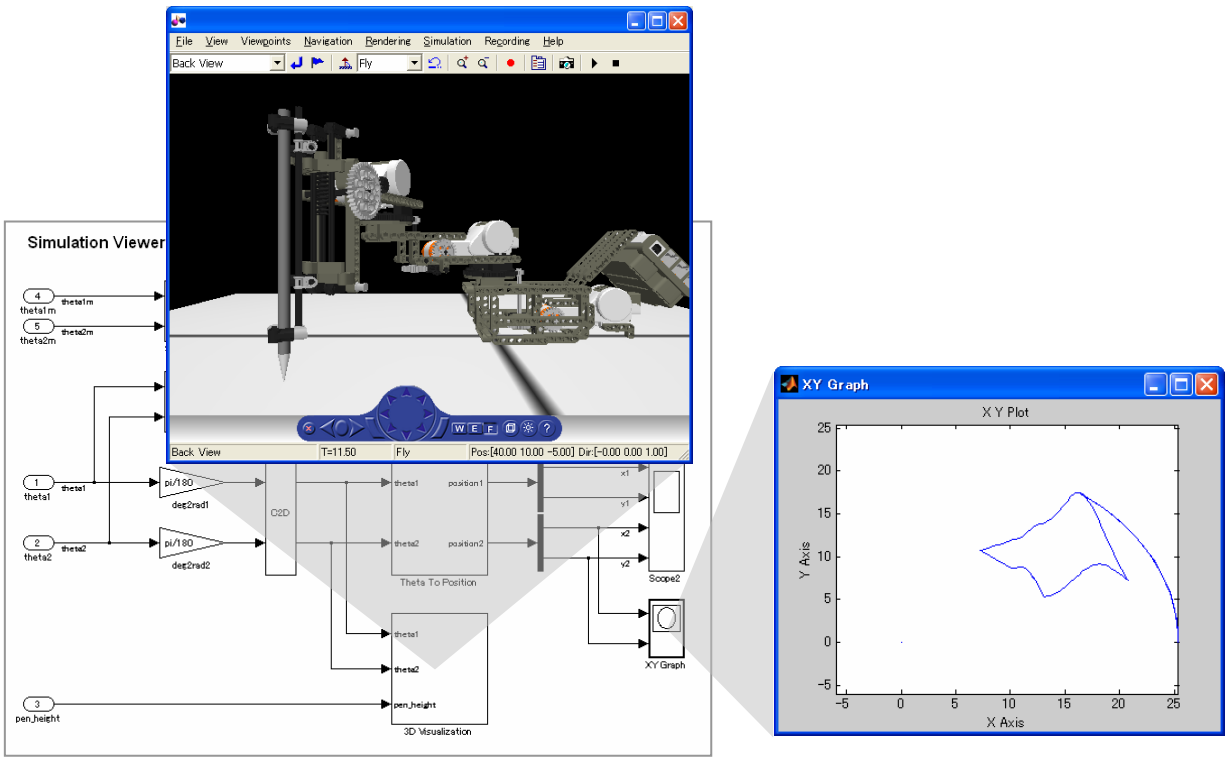


図 6-8　Viewer サブシステム (nxtscara_vr.mdl)

Trajectory Setting

目標軌道（描画する図形）の選択、プロットを行うことができます。4 種類の軌道（円、螺旋、スマイルマーク、MATLAB ロゴ）が用意されています。

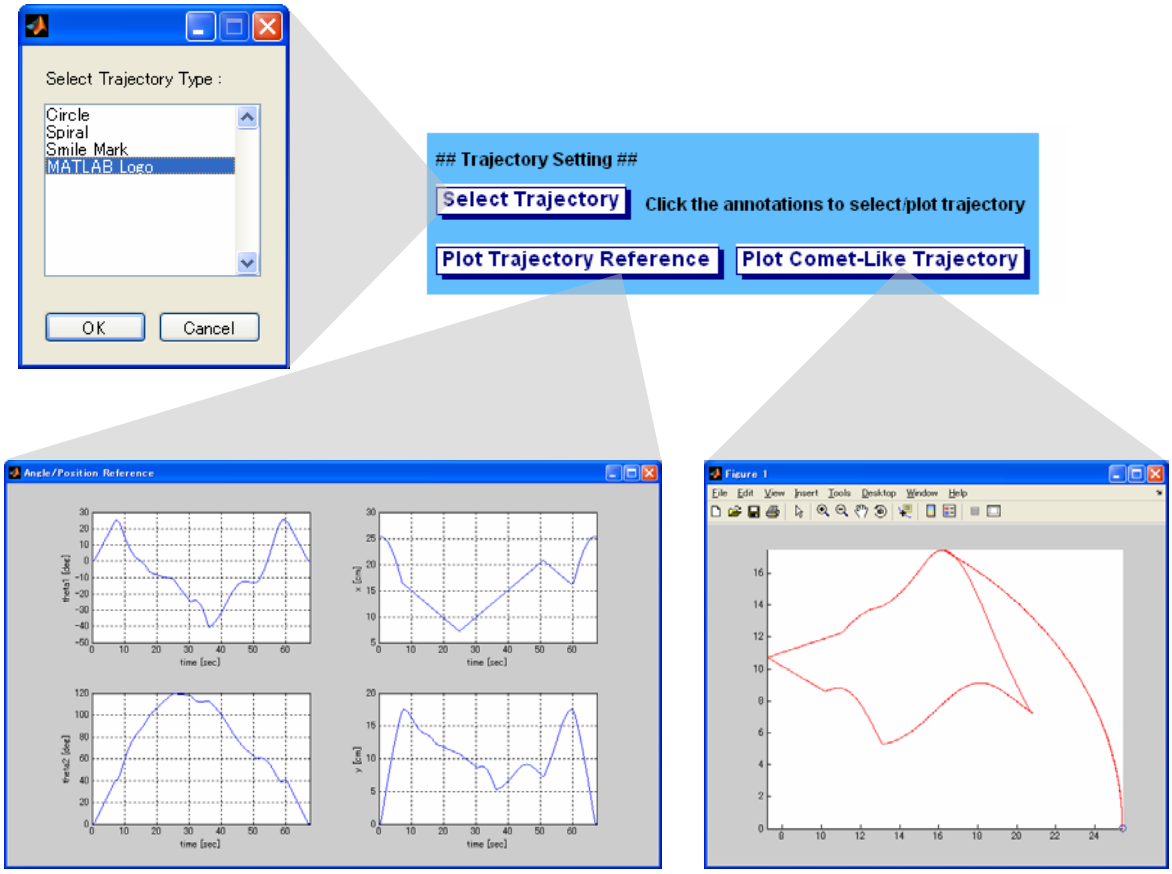


図 6-9　Trajectory Setting

6.2 パラメータ定義

表 6-1 にシミュレーション・コード生成に必要なパラメータを定義する M-ファイルを示します。

表 6-1　パラメータ定義ファイル

ファイル名	内容
cal_cp_ptp.m	CP 軌道および PTP 軌道計算用 M-関数
param_controller.m	制御器（コントローラ）パラメータ定義用 M-スクリプト
param_nxtscara.m	NXT SCARA パラメータ定義ファイル（param_***.m を実行）
param_plant.m	制御対象（プラント）パラメータ定義用 M-スクリプト
param_ref.m	目標座標および目標角度計算用 M-スクリプト
param_sim.m	シミュレーションパラメータ定義用 M-スクリプト

param_nxtscara.m を実行すると、param_***.m（***は controller, plant, ref, sim）および cal_cp_ptp.m が呼び出され、パラメータおよび軌道データがワークスペース上に定義されます。

param_nxtscara.m

```
% Load NXT SCARA Parameters

param_plant          % Plant Parameters
param_controller     % Controller Parameters
param_sim            % Simulation and Virtual Reality Parameters

% Reference Parameters (Circle)
cp_ptp = cal_cp_ptp('Circle', ts1, l1, l2);
param_ref
```

デモモデルでは、モデル起動時に param_nxtscara.m を自動実行するようにモデルコールバック関数を設定しています。

> モデルコールバック関数を確認するには、モデルウィンドウの［**ファイル**］メニューから［**モデルプロパティ**］を選択後、［**コールバック**］を選択してください。

6.3 軌道データ計算

表 6-2 に軌道データ計算用 M-ファイルを示します。

表 6-2　軌道データ計算用ファイル

ファイル名	内容
cal_cp_ptp.m	CP 軌道および PTP 軌道計算用 M-関数
cal_eta.m	基本軌道計算用 M-関数
cal_ptp.m	PTP 軌道計算用 M-関数
cal_time_data.m	終了時間・ペン操作時間計算用 M-関数
cat_cp_ptp.m	CP 軌道・PTP 軌道結合用 M-関数
chk_limit.m	角度・角速度制限チェック用 M-関数
cp_circle.m	CP 軌道計算用 M-関数（円）
cp_ml_logo.m	CP 軌道計算用 M-関数（MATLAB ロゴ）
cp_smile.m	CP 軌道計算用 M-関数（スマイルマーク）
cp_spiral.m	CP 軌道計算用 M-関数（螺旋）
ml_logo.mat	MATLAB ロゴ用座標データ
theta2xy.m	角度・座標変換用 M-関数
xy2theta.m	座標・角度変換用 M-関数

軌道データの計算フローは図 6-10 の通りです。CP 軌道および PTP 軌道の計算には基本軌道関数（cal_eta.m）を用いています。

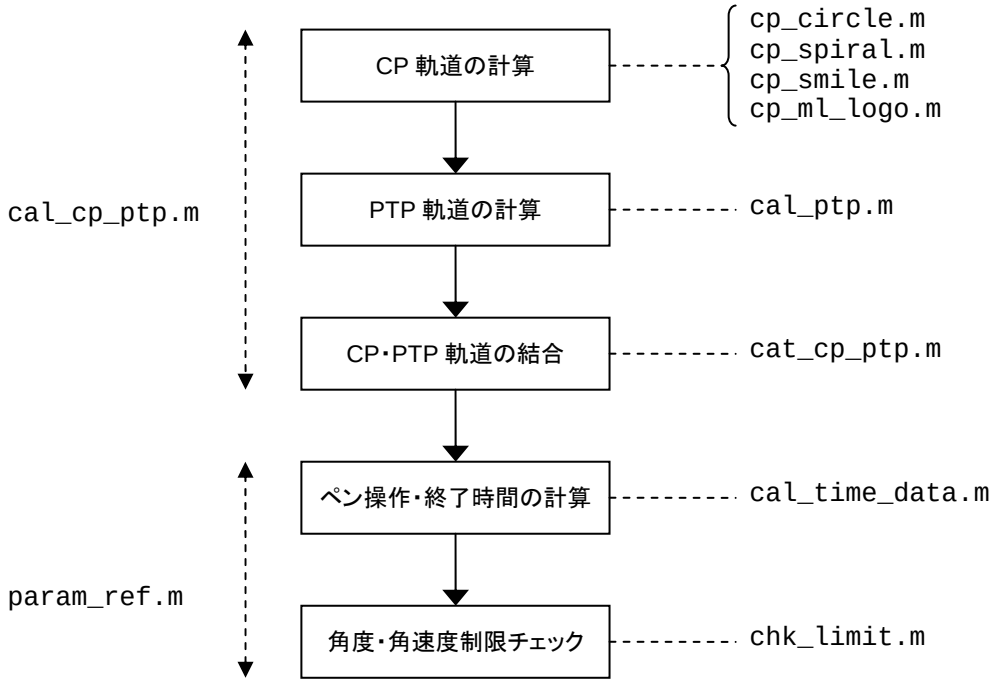


図 6-10　軌道データの計算フロー

7　プラントモデル

制御対象（プラント）モデルである NXT SCARA サブシステムについて説明します。

7.1　モデル概要

NXT SCARA サブシステムは次の 3 要素から構成されています。

- プラント（リンク 1、リンク 2、ペン）
- バックラッシュ検知・除去
- シミュレーション停止

制御器（コントローラ）からの入力信号のデータタイプを double に変換してから倍精度浮動小数点演算を行い、その結果を適当に量子化してからコントローラへ出力しています。

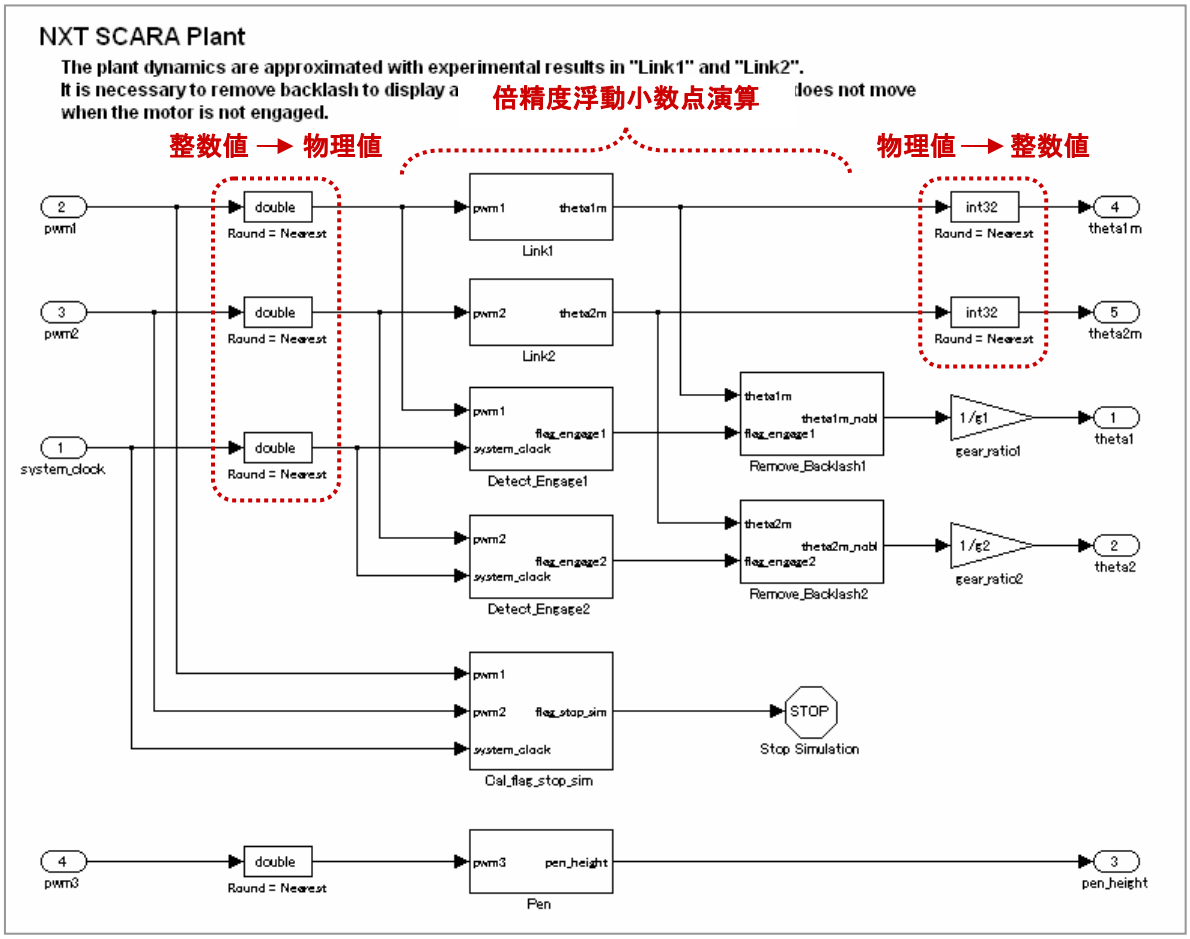


図 7-1　NXT SCARA サブシステム

7.2 プラント

リンク 1&リンク 2

各リンクの運動方程式は(2.3)式に従うものとしてモデリングしています。

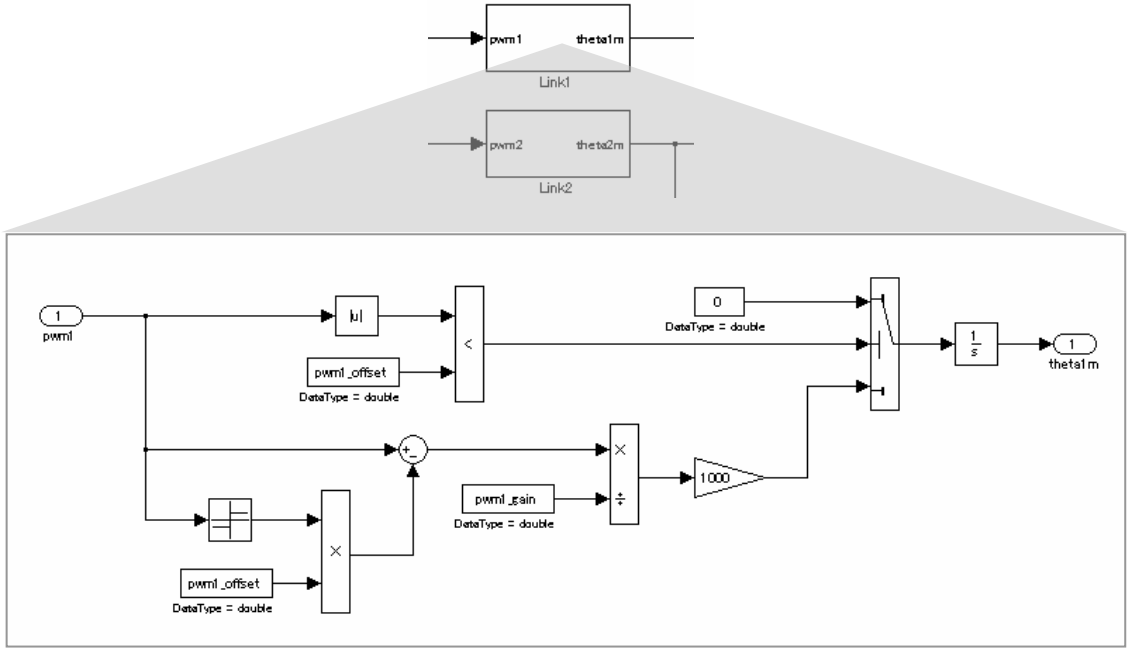


図 7-2　Link1 サブシステム

ペン

3D 表示用にペンの高さを求めています。

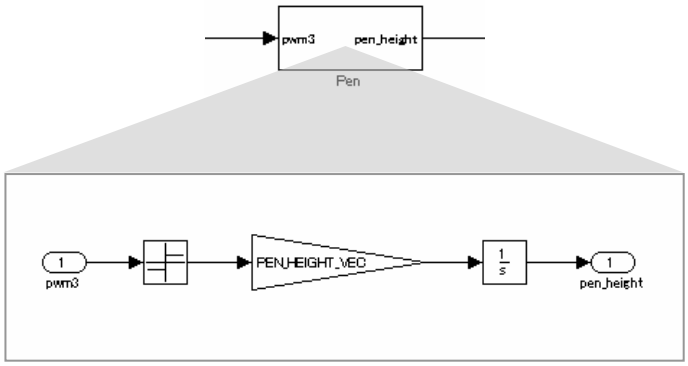


図 7-3　Pen サブシステム

7.3 バックラッシュ検知・除去

モータ回転角度にはバックラッシュ補正のための余分な回転が含まれています。このバックラッシュ補正分を除去して各リンクの回転角度を求めています。

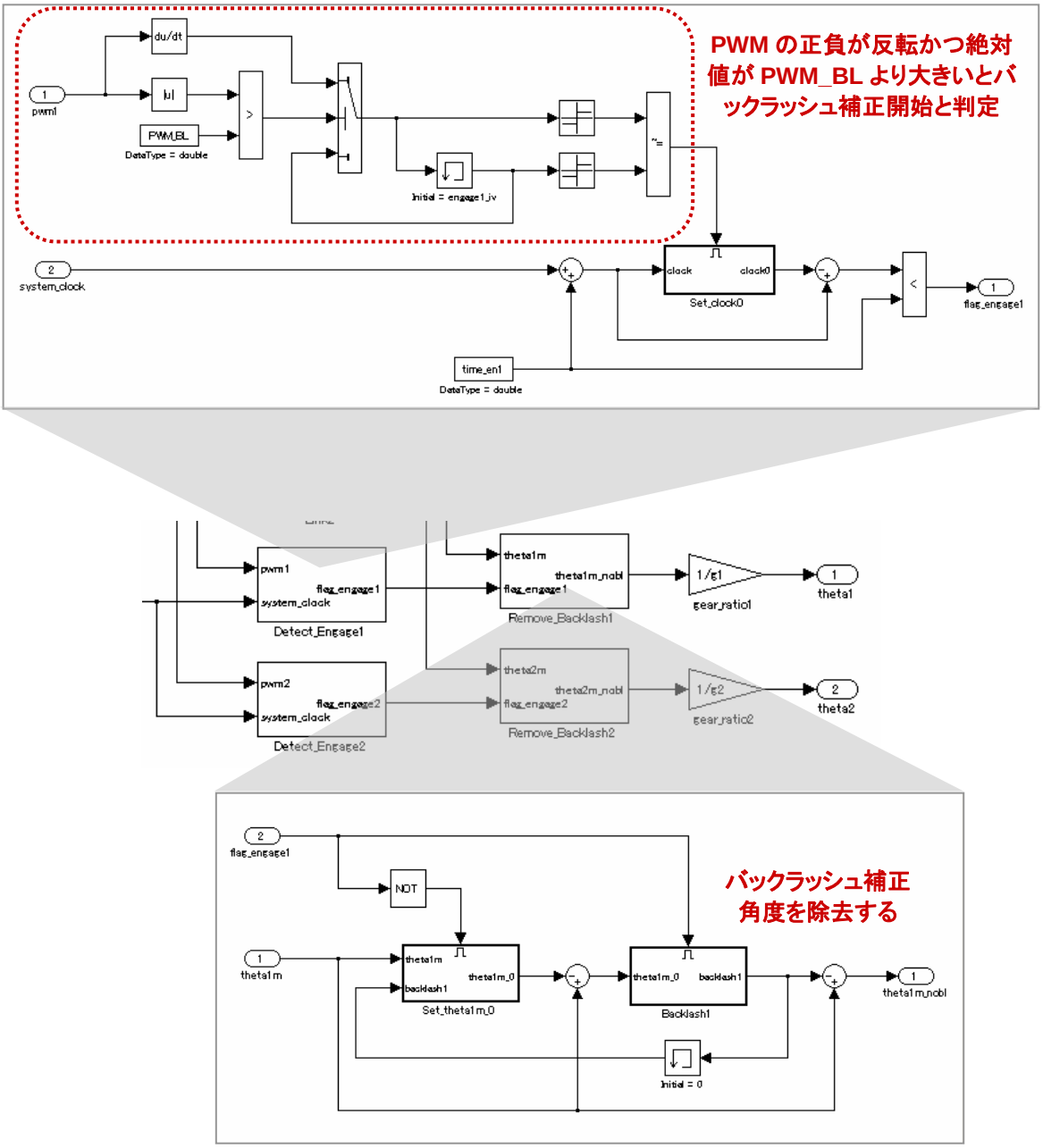


図 7-4　Detect_Engage1 / Remove_Backlash1 サブシステム

7.4 シミュレーション停止

PWM1 と PWM2 の計算値が共に 0 になってから `TIME_STOP_SIM` [ms] 経過するとシミュレーションが停止するようにモデリングしています。

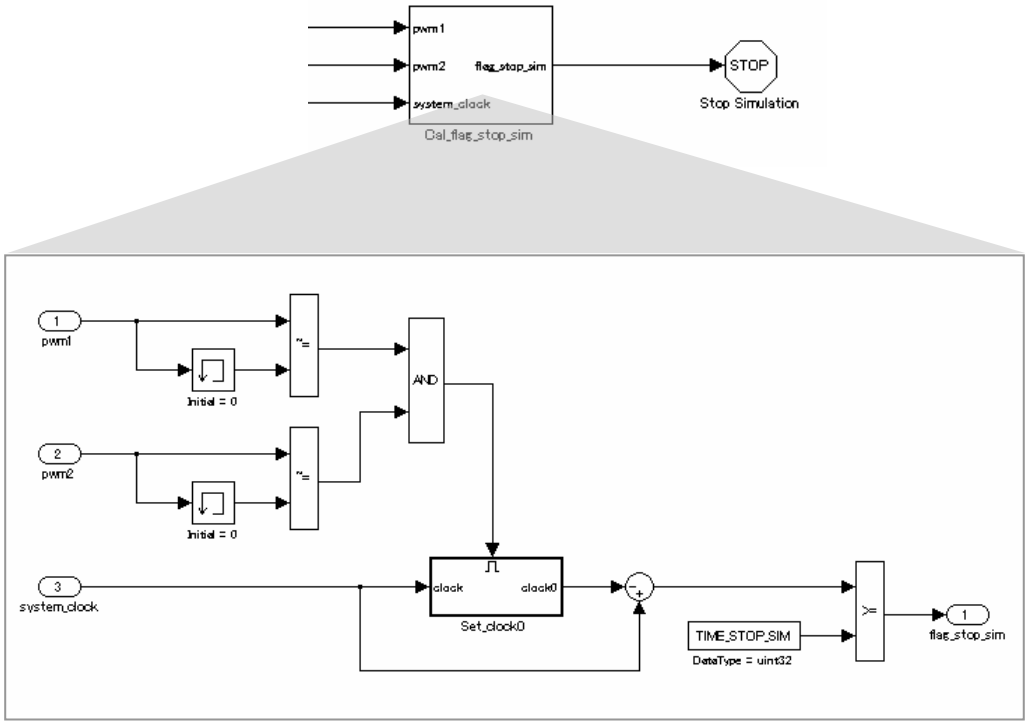


図 7-5　Cal_flag_stop_sim サブシステム

8　コントローラモデル

制御器（コントローラ）モデルである nxtscara_controller.mdl の制御プログラム・タスク構成・モデル内容について説明します。

8.1　制御プログラム概要

ステートマシン図

図 8-1 は制御プログラムのステートマシン図です。nxtscara_controller.mdl に実装されている制御プログラムには軌道追従モードと調節モードの 2 つの動作モードが用意されています。

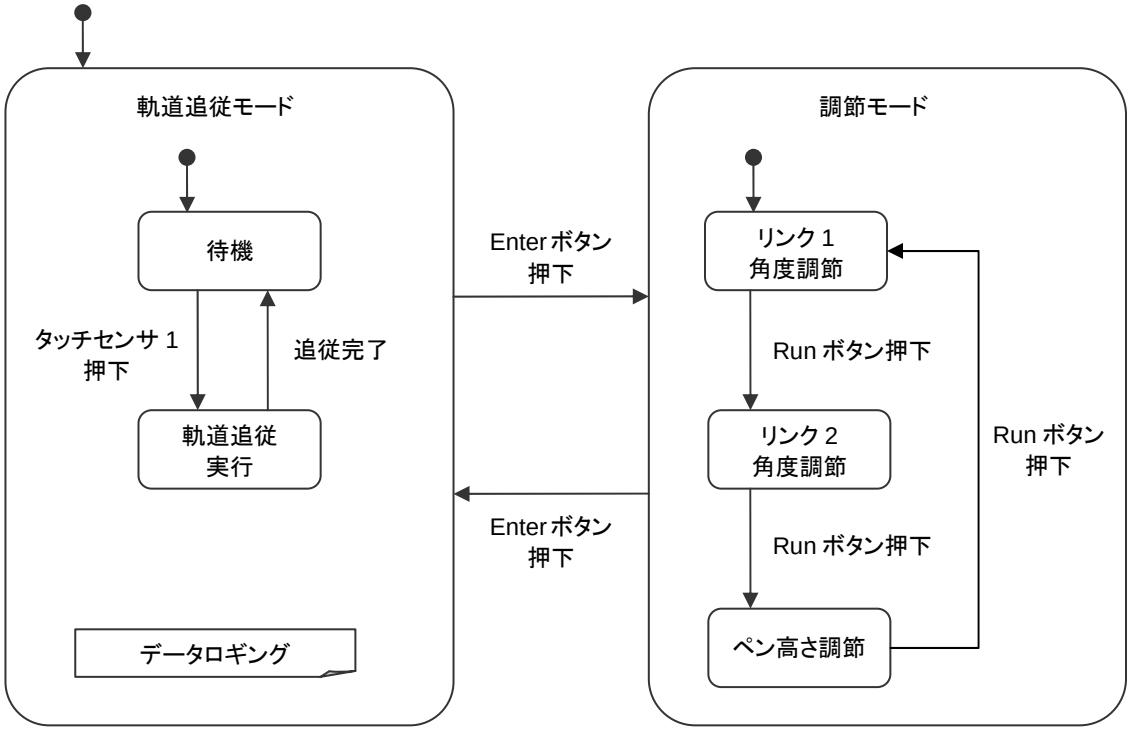


図 8-1　制御プログラムのステートマシン図

調節モードの各サブステートでは、タッチセンサによりリンク角度およびペンの高さを調節することができます。図 8-2 はそのステートマシン図です。

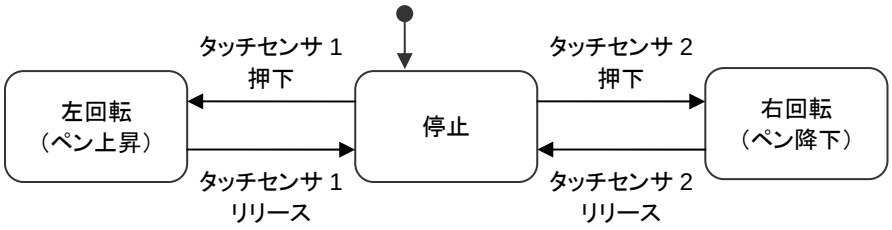


図 8-2　調節モード内サブステートのステートマシン図

タスク構成

制御プログラムを表 8-1 の 3 つのタスクで構成します。

表 8-1　制御プログラムのタスク構成

タスク名	駆動タイミング	主な処理
task_init	起動時のみ	初期値設定
task_ts1	50 [ms] 周期	軌道追従制御 リンク角度・ペン高さ調節 データロギング
task_ts2	100 [ms] 周期	動作モード変更（軌道追従／調節モード） 調節モード時モータ変更

軌道追従制御には**5.2 制御器設計**で説明した制御器を使用します。

データタイプ

演算誤差を抑える目的で、軌道追従制御演算を単精度浮動小数点データで行うことにします。LEGO Mindstorms NXT に搭載されている ARM 7 プロセッサには FPU（浮動小数点演算装置）が搭載されていませんが、GCC の浮動小数点演算ライブラリを用いてソフトウェア的に単精度浮動小数点演算を行うことができます。

8.2 モデル概要

nxtscara_controller.mdl は Embedded Coder Robot NXT フレームワークに基づいてモデリングされています。

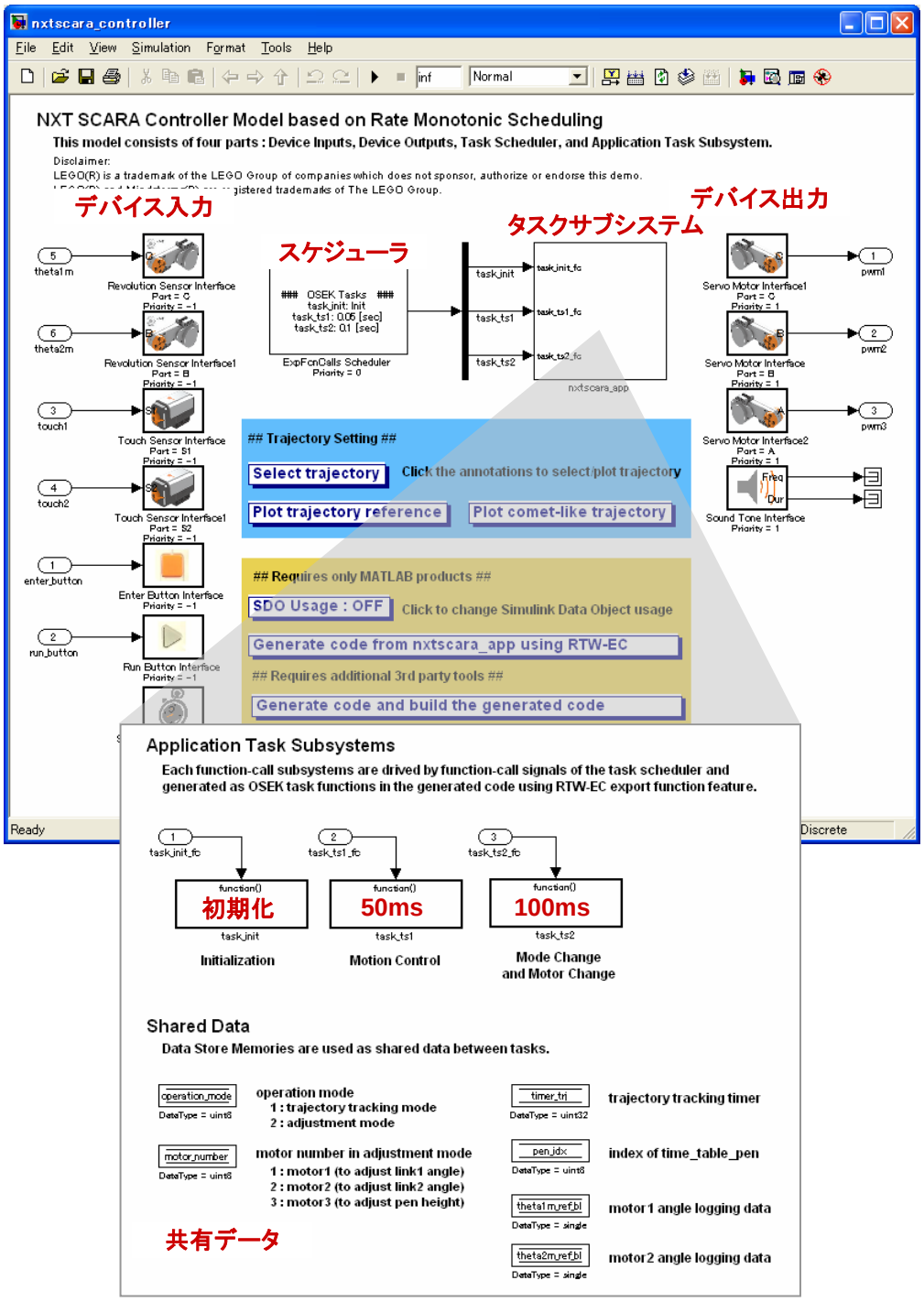


図 8-3　nxtscara_controller.mdl

デバイスインタフェース

Embedded Coder Robot NXT ライブラリに用意されている各種センサ・アクチュエータ用ブロックを用いてデバイス入出力インタフェースを作成しています。

図 8-4　DC モータの入出力インタフェース

スケジューラ＆タスク

ExpFcnCalls Scheduler ブロックを用いてタスク設定（タスク名、サンプル時間、プラットフォーム、スタックサイズ）を行い、同ブロックから出力される Function-Call 信号を Function-Call Subsystem に接続してタスク用サブシステムを作成しています。

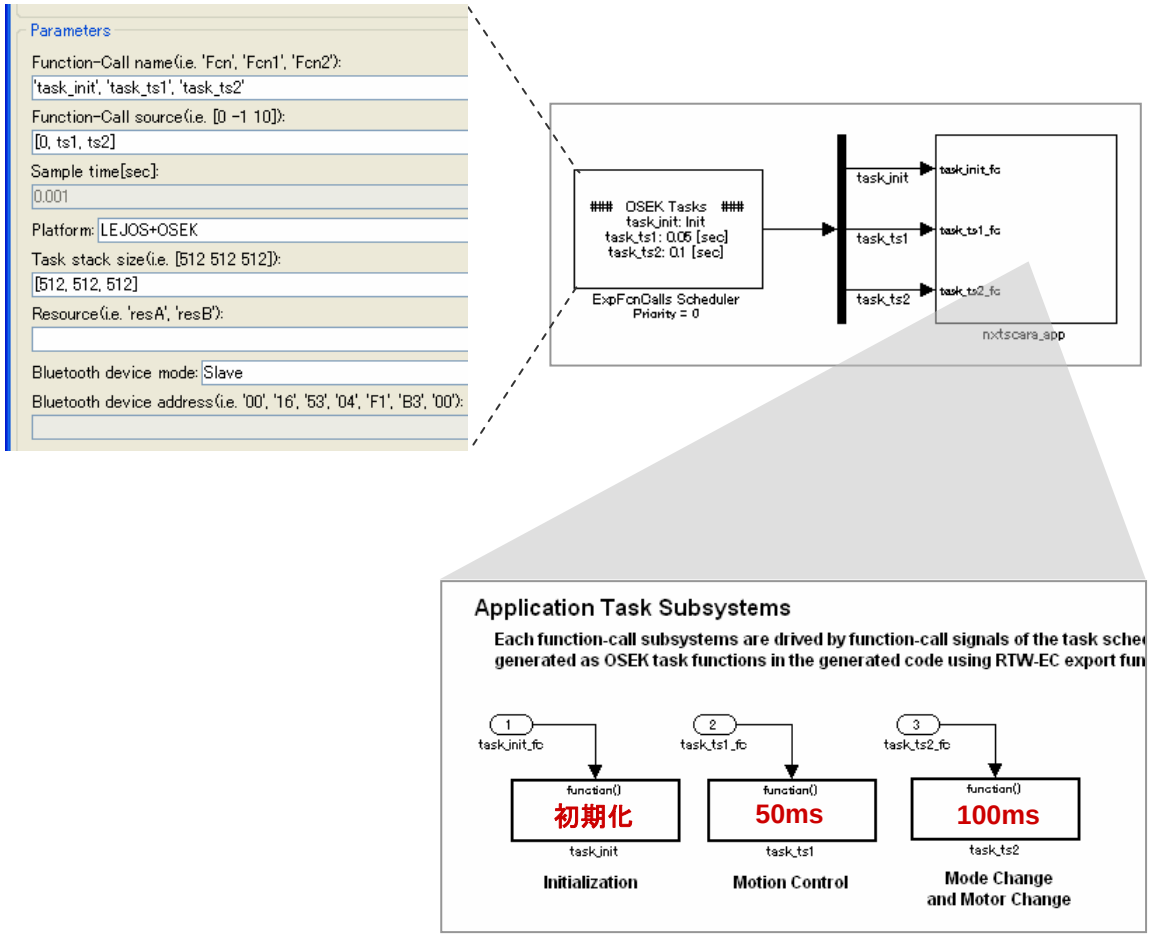


図 8-5　スケジューラ＆タスク

優先度

デバイス入力→タスク→デバイス出力の順で処理が行われるように、各ブロックに優先度を設定しています。数値が小さいと優先度が高いことを示します。負の値も設定可能です。

> 優先度を設定するには、ブロックを右クリックして表示されるコンテキストメニューから [**ブロックプロパティ**] を選択してください。

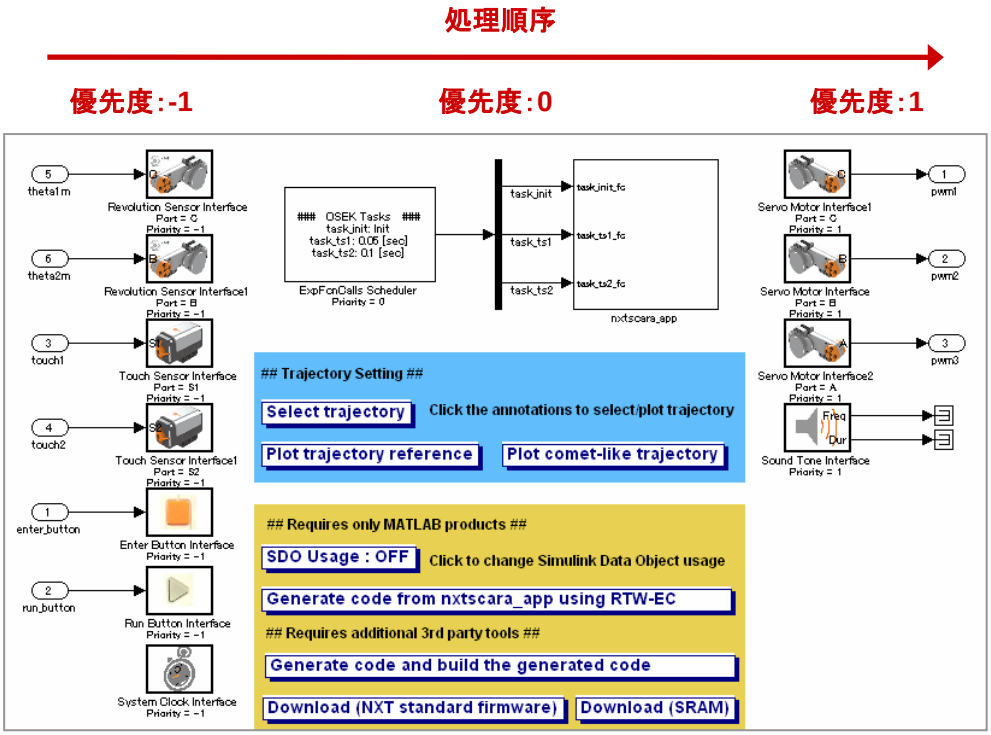


図 8-6　優先度設定による処理順序

共有データ

タスク間で共有するデータとして Data Store Memory を使用しています。

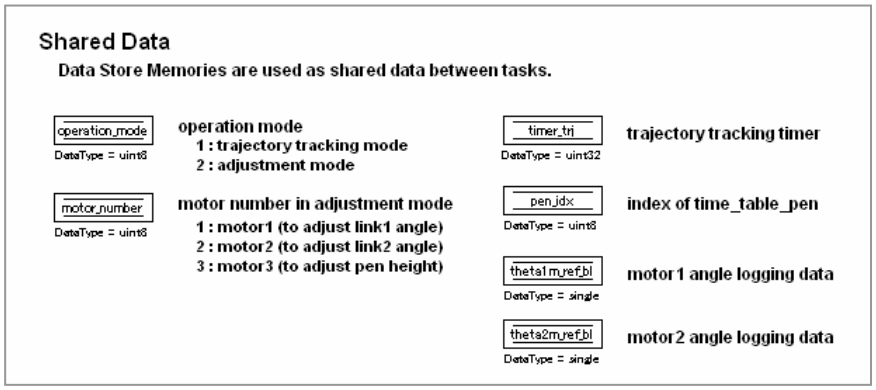


図 8-7　共有データ

8.3 初期化タスク：task_init

初期値設定を行うタスクです。

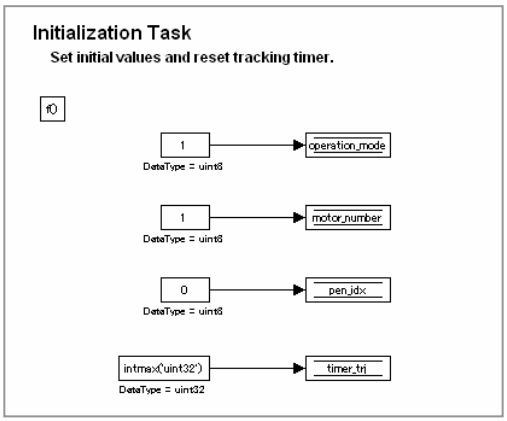


図 8-8　task_init サブシステム

8.4 50ms タスク：task_ts1

軌道追従モードおよび調節モードを含むタスクです。`operation_mode = 1` で軌道追従モード、`operation_mode = 2` で調節モードになります。

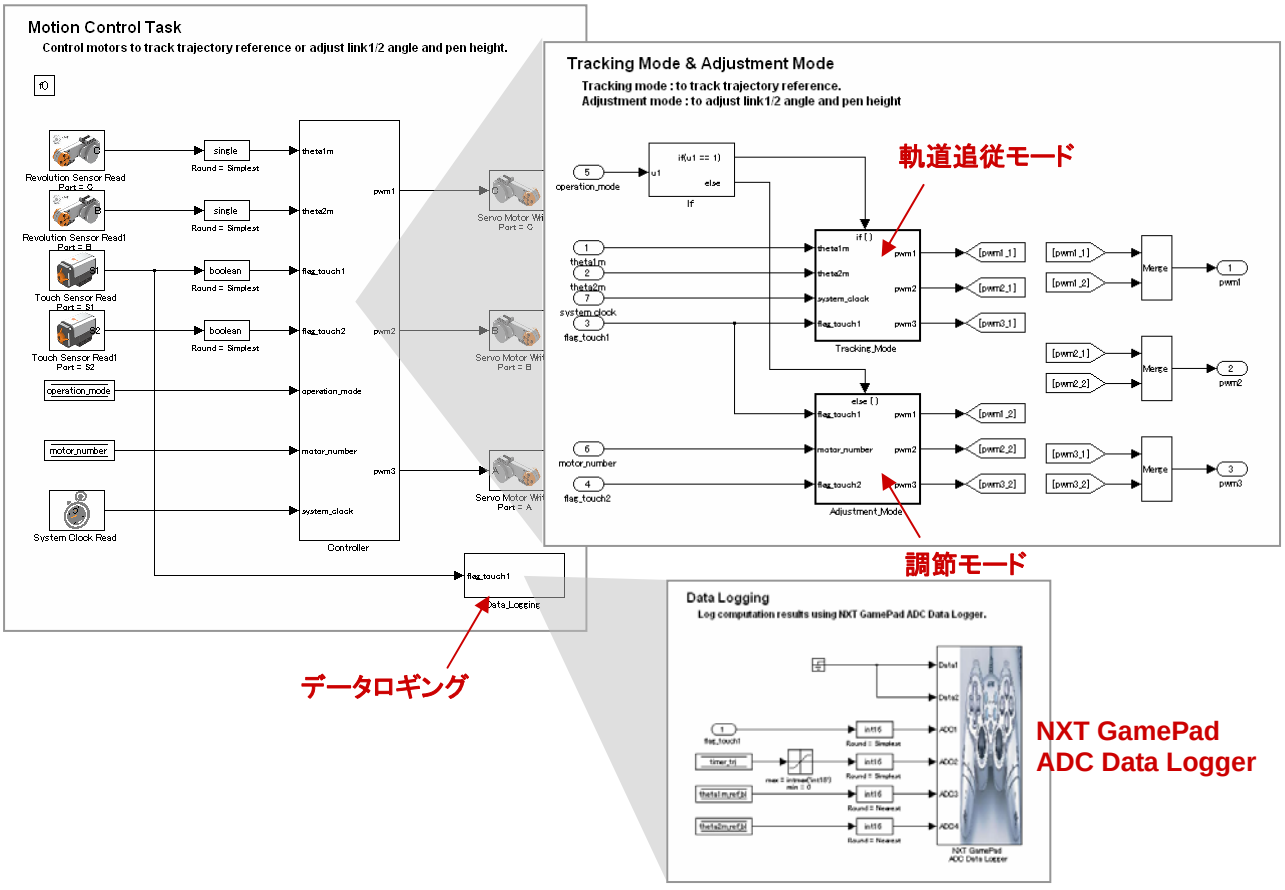


図 8-9　task_ts1 サブシステム

軌道追従モード

軌道追従モードではタイマ変数 `timer_trj` を用いて状態遷移が行われます。

1. 初期状態は待機状態、`timer_trj` の初期値は符号無し 32 ビットデータの最大値です。
2. タッチセンサ 1 が押されると `timer_trj` が 0 に設定されて軌道追従実行状態に遷移します。
3. 軌道追従実行状態では `timer_trj` がカウントアップされます。その値が終了時刻 `time_finish` を過ぎると待機状態に戻り、`timer_trj` が初期値にリセットされます。

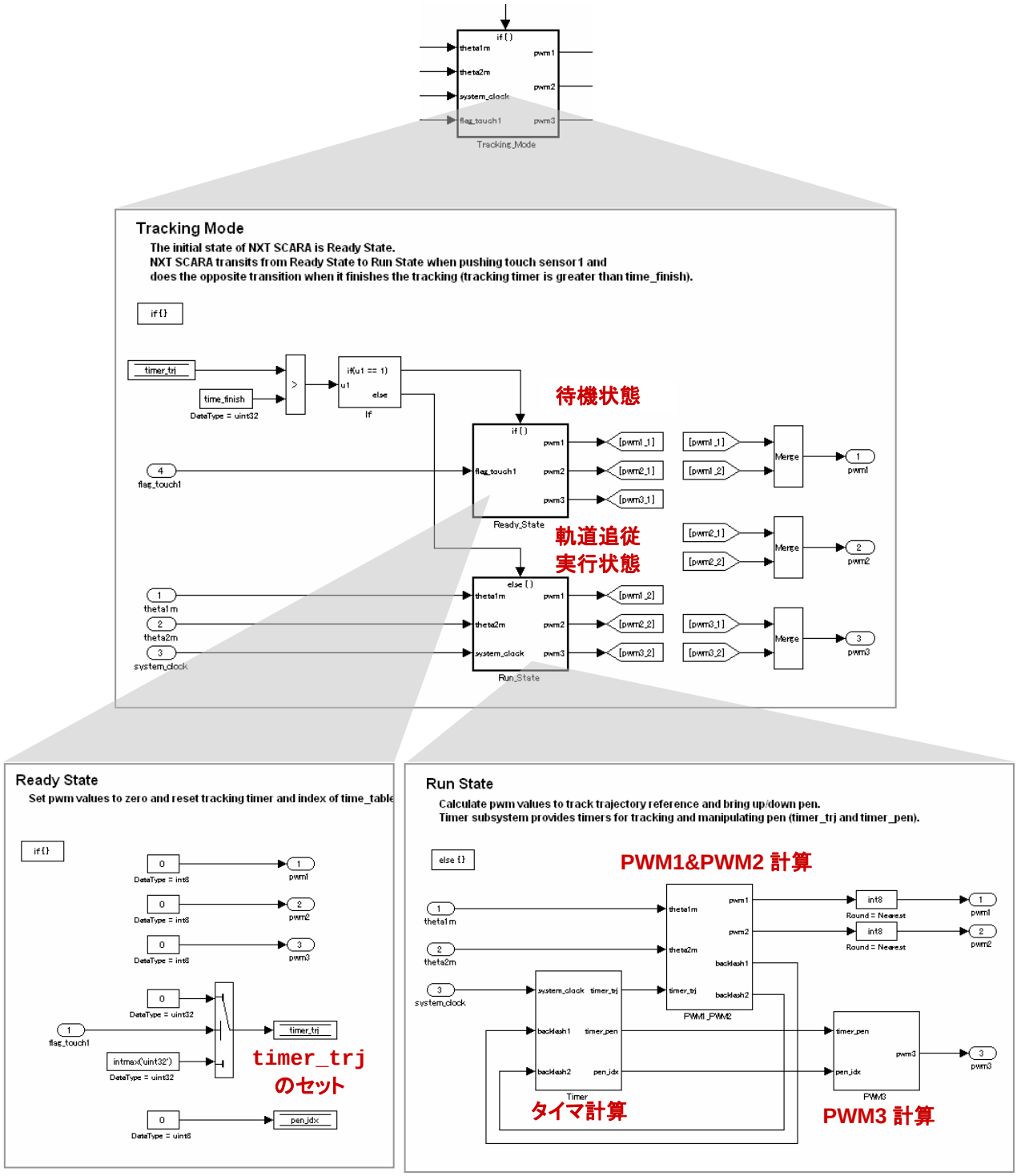


図 8-10　軌道追従モード（Tracking_Mode サブシステム）

タイマ計算（Timer）

軌道追従用タイマ変数 `timer_trj` および各種タイマ変数を計算しています。`timer_en1`、`timer_en2` は DC モータ 1、2 のバックラッシュ補正を開始した時刻で 0 初期化されるタイマ変数、`timer_pen` はペン操作時刻で 0 初期化されるタイマ変数となっています。バックラッシュ補正およびペン操作時に目標軌道座標を同位置に保つため、バックラッシュ補正時間（`time_en1`、`time_en2`）およびペン操作時間（`time_pen`）中は `timer_trj` のカウントを止めています。図 8-11 は各タイマ変数の時間変化の模式図です。

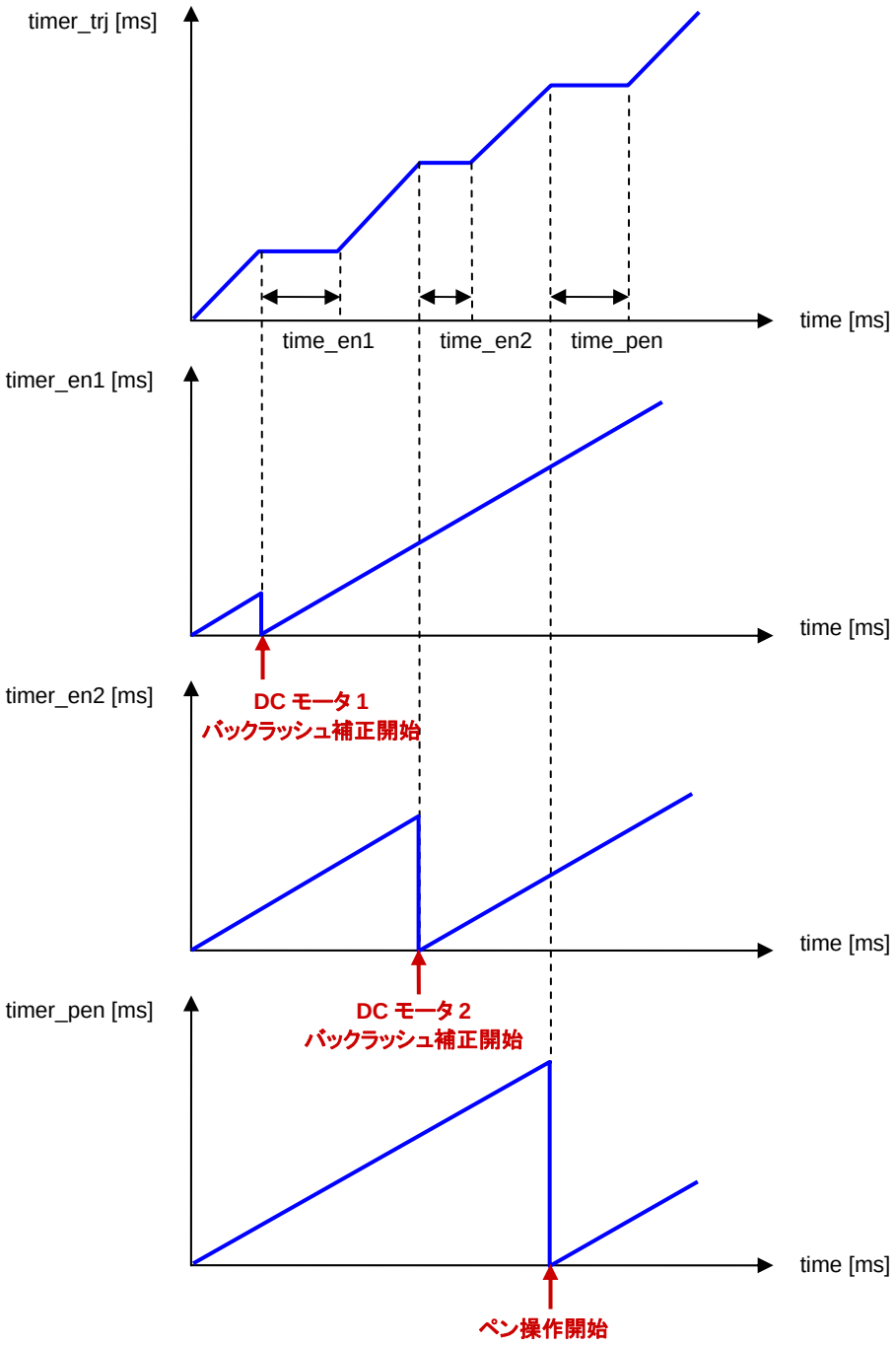


図 8-11　タイマ変数の時間変化の模式図

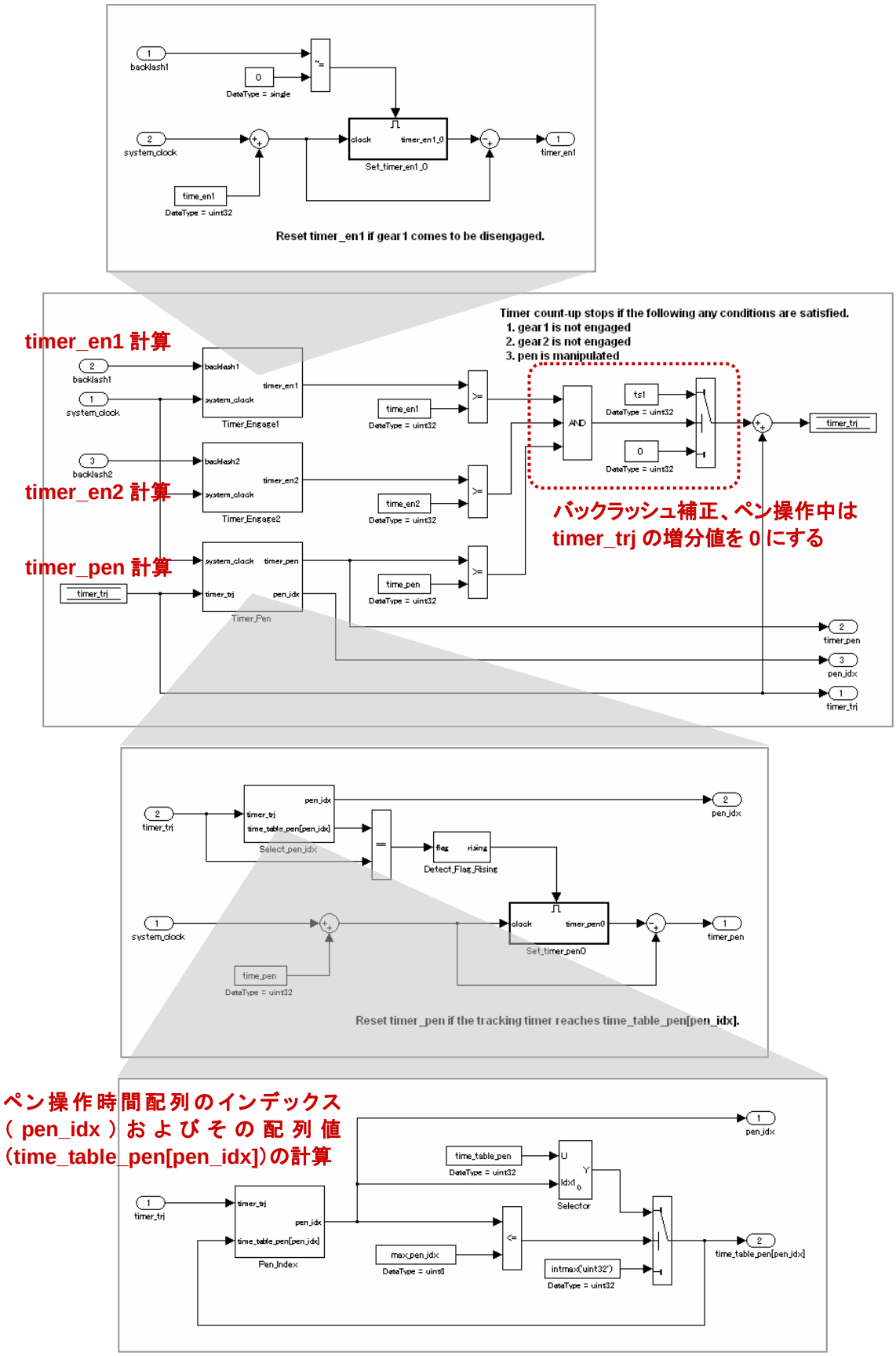


図 8-12　タイマ計算（Timer サブシステム）

PWM1&PWM2 計算（PWM1_PWM2）

図 5-3 のブロック線図を基に PWM デューティ値を計算しています。

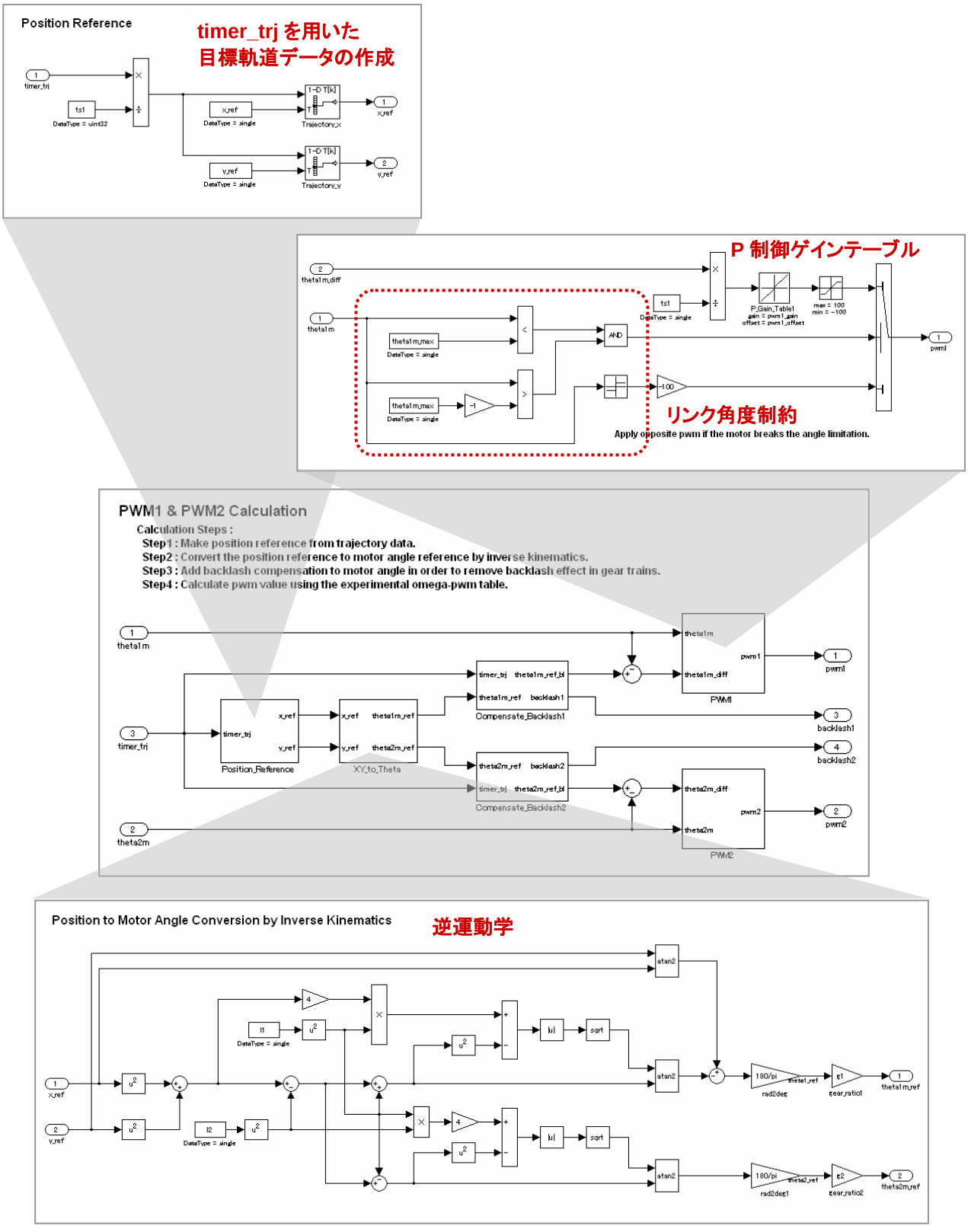


図 8-13　PWM1&PWM2 計算（PWM1_PWM2 サブシステム）

バックラッシュ補正（Compensate_Backlash1 & Compensate_Backlash2）

ギヤトレインのエンゲージ状態を保存しておき、参照モータ角度差の正負が反転かつその絶対値が dthetam_bl [deg]よりも大きい時にバックラッシュ補正を行います（dthetam_bl は演算誤差抑止のための閾値）。バックラッシュ補正では、参照モータ角度に適当な角度（backlash1/2）を加減しています。また、軌道追従終了後に初期エンゲージ状態を回復する処理を行っています。

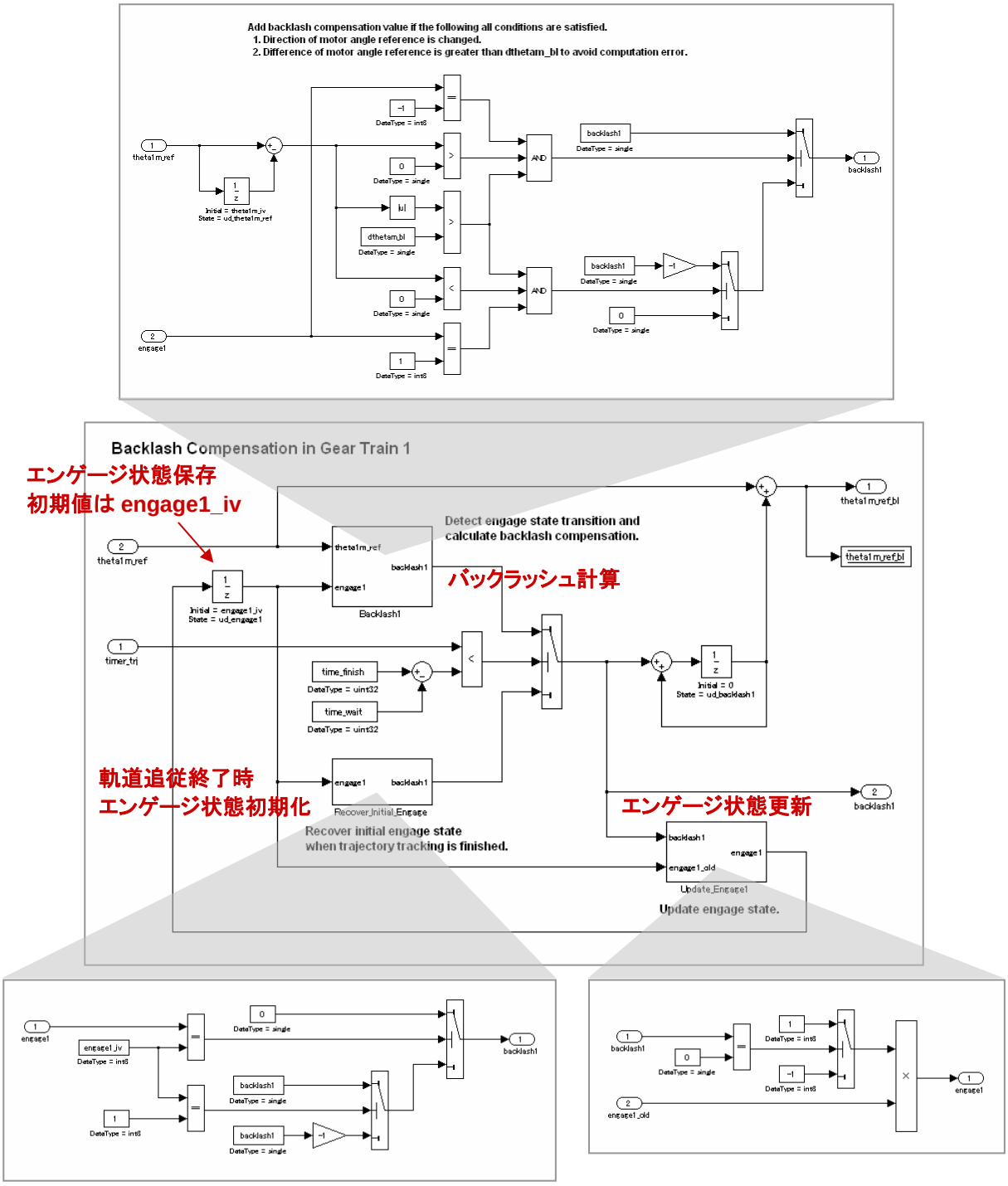


図 8-14　バックラッシュ補正（Compensate_Backlash1 サブシステム）

PWM3 計算（PWM3）

図 8-15 に示すように、Timer サブシステムで求めた `timer_pen` を用いて pwm3 を計算しています。`pen_idx` の偶／奇によってペンの上昇／下降（pwm3 の正／負）を判定しています。

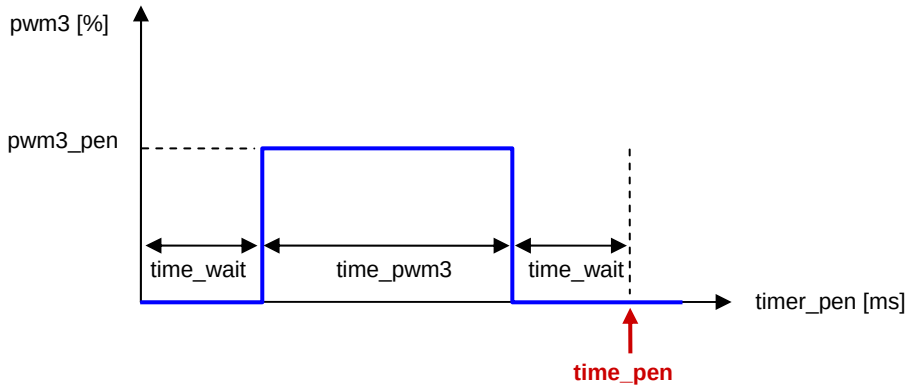


図 8-15　pwm3 の時間変化

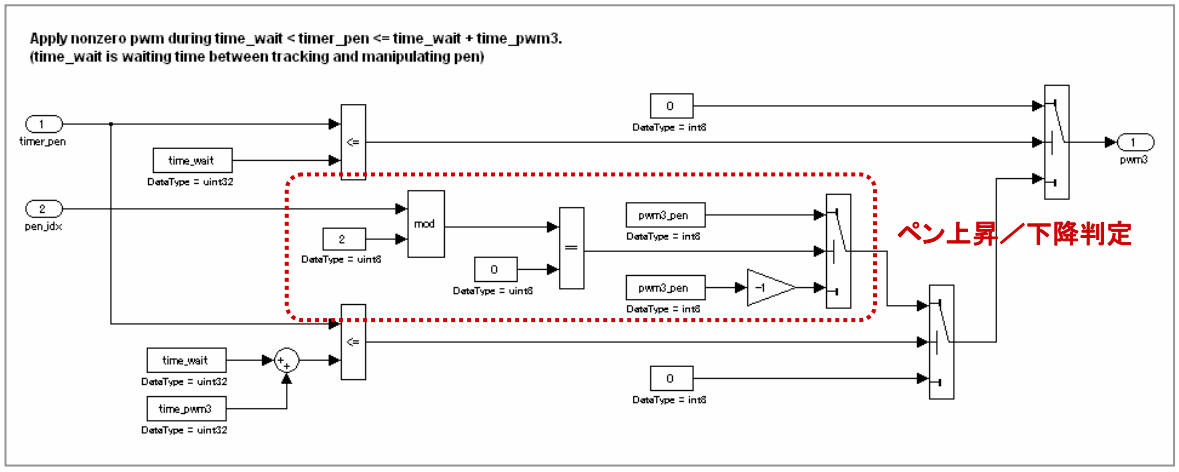


図 8-16　PWM3 計算（PWM3 サブシステム）

調節モード

タッチセンサ 1 を押すと正の PWM、タッチセンサ 2 を押すと負の PWM を出力します。調節対象となるモータは `motor_number` で切り替わります。

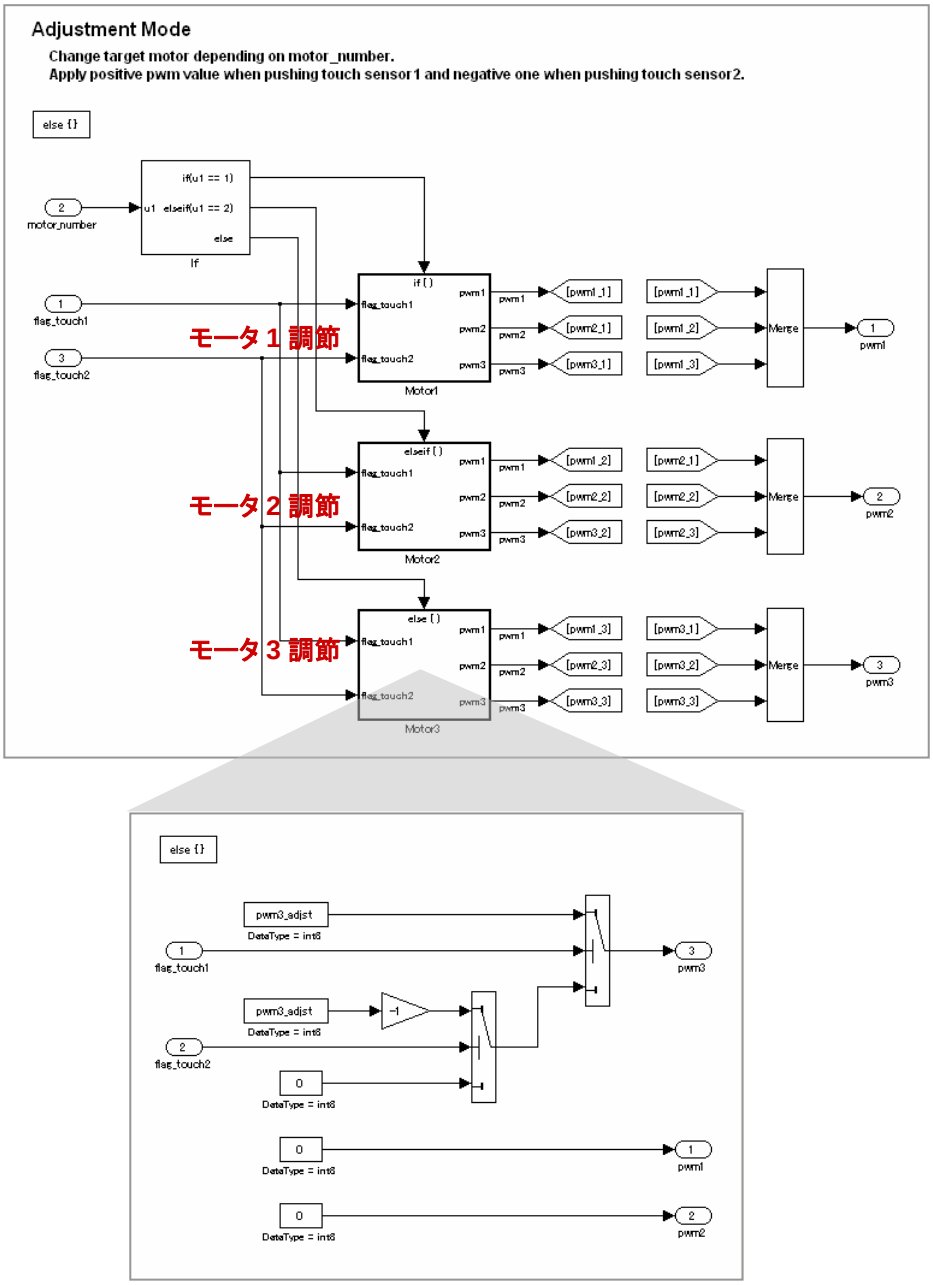


図 8-17　調節モード

8.5　100ms タスク：task_ts2

動作モード変更および調節モード時における対象モータの変更を行うタスクです。`operation_mode` が動作モード、`motor_number` が調節対象モータを表す変数となっています。

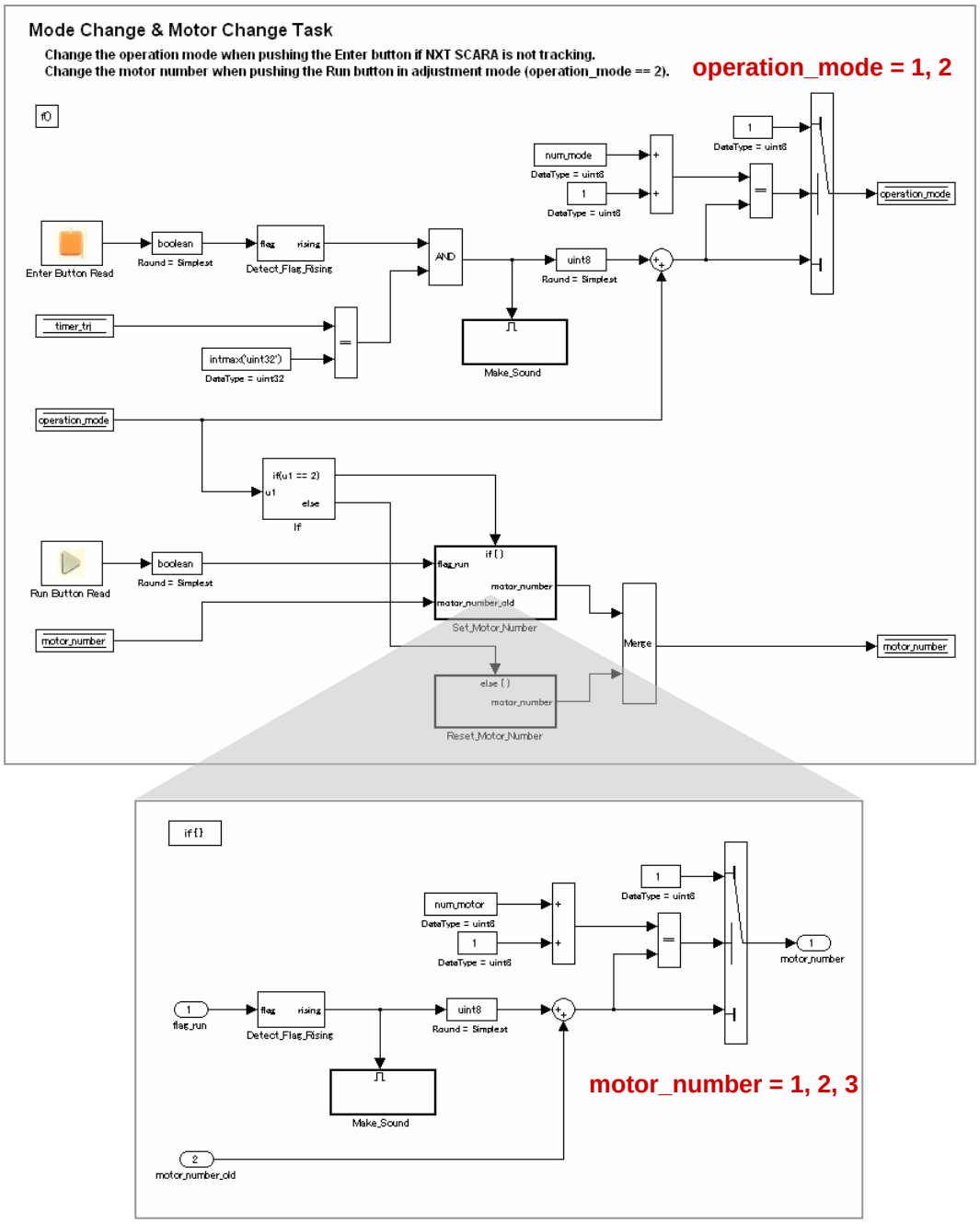


図 8-18　task_ts2 サブシステム

8.6 チューニングパラメータ

軌道追従制御性能を決めるチューニングパラメータは表 8-2 の通りです。LEGO Mindstorms NXT のパーツ・ブロックやセンサ・アクチュエータの個体差の関係で、チューニングパラメータを再調節する必要が生じる可能性がありますのでご注意ください。

表 8-2　チューニングパラメータ

パラメータ	機能
pwm1_gain	DC モータ 1 用 P 制御ゲインテーブル ゲイン
pwm1_offset	DC モータ 1 用 P 制御ゲインテーブル オフセット
pwm2_gain	DC モータ 2 用 P 制御ゲインテーブル ゲイン
pwm2_offset	DC モータ 2 用 P 制御ゲインテーブル オフセット
backlash1	ギヤトレイン 1 バックラッシュ補正量
backlash2	ギヤトレイン 2 バックラッシュ補正量
time_en1	ギヤトレイン 1 バックラッシュ補正時間
time_en2	ギヤトレイン 2 バックラッシュ補正時間

9　シミュレーション

NXT SCARA モデルのシミュレーション方法およびシミュレーション結果について説明します。また、nxtscara_vr.mdl の 3D 表示について説明します。

9.1　シミュレーション方法

シミュレーション方法は通常の Simulink モデルと同じです。 [Trajectory Setting] で目標軌道を選択してから、シミュレーションを実行してください。また、Button and Touch Sensor サブシステム内 Signal Builder ブロックの信号グループを変更することにより、軌道追従モードと調節モードの切り替えを行うことができます。

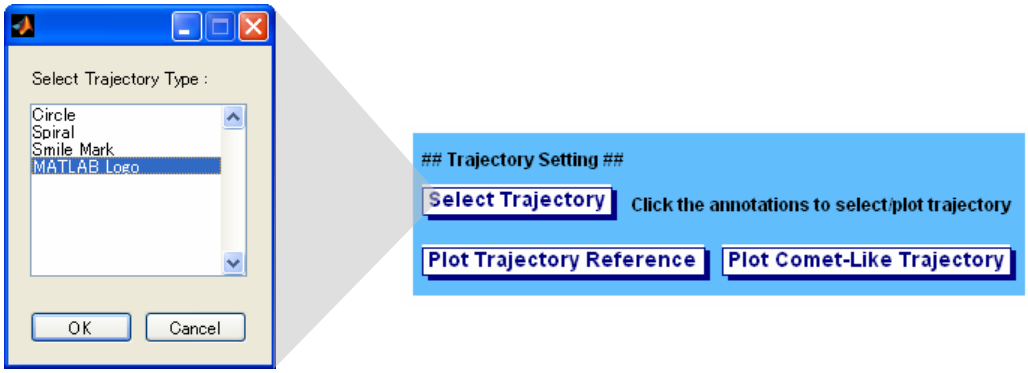


図 9-1　Trajectory Setting

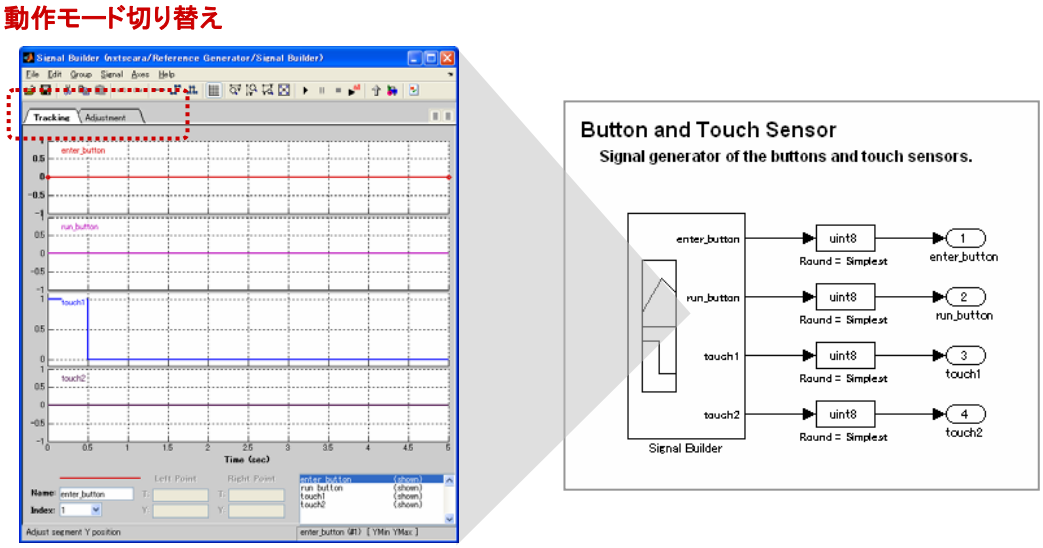
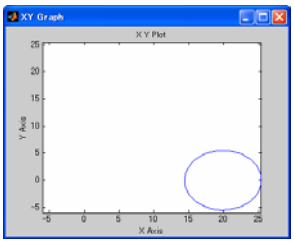


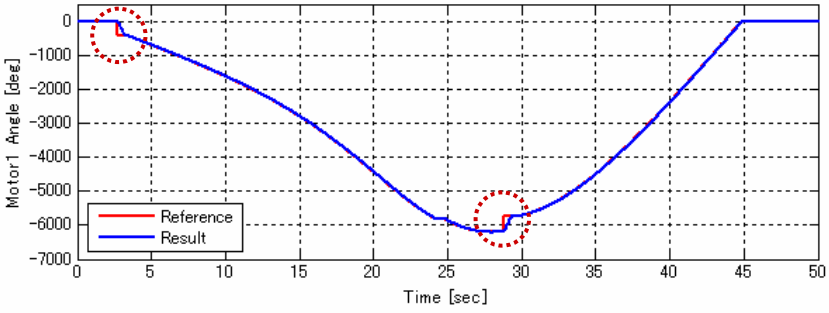
図 9-2　動作モードの切り替え

9.2 シミュレーション結果

図 9-3 は円軌道のシミュレーション結果、図 9-4 は MATLAB ロゴ軌道のシミュレーション結果です。モータ回転角のバックラッシュ補正の効果を確認することができます。



DC モータ 1
バックラッシュ補正



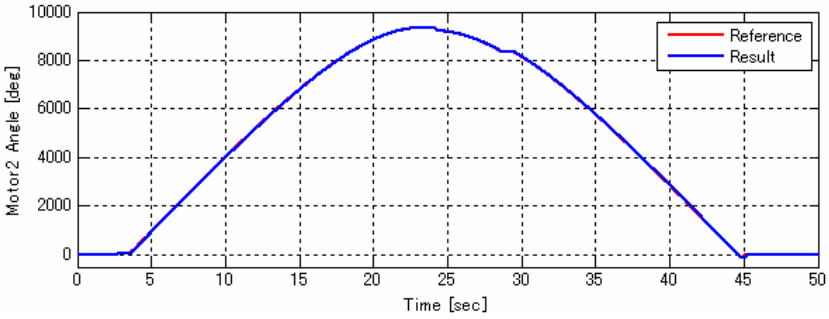
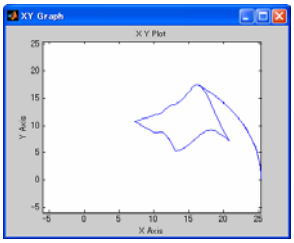
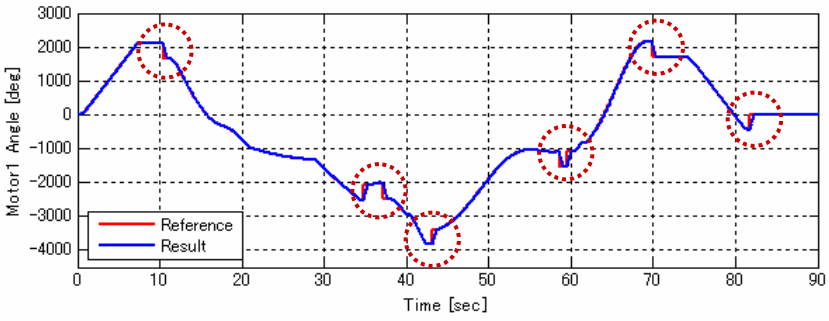


図 9-3　円軌道のシミュレーション結果



DC モータ 1
バックラッシュ補正



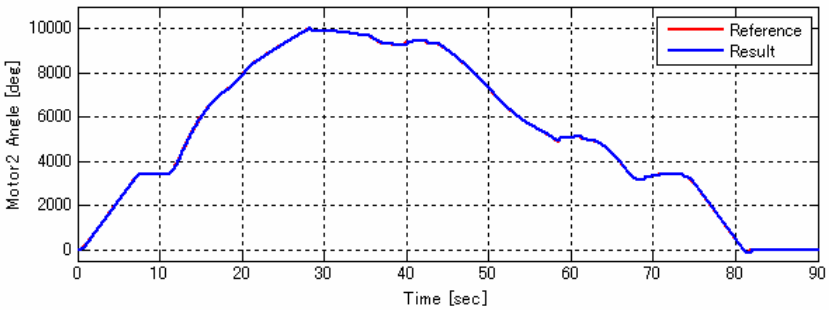


図 9-4　MATLAB ロゴ軌道のシミュレーション結果

下記 URL にシミュレーション動画が公開されています。MATLAB ロゴ描画のシミュレーションの様子をご覧になれます。

http://www.youtube.com/watch?v=kanmZErt4io

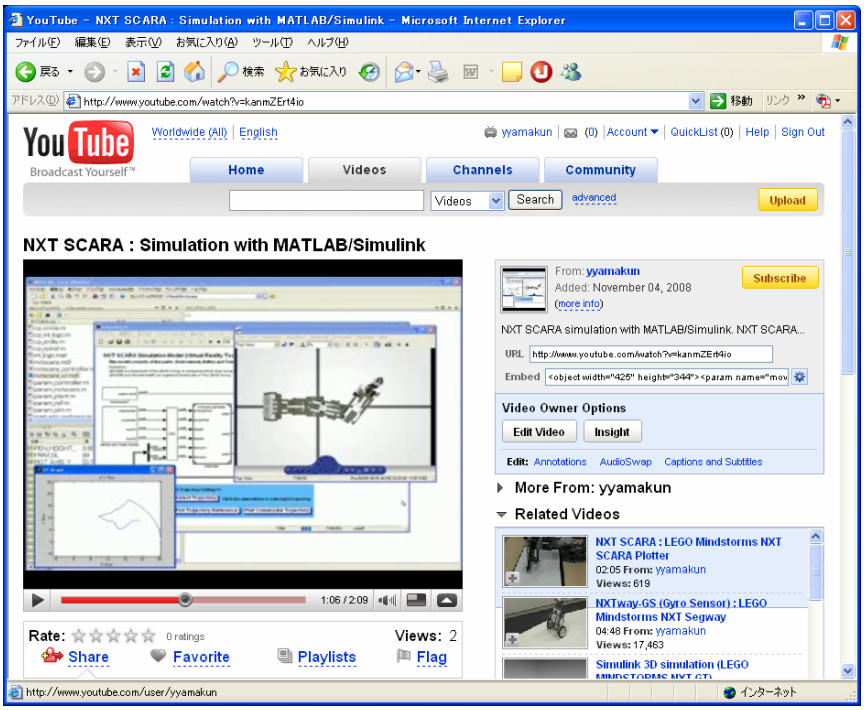


図 9-5　シミュレーション動画

9.3 3D 表示

nxtscara_vr.mdl では、Virtual Reality Toolbox を用いた 3D 表示を行うことができます。Virtual Reality ウィンドウのビューモードを変更することにより、カメラ位置を変更することができます。nxtscara_vr.mdl では次の 4 つのビューモードが用意されています。

- Top View　：　俯瞰視点
- Front View　：　正面カメラ視点
- Back View　：　背面カメラ視点
- Vista View　：　固定カメラ視点

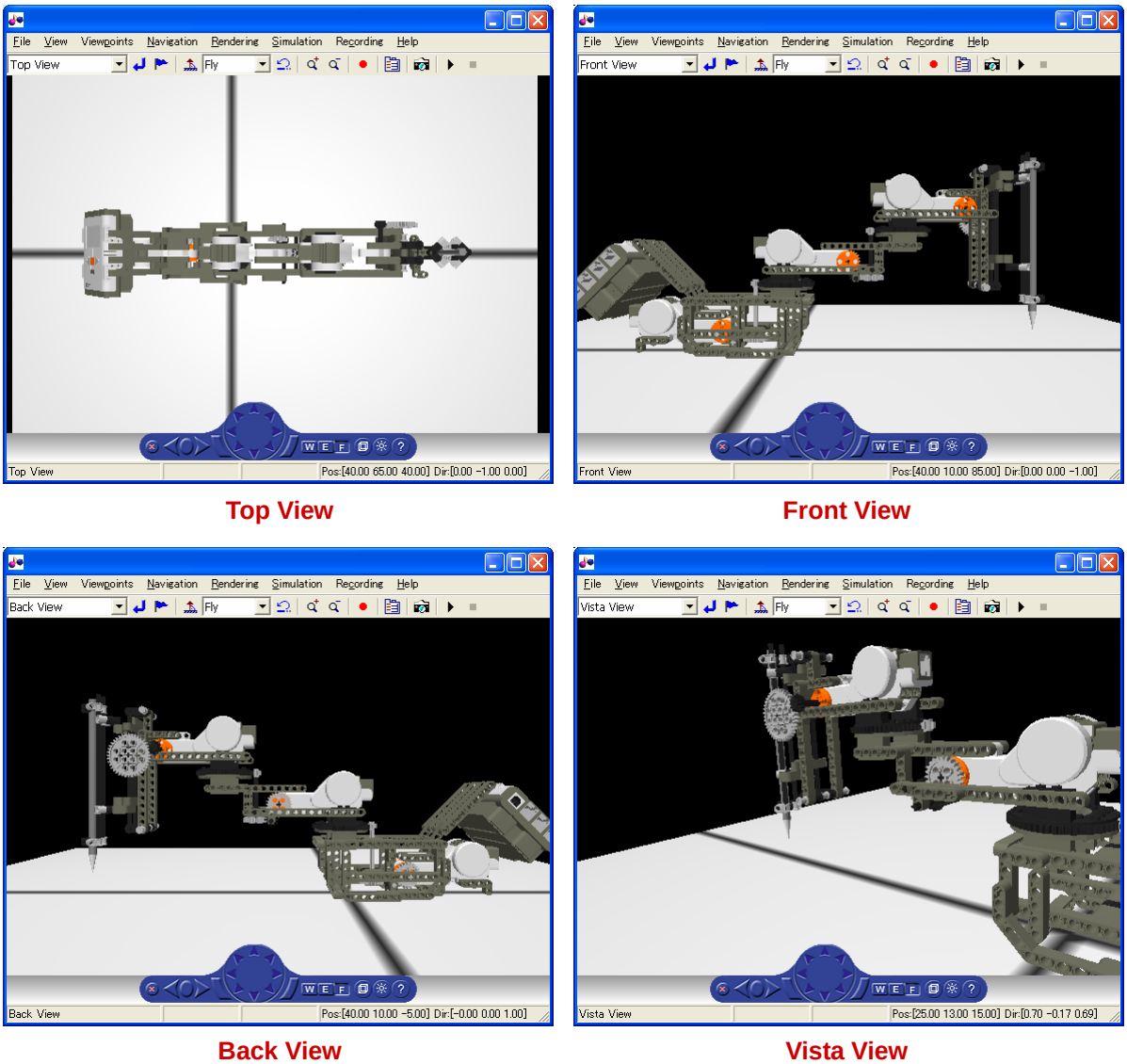


図 9-6　ビューモード

10 コード生成と実装

nxtscara_controller.mdl からのコード生成および生成コードの実装手順について説明します。また、実機を用いた実験結果を紹介します。

10.1 実装環境

表 10-1 は LEGO Mindstorms NXT の実装環境としての特徴および Embedded Coder Robot NXT のソフトウェア仕様をまとめた表です。

表 10-1　LEGO Mindstorms NXT & Embedded Coder Robot NXT 仕様

ハードウェア	マイコン	ATMEL 32-bit ARM 7 (AT91SAM7S256) 48MHz
	フラッシュメモリ	256 K バイト（書き込みは 10000 回まで保証）
	SRAM	64 K バイト
インタフェース	アクチュエータ	DC モータ×3
	センサ	超音波センサ、接触センサ、光センサ、サウンドセンサ
	液晶表示	100×64 ピクセル
	通信	Bluetooth
ソフトウェア	RTOS	LEJOS C / nxtOSEK
	コンパイラ	GCC
	ライブラリ	GCC ライブラリ（C 標準・浮動小数点演算ライブラリ）

RAM サイズまたはフラッシュメモリサイズを超えるプログラムをダウンロードすることはできません（ダウンロードに失敗します)。NXT SCARA に実装する制御プログラムおよび軌道データは、この RAM サイズ／フラッシュメモリサイズの制約を満足しなければならない点にご注意下さい。

10.2 コード生成・実装手順

図 10-1 に示す通り、nxtscara_controller.mdl 内のアノテーションをクリックすることにより、コード生成・ビルド・プログラムダウンロードを行うことができます。その手順は次の通りです。

1. [**SDO Usage**] をクリックしてコード生成時に Simulink データオブジェクトを使用するかどうかを設定します。Simulink データオブジェクトを使用すると、生成コードにユーザ変数情報（変数名、記憶クラス、修飾子等）を反映させることができます。Simulink データオブジェクトの詳細については参考文献[4]を参照してください。
2. [**Generate code and build the generated code**] をクリックしてモデルからコードを生成し、生成コードをビルドします。
3. NXT と PC を USB ケーブルで接続します。NXT のブートモード（NXT 標準ファームウェア／SRAM ブート）に合わせて、[**Download (NXT standard firmware)**] または [**Download (SRAM)**] をクリックして NXT 実機にプログラムをダウンロードします。

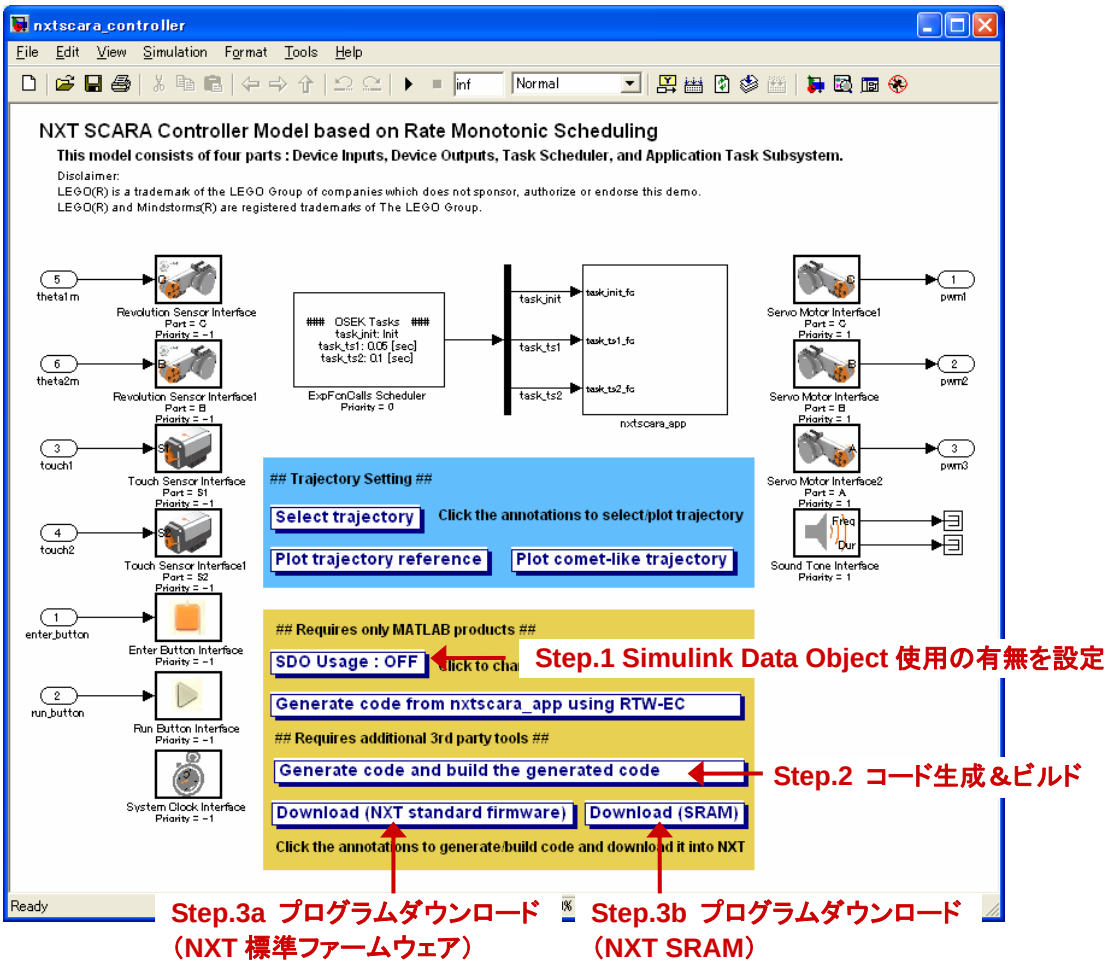


図 10-1　コード生成＆ビルド / プログラムダウンロード用アノテーション

コード生成結果については**付録**を参照してください。

10.3 実験結果

NXT 実機を用いた円軌道および MATLAB ロゴ軌道描画の実験結果は次の通りです。シミュレーションとほぼ同じ結果が得られました。

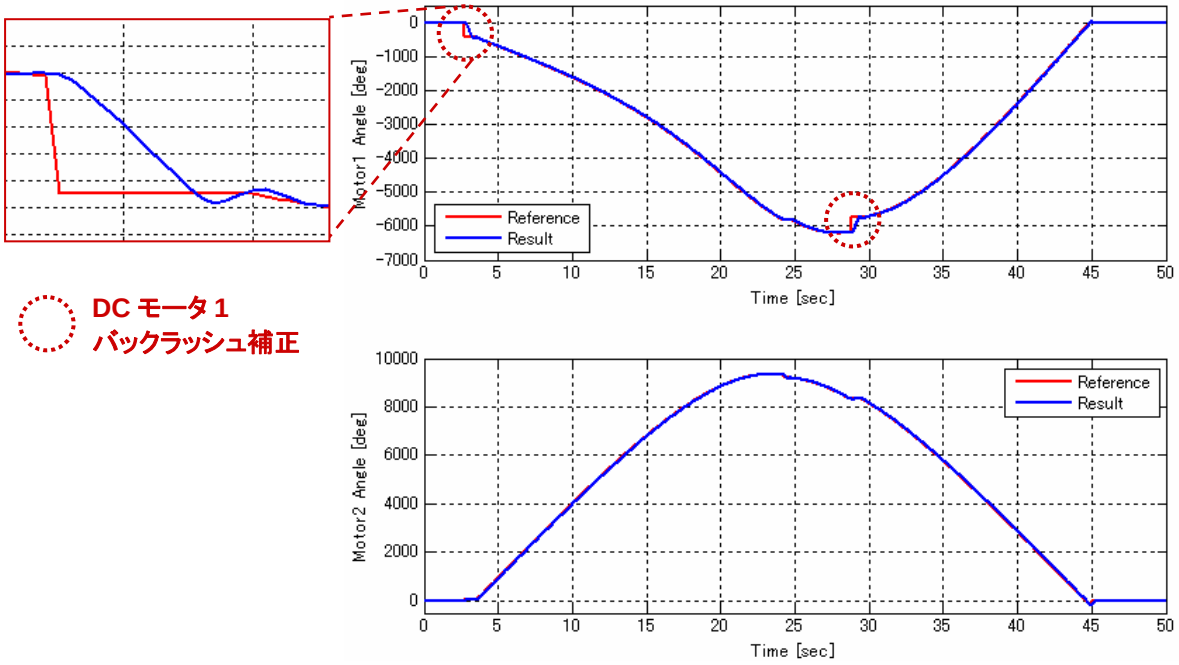


図 10-2　円軌道の実験結果

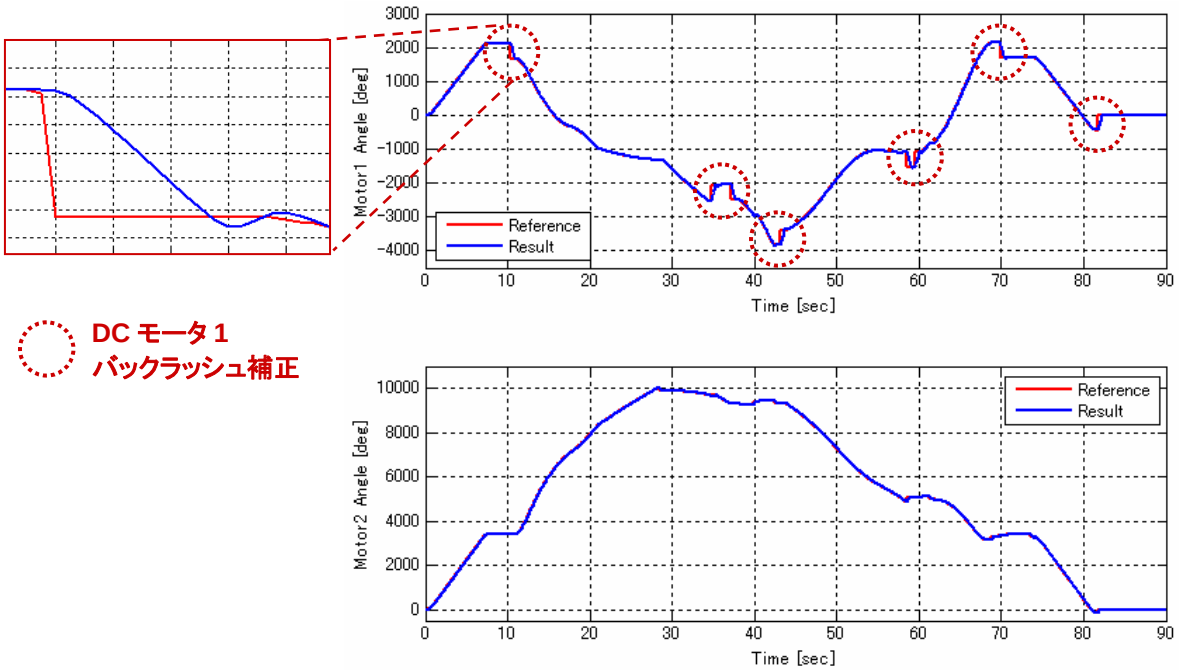


図 10-3　MATLAB ロゴ軌道の実験結果

下記 URL に NXT SCARA の制御実験動画が公開されています。NXT SCARA が MATLAB ロゴ軌道を描画する様子をご覧になれます。

http://www.youtube.com/watch?v=7F2H19teyMY

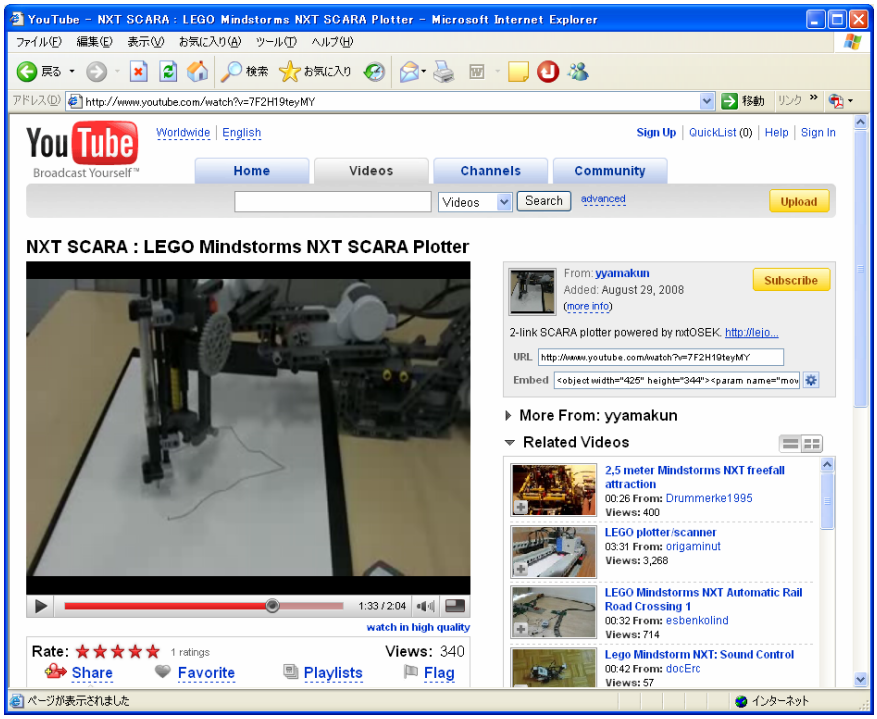


図 10-4　NXT SCARA 制御実験動画

動作前準備作業

NXT SCARA を動作させる前に、以下の手順でリンク角度、バックラッシュ、およびペンの高さの調節を行ってください。

1. プログラムを起動します。
2. Enter ボタンを押して調節モードに入ります。
3. タッチセンサを押してリンク 1 の角度が 0 度になるように調節します。プログラム内のギヤトレイン初期エンゲージ状態は NXT 本体側から見て左側に設定されているため、最後は右回転→左回転となるように調節してください。
4. Run ボタンを押すと、調節対象がリンク 1 からリンク 2 に切り替わります。3.と同じ要領でリンク 2 の角度が 0 度になるように調節してください。
5. Run ボタンを押すと、調節対象がリンク 2 からペンに切り替わります。タッチセンサを押してペンの高さを調節してください。ペンの高さは描画結果に大きな影響を与えますので、適切な高さに調節する必要があります（多少の試行錯誤が必要です）。
6. Run ボタンを押すと、調節対象がペンからリンク 1 に切り替わります。必要があれば、上記の調節作業を繰り返します。
7. 調節作業完了後、プログラムをいったん終了させて再起動します。

11 読者への課題

以下の問題を読者への課題とします。興味のある方はトライして下さい。

- 軌道追従制御器の改善（ゲインチューニング、制御ロジックの改善等）
- 新規軌道の作成
- 2 リンク水平多関節ロボットの運動方程式の導出・シミュレーション
- NXT SCARA 本体の改良

付録　モデル生成コード

nxtscara_controller.mdl から生成される C コードのうち、タスクを実装しているコードを紹介します。スペース節約のためコメントは省略しています。

nxtscara_app.c

```
#include "nxtscara_app.h"
#include "nxtscara_app_private.h"

void task_init(void)
{
  motor_number = 1U;
  operation_mode = 1U;
  timer_trj = MAX_uint32_T;
  pen_idx = 0U;
}

void task_ts1_Start(void)
{
  ud_theta1m_ref = theta1m_iv;
  ud_engage1 = engage1_iv;
  ud_theta2m_ref = theta2m_iv;
  ud_engage2 = engage2_iv;
}

void task_ts1(void)
{
  real32_T rtb_DataTypeConversion3;
  real32_T rtb_DataTypeConversion4;
  real32_T rtb_TrigonometricFunction1;
  real32_T rtb_Divide;
  real32_T rtb_gear_ratio1;
  real32_T rtb_Sum2;
  real32_T rtb_gear_ratio2;
  real32_T rtb_Switch6;
  real32_T rtb_Sum_j;
  real32_T rtb_Sum7;
  real32_T rtb_Switch2_j;
  real32_T rtb_Switch2_l;
  uint32_T rtb_SystemClock_;
  uint32_T rtb_Sum1_h;
  uint32_T rtb_Sum1_k;
  uint32_T rtb_Selector;
  uint32_T rtb_DataStoreRead3_dz;
  uint32_T rtb_Switch1_o;
  int16_T rtb_DataTypeConversion3_e;
  int16_T rtb_DataTypeConversion4_f;
  int16_T rtb_DataTypeConversion2;
  int16_T rtb_DataTypeConversion1;
  int8_T rtb_Switch2_d;
  int8_T rtb_Switch2_o;
  int8_T rtb_Switch2_m;
  int8_T rtb_Product;
  int8_T rtb_Switch3_d;
  uint8_T rtb_DataStoreRead1;
  uint8_T rtb_TouchSensor_S1;
  uint8_T rtb_Switch1_b;
  boolean_T rtb_DataTypeConversion5;
  boolean_T rtb_DataTypeConversion1_i;
  rtb_DataTypeConversion3 = (real32_T)ecrobot_get_motor_rev(NXT_PORT_C);
  rtb_DataTypeConversion4 = (real32_T)ecrobot_get_motor_rev(NXT_PORT_B);
  rtb_SystemClock_ = ecrobot_get_systick_ms();
  rtb_TouchSensor_S1 = ecrobot_get_touch_sensor(NXT_PORT_S1);
```

```
  rtb_DataTypeConversion5 = (ecrobot_get_touch_sensor(NXT_PORT_S1) != 0U);
  rtb_DataTypeConversion1_i = (ecrobot_get_touch_sensor(NXT_PORT_S2) != 0U);
  if (operation_mode == 1U) {
    if (timer_trj > time_finish) {
      if (rtb_DataTypeConversion5) {
        rtb_SystemClock_ = 0U;
      } else {
        rtb_SystemClock_ = MAX_uint32_T;
      }

      timer_trj = rtb_SystemClock_;
      pen_idx = 0U;
      rtb_Switch2_m = 0;
      rtb_Switch2_o = 0;
      rtb_Switch2_d = 0;
    } else {
      rtb_DataStoreRead3_dz = timer_trj;
      rtb_Switch1_o = timer_trj / 50U;

      {
        uint32_T rt_uClip = rtb_Switch1_o;
        rt_uClip = rt_MIN(rt_uClip,804);
        rtb_Sum7 = y_ref[rt_uClip];
      }

      {
        uint32_T rt_uClip = rtb_Switch1_o;
        rt_uClip = rt_MIN(rt_uClip,804);
        rtb_Switch2_j = x_ref[rt_uClip];
      }

      rtb_TrigonometricFunction1 = rt_atan232(rtb_Sum7, rtb_Switch2_j);
      rtb_Switch2_j = rtb_Switch2_j * rtb_Switch2_j + rtb_Sum7 * rtb_Sum7;
      rtb_Sum7 = 1.392399985E-002F;
      rtb_Divide = 4.0F * rtb_Switch2_j * rtb_Sum7;
      rtb_Switch2_j -= 1.849600114E-002F;
      rtb_Sum_j = rtb_Switch2_j + rtb_Sum7;
      rtb_Switch2_l = fabsf(rtb_Divide - rtb_Sum_j * rtb_Sum_j);
      if (rtb_Switch2_l < 0.0F) {
        rtb_Switch2_l = -sqrtf(-rtb_Switch2_l);
      } else {
        rtb_Switch2_l = sqrtf(rtb_Switch2_l);
      }

      rtb_gear_ratio1 = (rtb_TrigonometricFunction1 - rt_atan232(rtb_Switch2_l,
        rtb_Sum_j)) * 5.729578018E+001F * 84.0F;
      rtb_Sum2 = rtb_gear_ratio1 + ud_backlash1;
      theta1m_ref_bl = rtb_Sum2;
      if (rtb_DataStoreRead3_dz < (uint32_T)((int32_T)time_finish - (int32_T)
          time_wait)) {
        rtb_Sum_j = rtb_gear_ratio1 - ud_theta1m_ref;
        rtb_DataTypeConversion5 = (fabsf(rtb_Sum_j) > dthetam_bl);
        if ((ud_engage1 == -1) && (rtb_Sum_j > 0.0F) && rtb_DataTypeConversion5)
        {
          rtb_Sum_j = backlash1;
        } else {
          if (rtb_DataTypeConversion5 && (rtb_Sum_j < 0.0F) && (ud_engage1 == 1))
          {
            rtb_Sum_j = -backlash1;
          } else {
            rtb_Sum_j = 0.0F;
          }
        }
      } else {
        if (ud_engage1 == engage1_iv) {
          rtb_Sum_j = 0.0F;
        } else {
          if (engage1_iv == 1) {
            rtb_Sum_j = backlash1;
```

```
        } else {
          rtb_Sum_j = -backlash1;
        }
      }
    }

    if (rtb_Sum_j == 0.0F) {
      rtb_Switch3_d = 1;
    } else {
      rtb_Switch3_d = -1;
    }

    rtb_Product = (int8_T)(rtb_Switch3_d * ud_engage1);
    rtb_Divide = rtb_Sum7 * 1.849600114E-002F * 4.0F;
    rtb_Sum7 = rtb_Switch2_j - rtb_Sum7;
    rtb_Switch2_l = fabsf(rtb_Divide - rtb_Sum7 * rtb_Sum7);
    if (rtb_Switch2_l < 0.0F) {
      rtb_Switch2_l = -sqrtf(-rtb_Switch2_l);
    } else {
      rtb_Switch2_l = sqrtf(rtb_Switch2_l);
    }

    rtb_gear_ratio2 = 5.729578018E+001F * rt_atan232(rtb_Switch2_l, rtb_Sum7) *
      84.0F;
    rtb_TrigonometricFunction1 = rtb_gear_ratio2 + ud_backlash2;
    theta2m_ref_bl = rtb_TrigonometricFunction1;
    if (rtb_DataStoreRead3_dz < (uint32_T)((int32_T)time_finish - (int32_T)
        time_wait)) {
      rtb_Divide = rtb_gear_ratio2 - ud_theta2m_ref;
      rtb_DataTypeConversion5 = (fabsf(rtb_Divide) > dthetam_bl);
      if ((ud_engage2 == -1) && (rtb_Divide > 0.0F) && rtb_DataTypeConversion5)
      {
        rtb_Switch6 = backlash2;
      } else {
        if (rtb_DataTypeConversion5 && (rtb_Divide < 0.0F) && (ud_engage2 == 1))
        {
          rtb_Switch6 = -backlash2;
        } else {
          rtb_Switch6 = 0.0F;
        }
      }
    } else {
      if (ud_engage2 == engage2_iv) {
        rtb_Switch6 = 0.0F;
      } else {
        if (engage2_iv == 1) {
          rtb_Switch6 = backlash2;
        } else {
          rtb_Switch6 = -backlash2;
        }
      }
    }

    if (rtb_Switch6 == 0.0F) {
      rtb_Switch3_d = 1;
    } else {
      rtb_Switch3_d = -1;
    }

    if ((rtb_DataTypeConversion3 < theta1m_max) && (rtb_DataTypeConversion3 >
        -theta1m_max)) {
      rtb_Divide = (rtb_Sum2 - rtb_DataTypeConversion3) / 50.0F;
      rtb_Divide = pwm1_offset * rt_FSGN(rtb_Divide) + pwm1_gain * rtb_Divide;
      rtb_Switch2_l = rt_SATURATE(rtb_Divide, -100.0F, 100.0F);
    } else {
      rtb_Switch2_l = -100.0F * rt_FSGN(rtb_DataTypeConversion3);
    }

    if ((rtb_DataTypeConversion4 < theta2m_max) && (rtb_DataTypeConversion4 >
```

```
        -theta2m_max)) {
      rtb_Divide = (rtb_TrigonometricFunction1 - rtb_DataTypeConversion4) /
        50.0F;
      rtb_Divide = pwm2_offset * rt_FSGN(rtb_Divide) + pwm2_gain * rtb_Divide;
      rtb_Switch2_j = rt_SATURATE(rtb_Divide, -100.0F, 100.0F);
    } else {
      rtb_Switch2_j = -100.0F * rt_FSGN(rtb_DataTypeConversion4);
    }

    rtb_Sum1_h = rtb_SystemClock_ + time_en1;
    if (rtb_Sum_j != 0.0F) {
      timer_en1_0 = rtb_Sum1_h;
    }

    rtb_Sum1_k = rtb_SystemClock_ + time_en2;
    if (rtb_Switch6 != 0.0F) {
      timer_en2_0 = rtb_Sum1_k;
    }

    rtb_SystemClock_ += time_pen;
    rtb_DataStoreRead1 = pen_idx;
    if (pen_idx <= 1U) {
      rtb_Switch1_o = time_table_pen[(int32_T)pen_idx];
    } else {
      rtb_Switch1_o = MAX_uint32_T;
    }

    rtb_DataTypeConversion5 = (rtb_Switch1_o == rtb_DataStoreRead3_dz);
    if (rtb_DataTypeConversion5 && (!ud_flag_pen)) {
      timer_pen_0 = rtb_SystemClock_;
    }

    rtb_Selector = (uint32_T)((int32_T)rtb_SystemClock_ - (int32_T)timer_pen_0);
    if (((uint32_T)((int32_T)rtb_Sum1_h - (int32_T)timer_en1_0) >= time_en1) &&
        ((uint32_T)((int32_T)rtb_Sum1_k - (int32_T)timer_en2_0) >= time_en2) &&
        (rtb_Selector >= time_pen)) {
      rtb_SystemClock_ = 50U;
    } else {
      rtb_SystemClock_ = 0U;
    }

    timer_trj = rtb_SystemClock_ + rtb_DataStoreRead3_dz;
    if (rtb_DataStoreRead3_dz == rtb_Switch1_o) {
      rtb_Switch1_b = 1U;
    } else {
      rtb_Switch1_b = 0U;
    }

    pen_idx = (uint8_T)((uint32_T)rtb_Switch1_b + (uint32_T)pen_idx);
    rtb_Switch2_m = (int8_T)floor((real_T)rtb_Switch2_l + 0.5);
    rtb_Switch2_o = (int8_T)floor((real_T)rtb_Switch2_j + 0.5);
    if (rtb_Selector <= time_wait) {
      rtb_Switch2_d = 0;
    } else {
      if (rtb_Selector <= time_wait + time_pwm3) {
        rtb_DataStoreRead1 %= 2U;
        if (rtb_DataStoreRead1 == 0U) {
          rtb_Switch2_d = pwm3_pen;
        } else {
          rtb_Switch2_d = (int8_T)(-pwm3_pen);
        }
      } else {
        rtb_Switch2_d = 0;
      }
    }

    ud_backlash1 = rtb_Sum_j + ud_backlash1;
    ud_theta1m_ref = rtb_gear_ratio1;
    ud_engage1 = rtb_Product;
```

```
      ud_backlash2 = rtb_Switch6 + ud_backlash2;
      ud_theta2m_ref = rtb_gear_ratio2;
      ud_engage2 = (int8_T)(rtb_Switch3_d * ud_engage2);
      ud_flag_pen = rtb_DataTypeConversion5;
    }
  } else {
    if (motor_number == 1U) {
      if (rtb_DataTypeConversion5) {
        rtb_Switch2_m = pwm1_adjst;
      } else {
        if (rtb_DataTypeConversion1_i) {
          rtb_Switch2_m = (int8_T)(-pwm1_adjst);
        } else {
          rtb_Switch2_m = 0;
        }
      }

      rtb_Switch2_o = 0;
      rtb_Switch2_d = 0;
    } else if (motor_number == 2U) {
      rtb_Switch2_m = 0;
      if (rtb_DataTypeConversion5) {
        rtb_Switch2_o = pwm2_adjst;
      } else {
        if (rtb_DataTypeConversion1_i) {
          rtb_Switch2_o = (int8_T)(-pwm2_adjst);
        } else {
          rtb_Switch2_o = 0;
        }
      }

      rtb_Switch2_d = 0;
    } else {
      rtb_Switch2_m = 0;
      rtb_Switch2_o = 0;
      if (rtb_DataTypeConversion5) {
        rtb_Switch2_d = pwm3_adjst;
      } else {
        if (rtb_DataTypeConversion1_i) {
          rtb_Switch2_d = (int8_T)(-pwm3_adjst);
        } else {
          rtb_Switch2_d = 0;
        }
      }
    }
  }

  ecrobot_set_motor_mode_speed(NXT_PORT_C, 1, rtb_Switch2_m);
  ecrobot_set_motor_mode_speed(NXT_PORT_B, 1, rtb_Switch2_o);
  ecrobot_set_motor_mode_speed(NXT_PORT_A, 1, rtb_Switch2_d);
  rtb_DataTypeConversion3_e = (int16_T)rtb_TouchSensor_S1;
  rtb_DataTypeConversion4_f = (int16_T)rt_MIN(timer_trj, 32767U);
  rtb_DataTypeConversion2 = (int16_T)floor((real_T)theta1m_ref_bl + 0.5);
  rtb_DataTypeConversion1 = (int16_T)floor((real_T)theta2m_ref_bl + 0.5);
  ecrobot_bt_adc_data_logger(0, 0, rtb_DataTypeConversion3_e,
    rtb_DataTypeConversion4_f, rtb_DataTypeConversion2, rtb_DataTypeConversion1);
}

void task_ts2(void)
{
  uint8_T rtb_DataStoreRead2_p;
  uint8_T rtb_RunButton_;
  boolean_T rtb_DataTypeConversion2_g;
  boolean_T rtb_LogicalOperator2_a;
  boolean_T rtb_DataTypeConversion1_e;
  boolean_T rtb_LogicalOperator2_e;
  rtb_DataTypeConversion2_g = (ecrobot_is_ENTER_button_pressed() != 0U);
  rtb_LogicalOperator2_a = (rtb_DataTypeConversion2_g && (!ud_flag_enter) &&
    (timer_trj == MAX_uint32_T));
```

```
  rtb_DataStoreRead2_p = operation_mode;
  rtb_RunButton_ = (uint8_T)((uint32_T)rtb_LogicalOperator2_a + (uint32_T)
    operation_mode);
  if ((uint8_T)((uint32_T)num_mode + 1U) == rtb_RunButton_) {
    rtb_RunButton_ = 1U;
  }

  operation_mode = rtb_RunButton_;
  rtb_DataTypeConversion1_e = (ecrobot_is_RUN_button_pressed() != 0U);
  rtb_RunButton_ = motor_number;
  if (rtb_DataStoreRead2_p == 2U) {
    rtb_LogicalOperator2_e = (rtb_DataTypeConversion1_e && (!ud_flag_run));
    if (rtb_LogicalOperator2_e) {
      ecrobot_sound_tone(880U, 200U, 70);
    }

    rtb_RunButton_ = (uint8_T)((uint32_T)rtb_LogicalOperator2_e + (uint32_T)
      rtb_RunButton_);
    if ((uint8_T)((uint32_T)num_motor + 1U) == rtb_RunButton_) {
      rtb_RunButton_ = 1U;
    }

    ud_flag_run = rtb_DataTypeConversion1_e;
  } else {
    rtb_RunButton_ = 1U;
  }

  motor_number = rtb_RunButton_;
  if (rtb_LogicalOperator2_a) {
    ecrobot_sound_tone(440U, 600U, 70);
  }

  ud_flag_enter = rtb_DataTypeConversion2_g;
}

void nxtscara_app_initialize(void)
{
  task_ts1_Start();
}
```

参考文献

[1] Philo’s Home Page　LEGO Mindstorms NXT
http://www.philohome.com/

[2] 島田明、大石潔、柴田昌明、市川修　EE テキスト モーションコントロール　オーム社　2004

[3] 2 リンクロボットマニピュレータのDCモータによる位置制御へのモデルベース開発の適用
－機構系と電気系のマルチドメインモデリングとPID 制御パラメータの最適チューニング－
http://www.cybernet.co.jp/matlab/library/library/detail.php?id=TA040

[4] RTW-ECを用いた組み込み制御プログラム開発 チュートリアル
http://www.cybernet.co.jp/matlab/library/library/detail.php?id=TT036

MATLAB ヘルプ・情報リソース

■ 関数・コマンドの使用方法を調べる

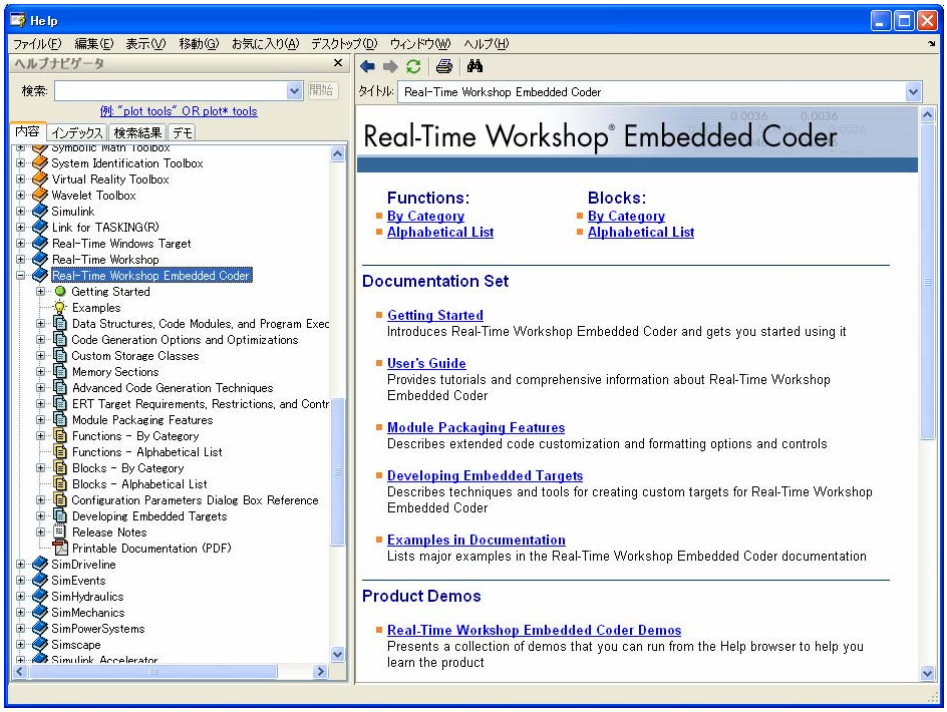
```
>> help 関数・コマンド名
       or
>> doc 関数・コマンド名
```

■ 関数・コマンドの一覧リストを表示する

```
>> helpwin
```

■ MATLABヘルプブラウザ

MATLABの［ ヘルプ ］メニュー→［ MATLABヘルプ ］を選択するか、 アイコンをクリックすることによりヘルプブラウザを開くことができます。ヘルプブラウザからはMATLABマニュアルやデモを参照することができます。



■ 弊社MATLABホームページ

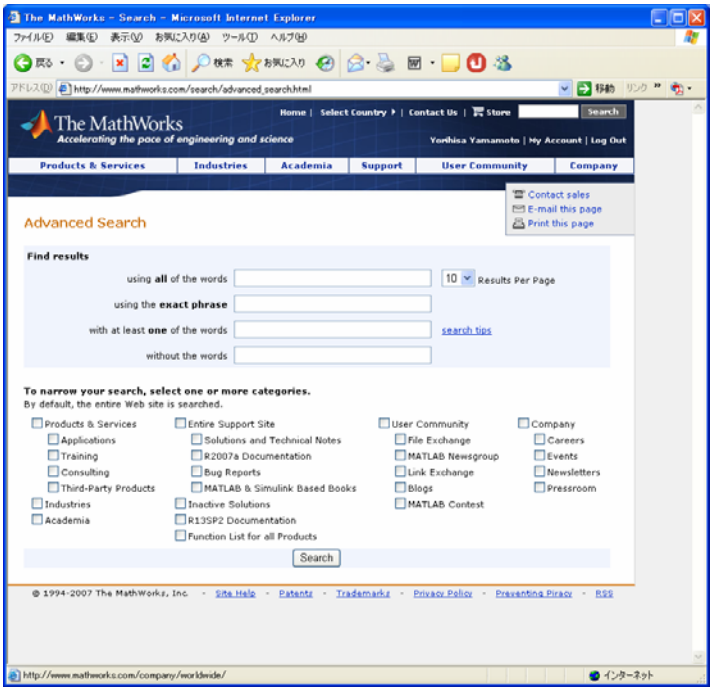
http://www.cybernet.co.jp/matlab/

製品情報・日本語ドキュメント・FAQ・技術資料等を掲載しています。

■ 開発元ソリューション検索ページ

http://www.mathworks.com/search/advanced_search.html

MATLABに関する最新の情報を検索することができます。



■ 著作権

本資料の一部あるいは全部を無断で転載、複製、複写されると著作権等の権利侵害となる場合がありますのでご注意ください。

■ 商標の帰属

MATLAB®は米国The MathWorks,Inc.の登録商標です。また、LEGO®はレゴ社の登録商標です。
その他、本資料に記載されている製品名とブランド名は、それぞれ該当する各社の商標または登録商標です。

■ 免責事項

本資料に掲載されている操作の影響につきましては、弊社では責任を負いかねますので予めご了承ください。

■ 問合せ先

本資料の内容についてお気づきの点がありましたら、弊社までご連絡頂ければ幸いです。

NXT SCARA のモデルベース開発
～LEGO Mindstorms NXT を用いた 2 リンク水平多関節ロボットの制御～
2008 年 9 月作成

CYBERNET
サイバネットシステム株式会社

応用システム第 1 事業部 http://www.cybernet.co.jp/MATLAB
応用システム第 1 事業部 営業部 E-mail: infomatlab@cybernet.co.jp
応用システム第 1 事業部 技術部 E-mail: techmatlab@cybernet.co.jp

本　　社	〒101-0022	東京都千代田区神田練塀町 3 富士ソフトビル14 階	Tel:03-5297-3565(営業)	03-5297-3546(技術)	Fax:03-5297-3648
中 部 支 社	〒460-0003	愛知県古屋市中区錦 1-6-26 富士ソフトビル 3 階	Tel:052-219-5197(営業)	052-219-5198(技術)	Fax:052-219-5970
西日本支社	〒542-0028	大阪市中央区常盤町 1-3-8 中央大通 FN ビル	Tel:06-6940-3613(営業)	06-6940-3611(技術)	Fax:06-6940-3602