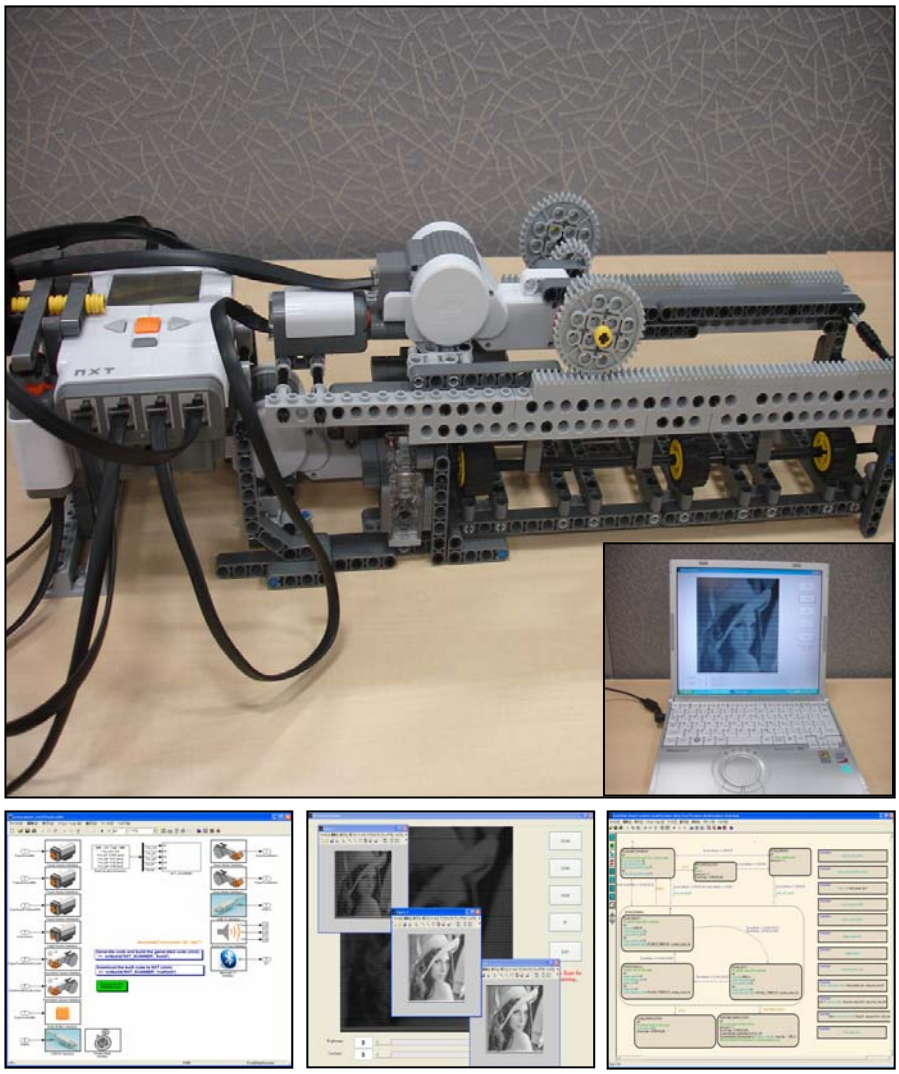
NXT Scanner のモデルベース開発

～LEGO Mindstorms NXT を用いた Image Scanner～



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
坂野　好陽　　E-mail : banno@cybernet.co.jp
福田　智樹　　E-mail : t_fukuda@cybernet.co.jp

■ **改訂履歴**

バージョン	年月	改訂内容	著者・編者
1.0	2009/02	初版	坂野好陽 banno@cybernet.co.jp
		初版（6.2 章、6.4 章、11.1 章、11.2 章、11.3 章）	福田智樹 t_fukuda@cybernet.co.jp
2.0	2009/04	・　バージョン 1.0 の日付誤記修正 ・　その他誤記修正	坂野好陽 banno@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 Scanner はLEGO Mindstorms NXT を用いて作成したシートフィードタイプのイメージスキャナです。NXT Viewer は、MATLAB の M スクリプトを用いた画像表示、補正プログラムです。本資料は MATLAB / Simulink を用いた NXT Scanner の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。また、新機能や、様々なオプション製品の具体的な使用方法についても説明しています。本資料の主な内容は次の通りです。

- NXT Scanner と NXT Viewer の概要
- NXT Scanner の機構
- NXT Scanner の設計
- NXT Scanner のモデリング
- シミュレーション結果および実験結果
- NXT Viewer による画像取り込みと画像補正処理

前準備

NXT Scannerの組み立て方法についてはNXT Scanner **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参考文献[1]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

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	3.16+
nxtOSEK/JSP	2.06
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®	7.7	R2008b
Simulink®	7.2	R2008b
Stateflow®	7.2	R2008b
Real-Time Workshop®	7.2	R2008b
Real-Time Workshop® Embedded Coder™	5.2	R2008b
Stateflow® Coder™	7.2	R2008b
Image Processing Toolbox™	6.2	R2008b
PolySpace® Server™ for C/C++	6.0	R2008b
PolySpace® Client™ for C/C++	6.0	R2008b
PolySpace® Model Link™ SL	5.2	R2008b
Simulink® Verification and Validation™	2.4	R2008b
Simulink® Design Verifier™	1.3	R2008b

ファイルリスト

分類	ファイル名	内容
共通用	setup_nxtscanner.m	PATH 設定ファイル
	nxtscanner_ctrl.mdl	NXT Scanner コントローラモデル
	nxtscanner_lib.mdl	NXT Scanner ライブラリモデル
	mode_control_verification_lib.mdl	サンプル： mode_control.mdl 検証仕様モデル
Data Object	nxtscanner.xls	Simulink Data Object 管理データ
	SDOxlsIF フォルダ以下全て	Excel Interface API for Simulink Data Object Tool
モード制御	mode_control_prm.m	パラメータ定義用 M-スクリプト
	mode_control.mdl	NXT Scanner モード制御部機能モデル
紙送り制御	paper_feed_mechanism.mdl	NXT Scanner 紙送り機構制御部機能モデル
スキャン制御	scanner_head_prm.m	パラメータ定義用 M-スクリプト
	scanner_head.mdl	NXT Scanner スキャナヘッド制御部機能モデル
USB 通信制御	data_communication.mdl	NXT Scanner USB データ通信部機能モデル
	usb_test_signal.mdl	NXT Scanner USB データ通信部テストモデル
	usb_test_data.mat	data_communication.mdl の検証用データ
	usb_test_number.mat	data_communication.mdl の検証用データ
	usb_test_state.mat	data_communication.mdl の検証用データ
列挙型定義	ControlModeEnum.m	列挙型定義 M ファイル
	ErrorCodeEnum.m	列挙型定義 M ファイル
	FeedModeEnum.m	列挙型定義 M ファイル
	ScanModeEnum.m	列挙型定義 M ファイル
	ScannerBufferInfoEnum.m	列挙型定義 M ファイル
NXT Viewer	NXTScannerViewer.fig	NXT Viewer の GUI 定義ファイル
	NXTScannerViewer.m	NXT Viewer の M-スクリプト
PolySpace 用	nxtscanner_ctrl_polyspace.cfg	PolySpace コンフィグファイル
	polyspace_main.c	検証用タスク構成定義ファイル
	polyspace.h	検証用コンパイルエラー回避設定ファイル
	polyspace_additional_file_list.txt	PolySpace コンフィグ補助ファイル

目次

はじめに i
前準備 i
使用MATLAB製品 ii
ファイルリスト iii
1 制御系モデルベース開発 1
1.1 モデルベース開発とは 1
1.2 モデルベース開発プロセス 2
1.3 モデルベース開発のメリット 3
2 NXT Scanner、NXT Viewerの要求分析 4
2.1 要求分析 4
3 NXT Scannerの機構 6
3.1 構造 6
3.2 バックラッシュ 11
3.3 センサ・アクチュエータ 11
4 NXT Scannerのシステム設計 12
4.1 システム概要 12
4.2 モード制御システム 13
4.3 紙送り制御システム 15
4.4 スキャン制御システム 16
4.5 USB通信制御システム 18
5 NXT Scannerの構成設計 19
5.1 ライブラリモデル 19
5.2 システム全体の共有データ 19
6 NXT Scannerの詳細設計 22
6.1 enumerated typeの表現（R2008b新機能） 22
6.2 Simulink Functionの活用（R2008b新機能） 24
6.3 モード制御モデル 25
6.4 紙送り制御モデル 28
6.5 スキャン制御モデル 39
6.6 USB通信制御モデル 46
7 機能モデルのシミュレーション 49
7.1 シミュレーション方法 49
7.2 検証ツールの機能紹介 50
8 NXT Scanner統合モデル 53
8.1 制御プログラム概要 53
8.2 モデル概要 54
8.3 初期化タスク：task_init 57

8.4 2msタスク：task_ts1 ... 57
8.5 10msタスク：task_ts2 ... 58
8.6 20msタスク：task_ts3 ... 58
8.7 60msタスク：task_ts4 ... 59
8.8 チューニングパラメータ ... 60
9 コード生成と実装 ... 61
9.1 実装環境 ... 61
9.2 コード生成・実装手順 ... 62
10 生成コードの検証 ... 63
10.1 PolySpaceについて ... 63
10.2 PolySpaceの設定 ... 64
10.3 PolySpaceのランタイムエラー検出 ... 66
10.4 PolySpace検証結果 ... 68
11 NXT Viewerについて ... 69
11.1 NXT Viewerの使い方 ... 69
11.2 USB通信用コマンド（nxtusb） ... 71
11.3 画像表示の概要 ... 72
11.4 画像補正の概要 ... 73
12 実験結果 ... 76
13 読者への課題 ... 77
付録 モデル生成コード ... 78
参考文献 ... 95
MATLABヘルプ・情報リソース ... 96

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。

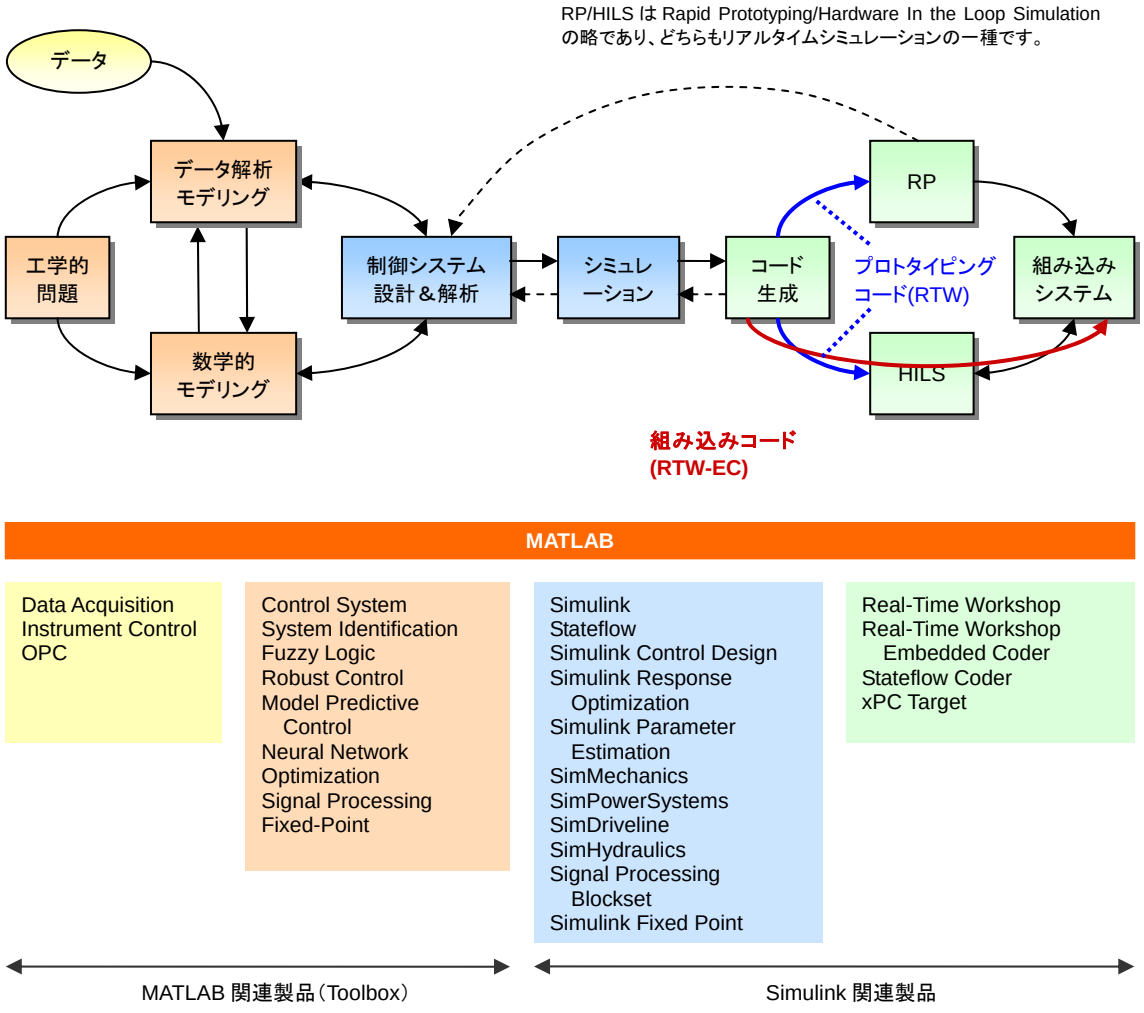


図 1-1　MATLAB製品を用いた制御系MBD

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。

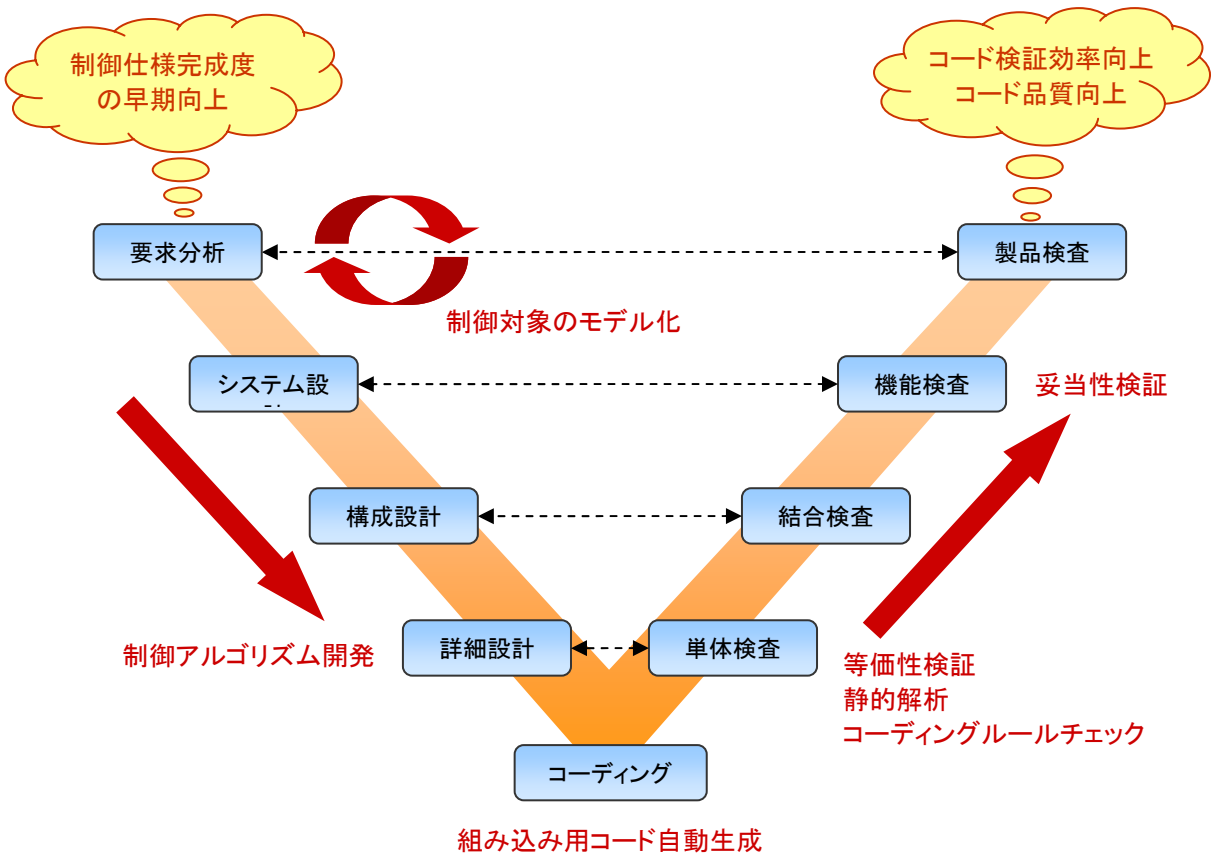


図 1-2　制御系MBDのVプロセス

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 Scanner、NXT Viewer の要求分析

NXT Scanner と NXT Viewer の要求分析について説明します。

2.1　要求分析

NXT Scanner と NXT Viewer は、図 2-1 で表すユースケース図の要求を満たします。各ユースケースの詳細を、表 2-1 から表 2-4 にてユースケース記述しています。

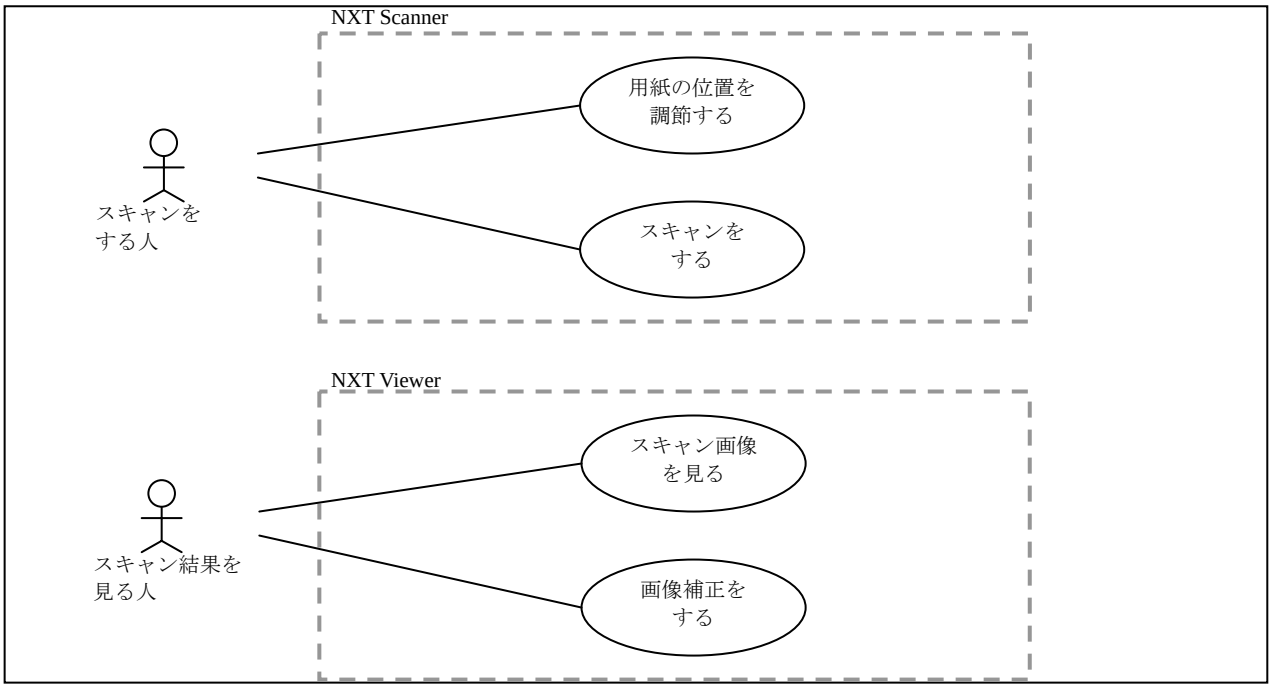


図 2-1　本質的ユースケース図

表 2-1　ユースケース記述（用紙の位置を調整する）

ユースケース名	NXT Scanner 用紙の位置を調整する
正常フロー	1. ユーザは、「紙送り」もしくは「紙戻し」を要求する 2. NXT Scanner は、要求に従い用紙を送る 3. ユーザは、「紙送り」もしくは「紙戻し」の停止を要求する 4. NXT Scanner は、要求に従い用紙送りをやめる
異常フロー	異常が発生した場合はすぐに処理を中断する

表 2-2　ユースケース記述（スキャンする）

ユースケース名	NXT Scanner スキャンをする
正常フロー	1. ユーザは、「スキャン開始」を要求する 2. NXT Scanner は、要求に従いスキャンを開始する 3. NXT Scanner は、スキャナヘッドを水平移動させながらスキャンデータを取得する 4. NXT Scanner は、取得したスキャンデータを NXT Viewer へ USB 送信する 5. NXT Scanner は、1 行スキャンする度に紙送りをする 6. NXT Scanner は、目的のスキャンが終わるか、ユーザーが、「スキャン停止」を要求するまでスキャンを実行する
異常フロー	異常が発生した場合はすぐに処理を中断する

表 2-3　ユースケース記述（スキャン画像を見る）

ユースケース名	NXT Viewer スキャン画像を見る
正常フロー	**<<リアルタイム画像の表示>>** 1. ユーザは、「スキャン開始」を要求する 2. NXT Viewer は、NXT Scanner から画像データを USB 受信する 3. NXT Viewer は、スキャン中の画像を 1 行単位で更新する 4. ユーザは、スキャンした画像の保存も可能 **<<スキャン済み画像の表示>>** 1. ユーザは、保存した画像の読み込みも可能
異常フロー	なし

表 2-4　ユースケース記述（画像補正をする）

ユースケース名	NXT Viewer 画像補正をする
正常フロー	1. ユーザは、補正したい画像を用意し(リアルタイム、読み込み)、画像補正を要求する 2. NXT Viewer は、画像を補正する 3. NXT Viewer は、補正した画像を表示する
異常フロー	なし

3　NXT Scanner の機構

NXT Scanner 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。

3.1　構造

NXT Scanner 本体の概観を図 3-1 に示します。

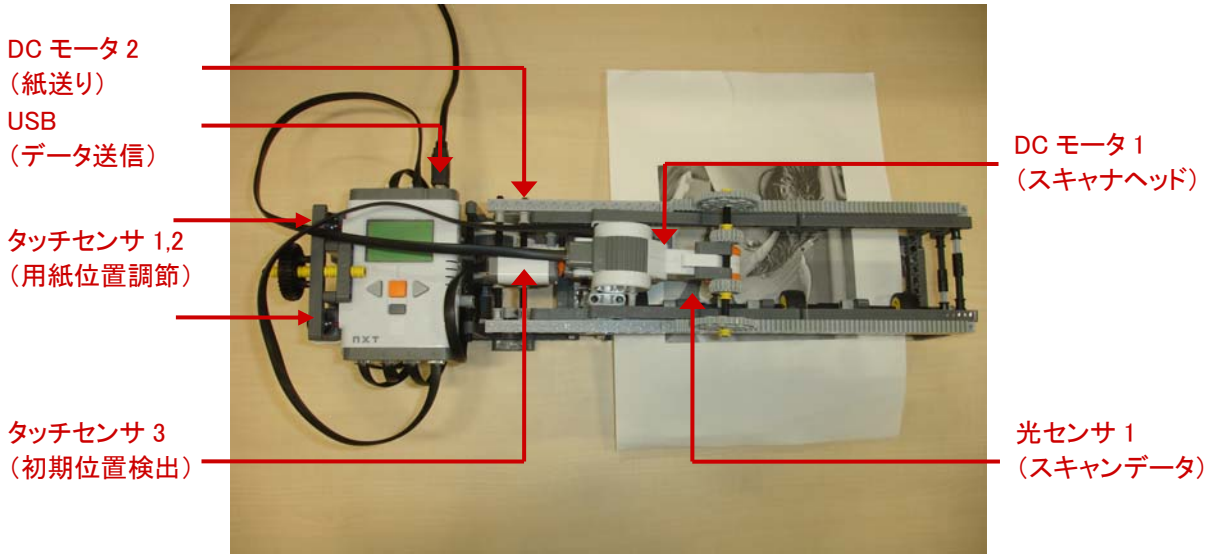


図 3-1　NXT Scanner本体

NXT Scanner 本体のハードウェア構成は機能的に 3 つに大別されます。

- 操作部
- スキャナーヘッド部
- 紙送り部

操作部

操作部は、NXT Scanner のユーザインタフェース部となります。図 3-2 は操作部の拡大図です。

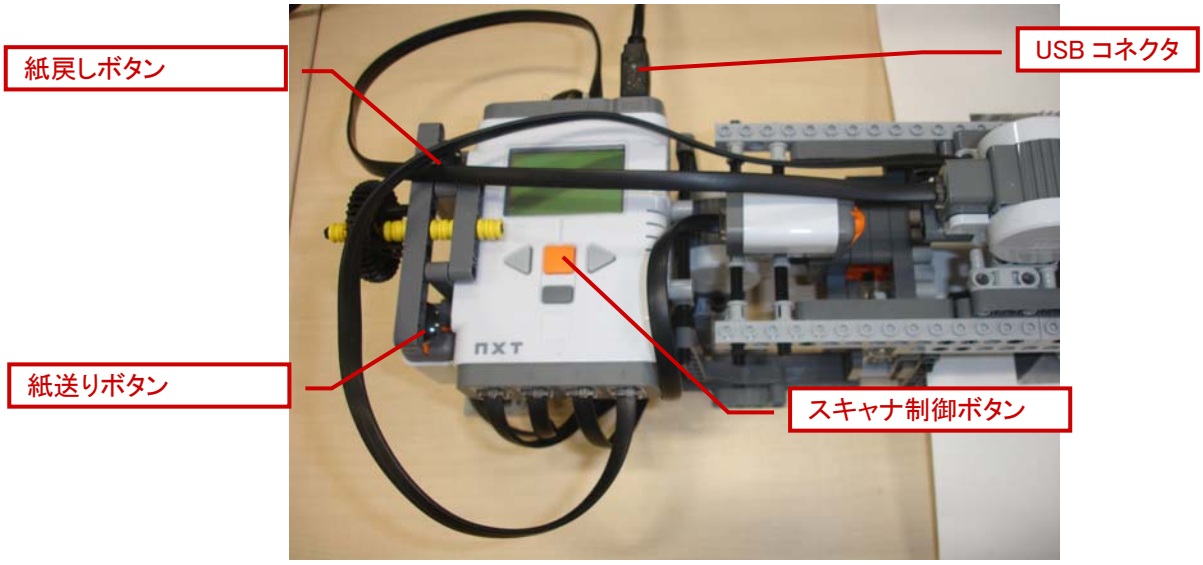


図 3-2　NXT Scanner 操作部

スキャナヘッド部

NXT Scanner のスキャナヘッド部を本体上でスライドさせてスキャンを行います。図 3-3 はスキャナヘッド部の拡大図です。図 3-4 のように、スキャナヘッド部のギヤ機構は、ラックアンドピニオンを用いており、スキャナヘッド制御モーターの回転を水平方向への駆動に変換し位置制御を行います。スキャナヘッド部にはスキャンセンサが固定されていて、ここで画像データを取り込みます。スキャナヘッド位置リセットスイッチの使用目的は、スキャナヘッドの初期位置の検出用と、スキャナヘッド位置制御のオーバーラン検出用に使用します。

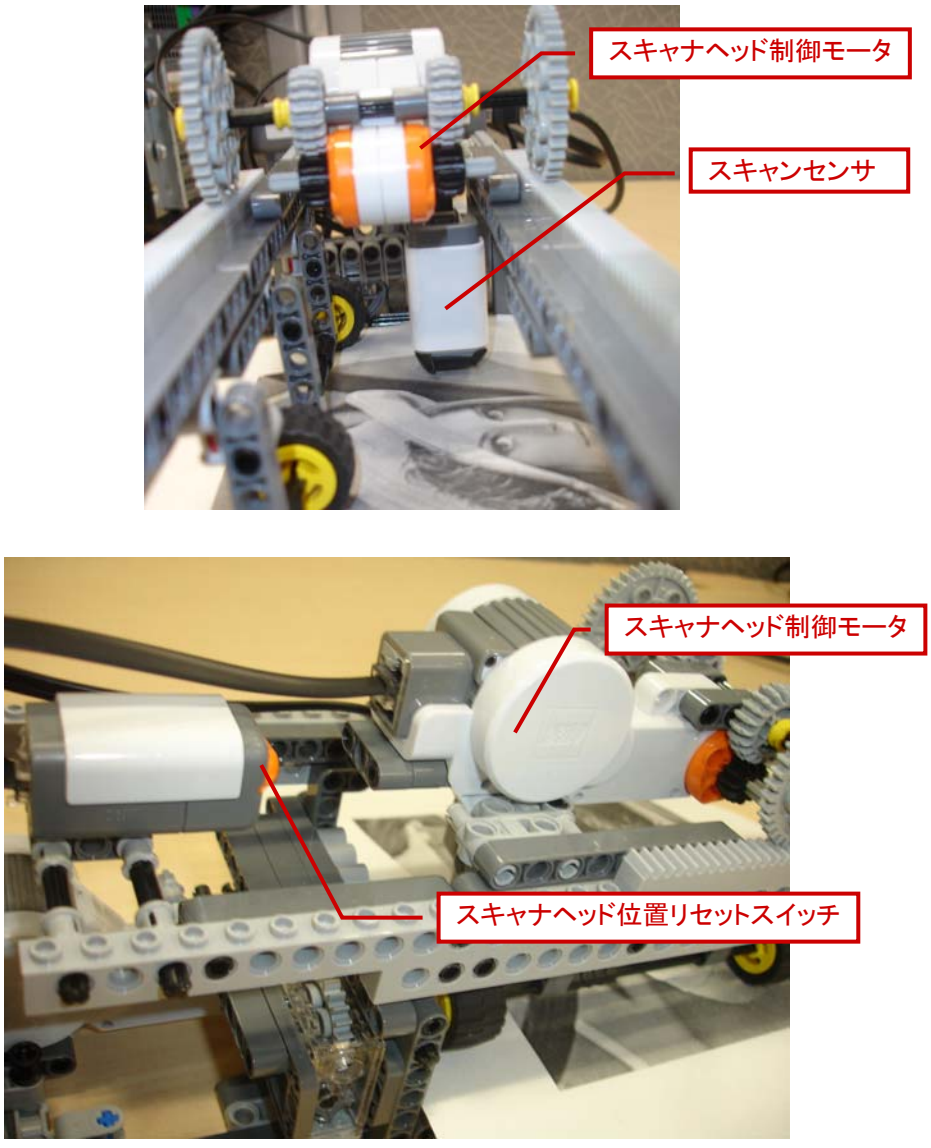


図 3-3　NXT Scanner スキャナヘッド部

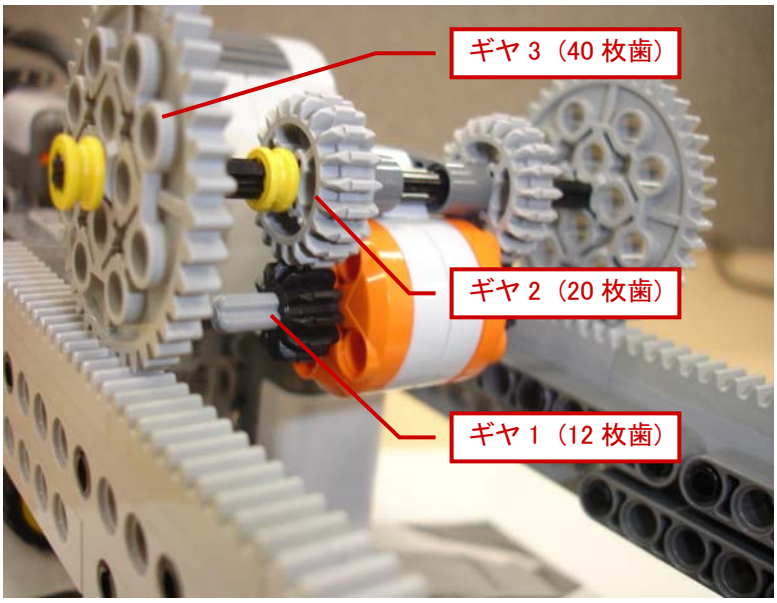


図 3-4　NXT Scanner スキャナヘッド部のギヤ機構

ギヤ比 g は次式で求まります。

$$g = \frac{gear2teeth}{gear1teeth} = \frac{20}{12} = 1.667 \tag{2.1}$$

ギヤ 3 は半径 2.0292 [cm]、1 回転の移動距離が 12.75 [cm]であることから、スキャナヘッド制御モータ回転角度 1 度でのスキャナヘッド機構の移動距離 d は次式で求まります。

$$d = \frac{12.75[cm]}{360[度]*1.667} = 0.0215[cm] \tag{2.2}$$

紙送り部

NXT Scanner の紙送り部では紙送り用ゴムローラーを回転させてスキャナ用紙の位置制御を行います。図 3-5 は紙送り部の拡大図です。図 3-6 のように、紙送り部のギヤ機構は、ウォームギヤを用いており、紙送り機構制御モーターの回転軸を変えて紙送りや紙戻しを行います。

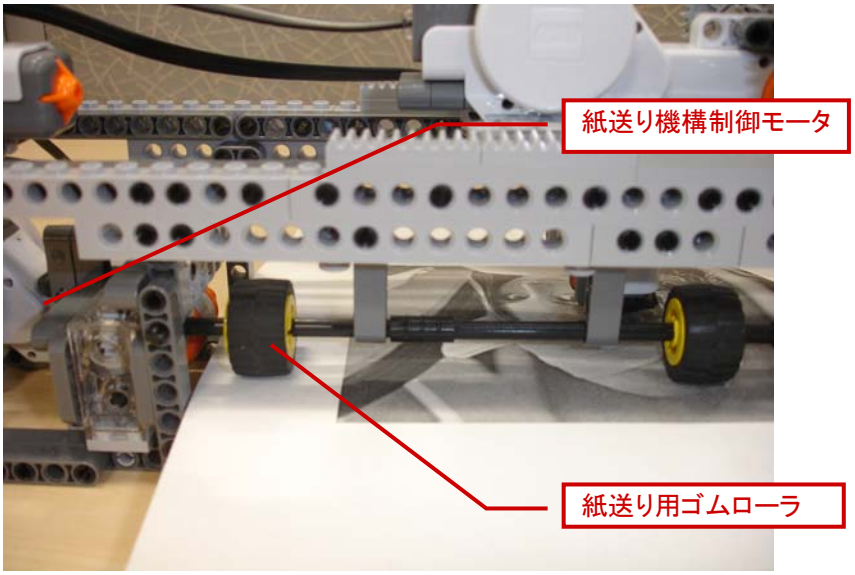


図 3-5　NXT Scanner 紙送り部

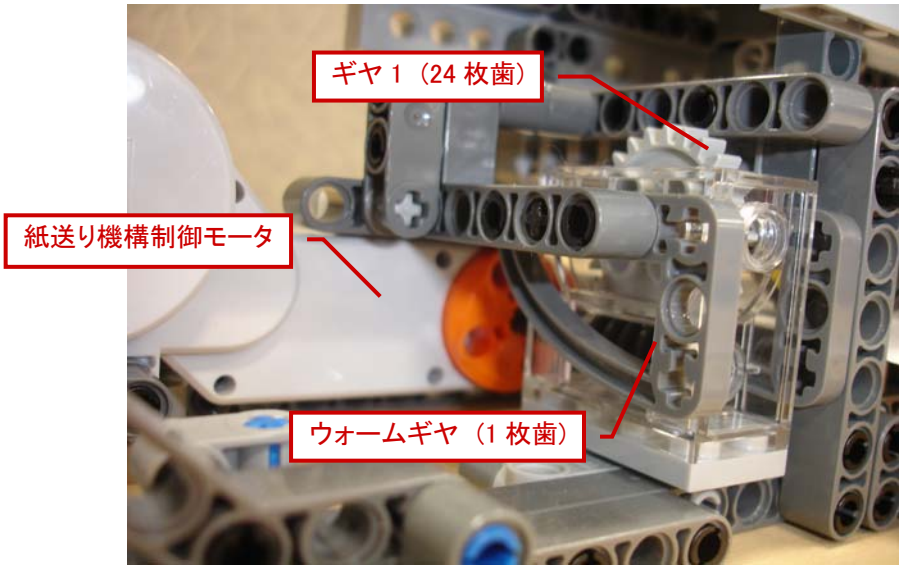


図 3-6　NXT Scanner 紙送り部のギヤ機構

ギヤ比 g は次式で求まります。

$$g = \frac{gear1teeth}{1} = \frac{24}{1} = 24 \tag{2.3}$$

紙送り用ゴムローラは半径 1.24141 [cm]、1 回転での移動距離が 7.8 [cm]であることから、紙送り機構制御モータ回転角度 1 度での用紙の移動距離 d は次式で求まります。

$$d = \frac{7.8[cm]}{360[度]*24} = 9.0278*10^{-4}[cm] \tag{2.4}$$

3.2 バックラッシュ

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 3-7 はバックラッシュの概念図です。

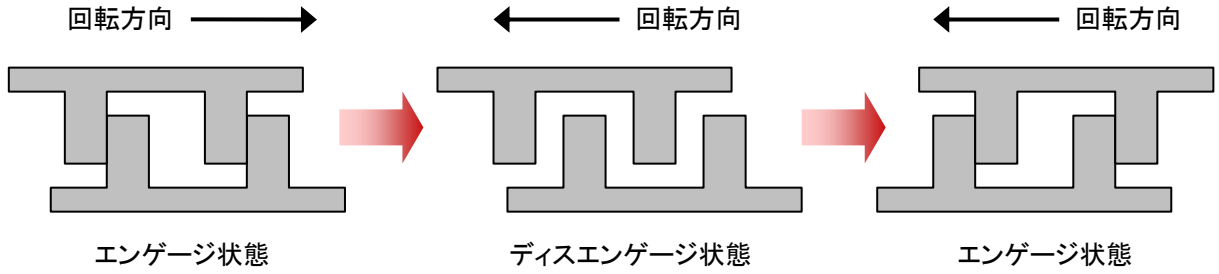


図 3-7　バックラッシュ（上側が駆動ギヤ）

3.3 センサ・アクチュエータ

NXT Scanner で使用されているセンサ・アクチュエータの特性を表 3-1 および表 3-2 に示します。

表 3-1　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
光センサ	赤外線反射光量		uint16	1000

表 3-2　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[2]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。

4　NXT Scanner のシステム設計

NXT Scanner のシステム構成を説明します。

4.1　システム概要

NXT Scanner のシステム構成図を図 4-1 に示します。「モード制御システム」「紙送り制御システム」「スキャン制御システム」「USB 通信制御システム」の 4 つに区分して制御を行います。

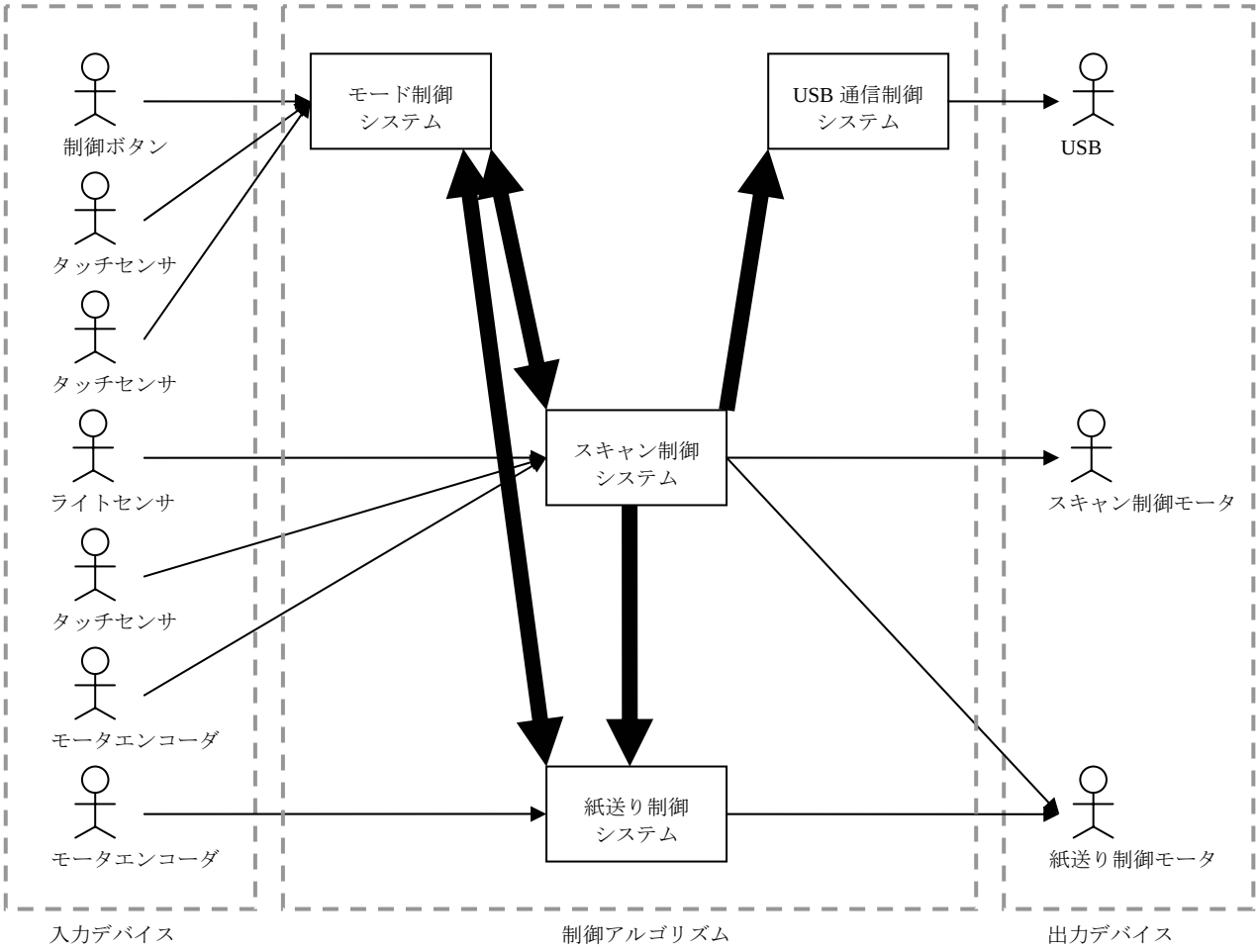


図 4-1　システム構成全体

4.2 モード制御システム

モード制御システムでは、発生したイベントに応じて NXT Scanner 内部の制御モードを切り替えます。制御モードは「初期化モード」「アイドルモード」「紙送りモード」「紙戻しモード」「スキャンモード」「異常モード」を用意しています。図 4-2 は制御モードの状態遷移を表しています。

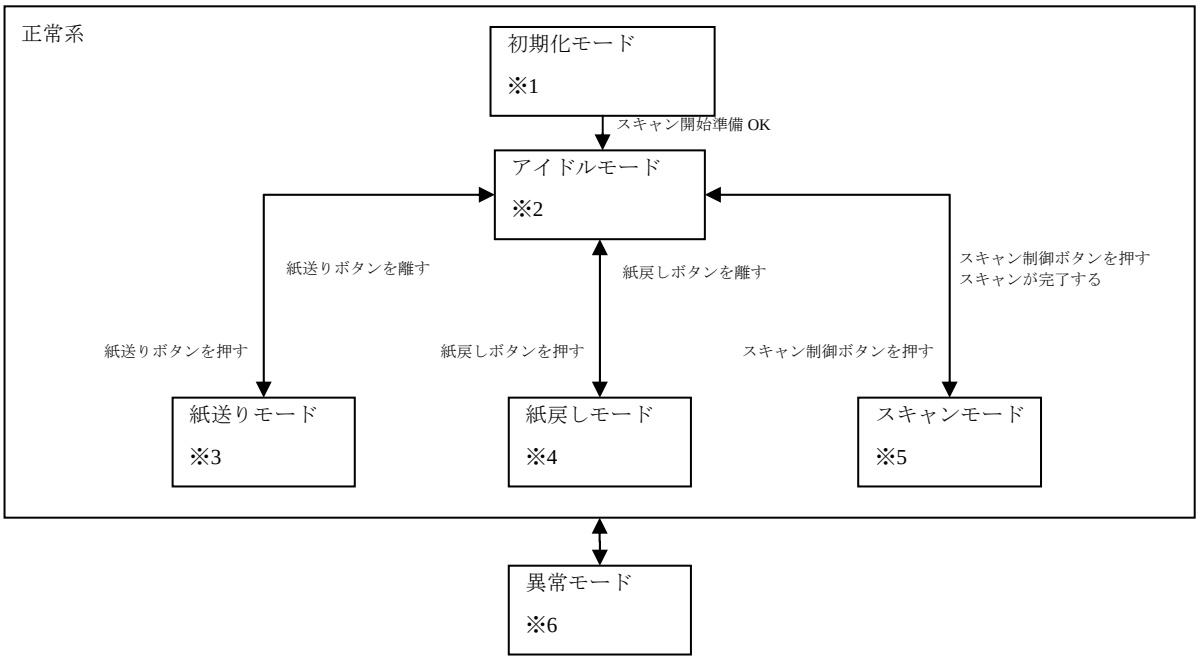


図 4-2　モード制御システム概要図

表 4-1　モード制御システム補足

システム名	モード制御システム
概要	NXT Scanner の制御モードを制御する
状態説明	1. 初期化モード：スキャン制御システム、もしくは、紙送り制御システムが初期化中の状態 2. アイドルモード：何もしていない待機状態 3. 紙送りモード：紙送り中状態 4. 紙戻しモード：紙戻し中状態 5. スキャンモード：スキャン中状態 6. 異常モード：スキャン制御システム、もしくは、紙送り制御システムに異常が発生した状態
補足	・ 異常モード時は、異常があった制御システムを区別できるよう異なるブザー音を鳴らす ・ 自発的にスキャンモードを終了させる時のために、スキャン制御システムの進捗状況を監視する

スキャン制御システムと紙送り制御システムは、この制御モードに従って制御を実施します。以下に正常時と異常時の例をそれぞれ示します。

正常時の紙送り要求に伴う NXT Scanner システム全体の処理フローを図 4-3 に示します。他の動作についても同様で、制御モードを要求として紙送り制御システムとスキャン制御システムを駆動します。

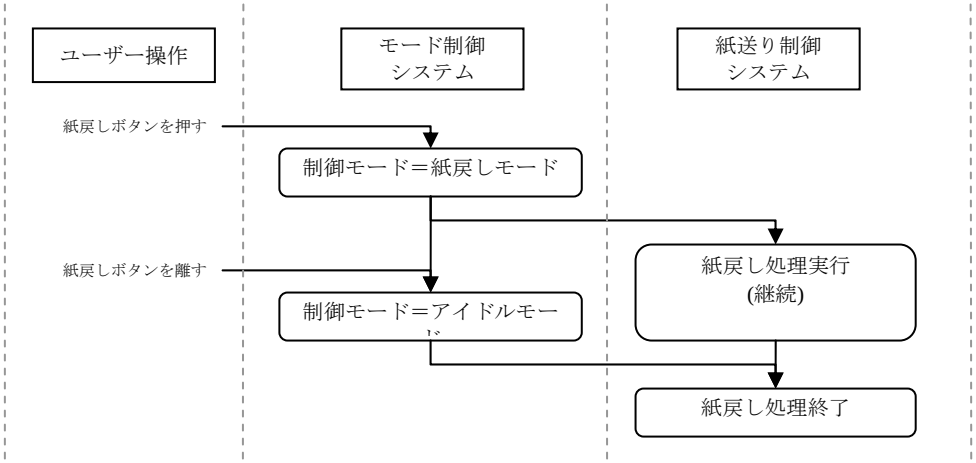


図 4-3　正常時の紙送りの処理の流れ

異常発生時の紙送り要求に伴う NXT Scanner システム全体の処理フローを図 4-4 に示します。他の動作中でも同様で、異常発生したシステムは、モード制御システムに異常を通知し、モード制御システムが制御モードを異常モードにすることで、各システムに異常処理の要求が発行されます。

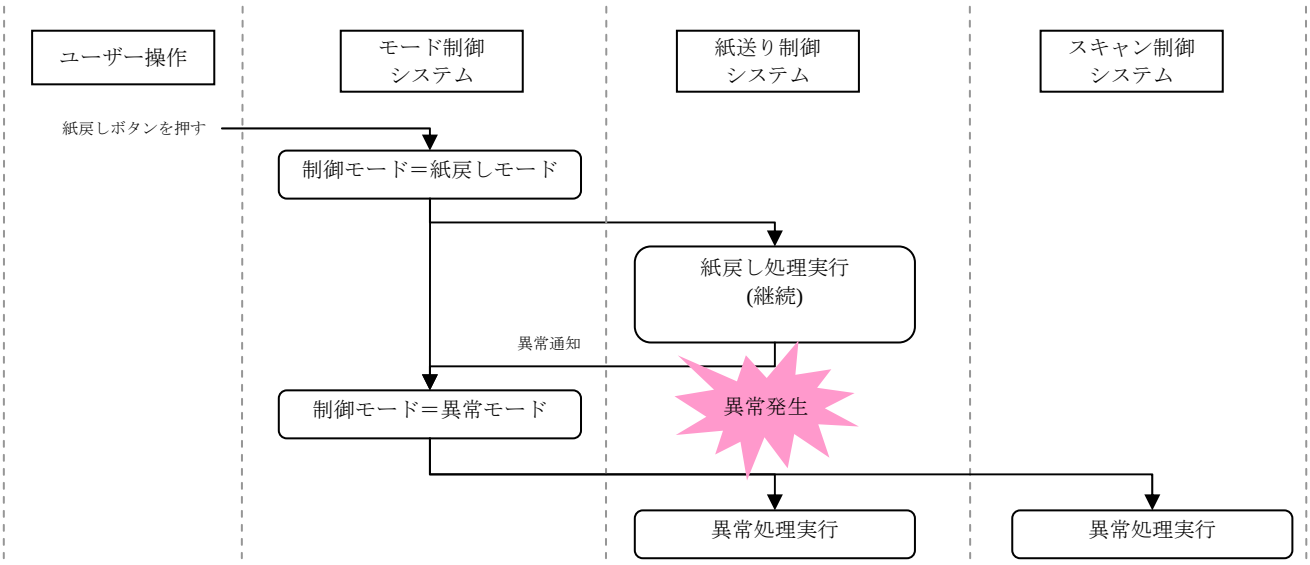


図 4-4　異常発生時の紙送りの挙動

4.3 紙送り制御システム

紙送り制御システムでは、NXT Scanner の用紙位置を制御します。用紙の位置制御には、スキャン動作前に行う制御と、スキャン動作中に行う制御があります。図 4-5 のように、スキャン制御中に行う紙送りだけは、スキャン制御システムの指示タイミングに従って紙送り制御を行います。

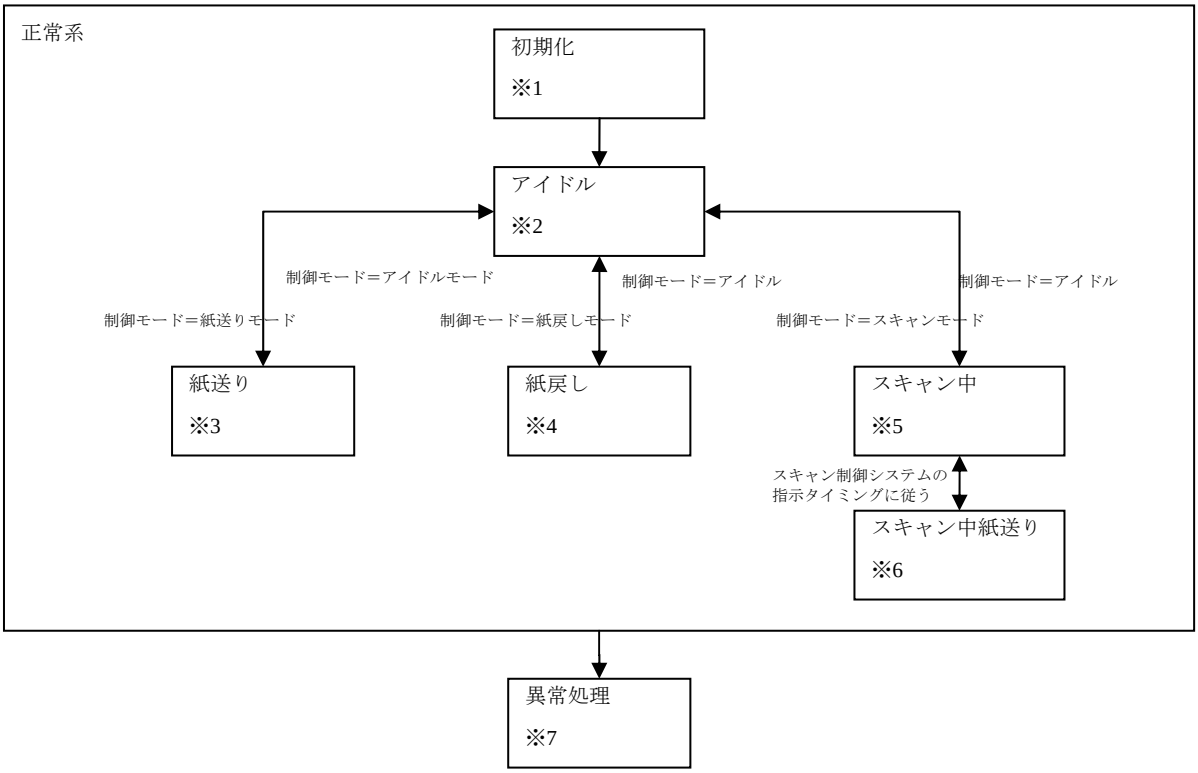


図 4-5　紙送り制御システム概要図

表 4-2　紙送り制御システム補足

システム名	紙送り制御システム
概要	NXT Scanner の用紙位置を制御する
状態説明	1. 初期化：初期化用紙送り中状態 2. アイドル：何もしていない待機状態 3. 紙送り：紙送り中状態 4. 紙戻し：紙戻し中状態 5. スキャン中：スキャン中、かつ、紙送りしていない状態 6. スキャン中紙送り：スキャン中、かつ、紙送り中状態 7. 異常モード：スキャン制御システム、もしくは、紙送り制御システムに異常が発生した状態
補足	・ 紙送り制御システムに異常が発生した場合は、ただちにモード制御システムに異常を通知する ・ 異常モードでは、紙送りを停止させる

4.4　スキャン制御システム

スキャン制御システムでは、スキャナヘッド移動とスキャン動作を行います。スキャン動作は、図 4-6 のように用紙上の 1 辺約 10cm 正方形の範囲内を 255×127 のポイントとしてスキャンします。縦方向のスキャン数が横方向の半分になっているのは、スキャンの高速化を目的としているためです。NXT viewer では、縦方向の情報不足分は、NXT Viewer での画像補正で補間をして 255×254 のデータにします。

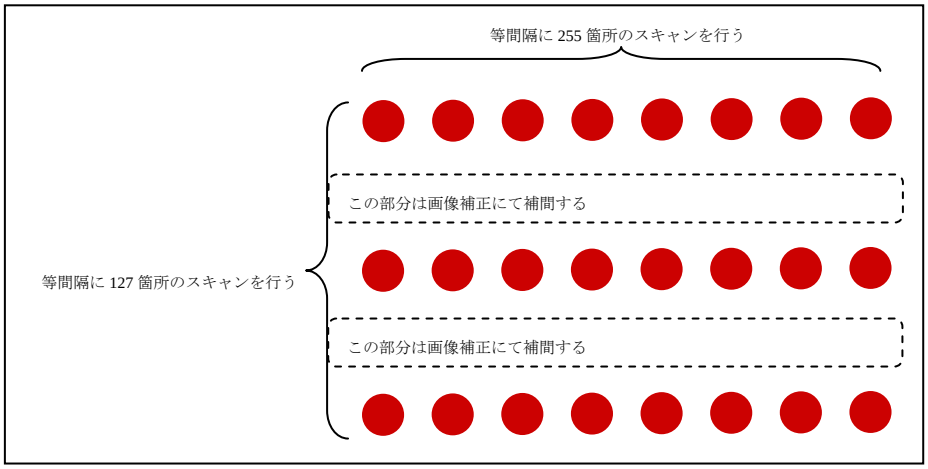


図 4-6　スキャンポイント

図 4-7 に示すように、スキャナヘッド移動、スキャナデータの取り込み、スキャン中の紙送り要求をシーケンシャルに行います。

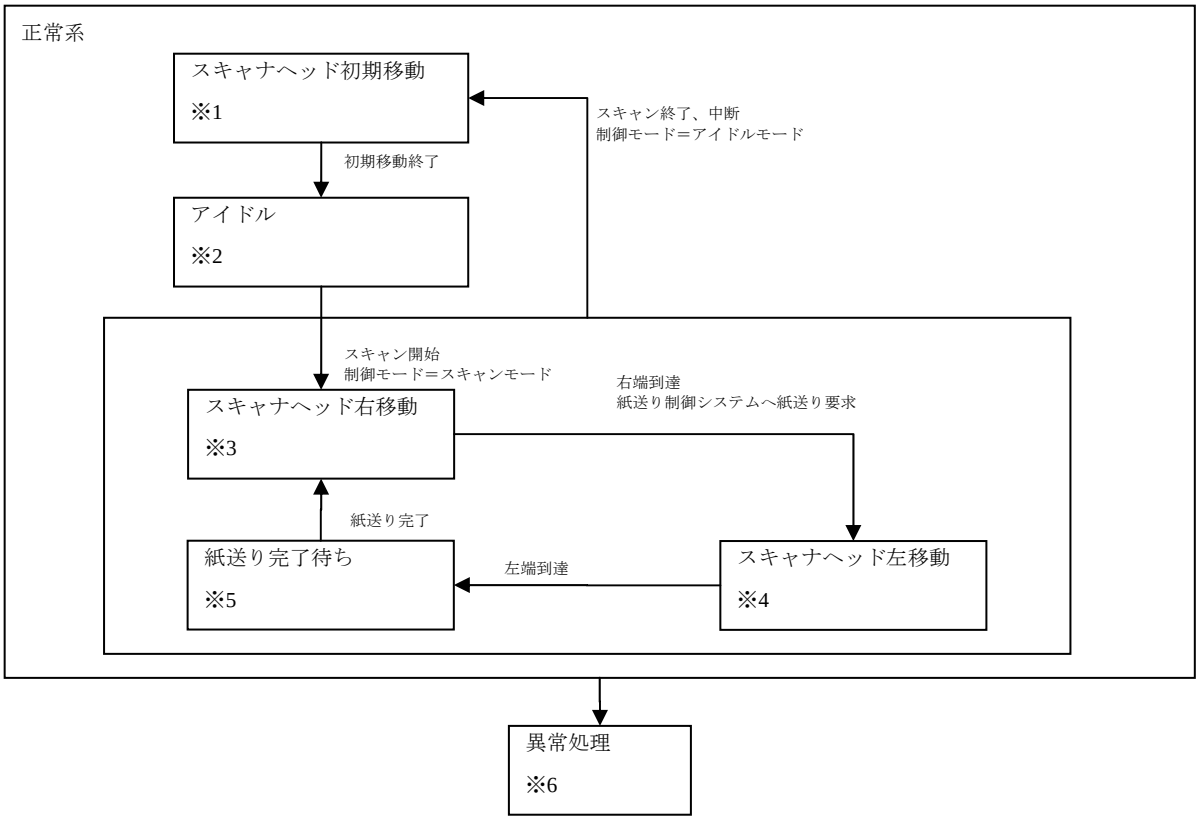


図 4-7　スキャン制御システム概要図

表 4-3　スキャン制御システム補足

システム名	スキャン制御システム
概要	NXT Scanner のスキャン動作を制御する
状態説明	1. スキャナヘッド初期移動：スキャナヘッドを初期位置に移動させている状態 （スキャナヘッドを左に移動させて、スキャナヘッド初期位置検出 SW を検出するまで移動を続ける） 2. アイドル：スキャナヘッドが初期位置にあり、スキャン開始の待機状態 3. スキャナヘッド右移動：スキャナヘッドが右に移動し、スキャンデータを取り込んでいる状態 （紙送り処理要求は、右移動終了時に行われる） 4. スキャナヘッド左送り：スキャナヘッドが左に移動している状態 （紙送り処理は、移動と平行して行われる） 5. 紙送り完了待ち：紙送りの完了監視 6. 異常モード：スキャン制御システム、もしくは、紙送り制御システムに異常が発生した状態
補足	・ スキャンシステムに異常が発生した場合は、ただちにモード制御システムに異常を通知する ・ スキャンデータの取り込みは、スキャナヘッド右移動時のみ行う ・ スキャンが終了したらスキャン動作を終了させて、スキャナヘッド初期移動を行う ・ 異常モードでは、ただちにスキャナヘッドの移動を停止させる

4.5 USB 通信制御システム

USB 通信制御システムでは、スキャンしたデータを NXT Viewer へ USB 送信します。USB 通信は NXT Viewer との独自通信プロトコルに従って行われます。図 4-8 に示すとおり、スキャンデータを格納するバッファを 2 つ用意します。USB 通信制御システムは、スキャン制御システムが格納するバッファを監視し、バッファが USB 送信可能状態になったら USB 送信します。スキャン制御システムと USB 通信制御システムは非同期で処理が行われます。USB 通信システムの周期は、スキャン制御システムがスキャンする周期を考慮して、2 つのバッファが満タンにならない周期に指定する必要があります。

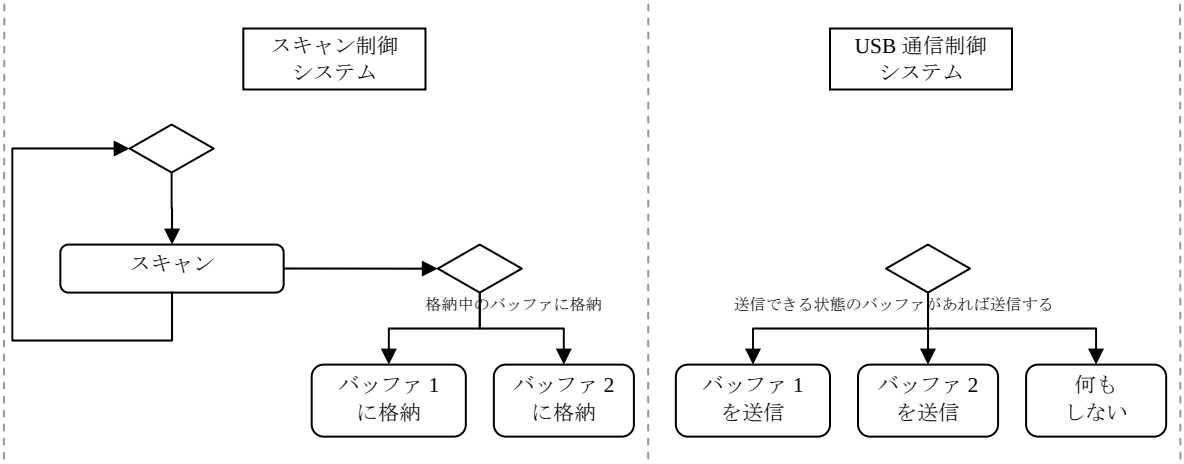


図 4-8　USB通信制御システムの流れ

5　NXT Scanner の構成設計

NXT Scanner のモデル構成を説明します。

5.1　ライブラリモデル

NXT Scanner では、システム設計した各制御システムを機能モデルとして独立させています。そして、単体検証済みの機能モデルをライブラリ登録し、NXT Scanner 統合モデルで結合しています。nxtscanner_lib.mdl は図 5-1 のように各制御システムをライブラリモデルとして登録しています。各機能モデルの詳細設計は次章で説明します。

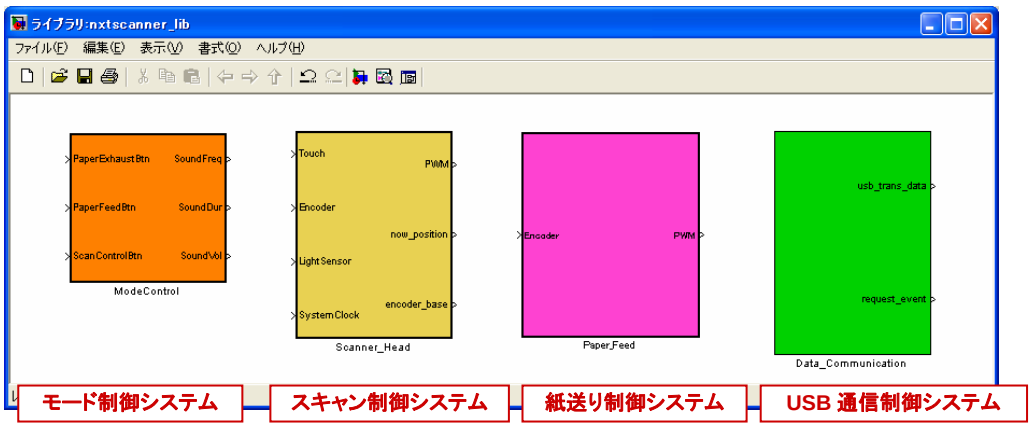


図 5-1　ライブラリモデル

5.2　システム全体の共有データ

表 5-1 は、NXT Scanner のシステム全体で共有するデータです。

表 5-1　システム間共有データ一覧

要求項目	項目詳細
制御モード： （ControlMode）	・　システム全体の制御モードを定義する。 ERROR　（エラー状態） INITIALIZE　（各機構初期化中状態） IDLE　（制御/操作無し状態） PAPERFEED　（紙送り機能動作状態） PAPEREXHAUST　（紙排出機能動作状態） SCAN　（スキャン制御機能動作状態） ・　初期値は、INITIALIZE。 ・　制御モードのデータスコープはシステム全体とする。 ・　書き込み権限はモード制御システムのみとする。

エラーコード： （ErrorCode）	・ システム全体のエラー（異常）を定義する。 NO_ERROR （正常動作） ERROR_001 （スキャナーヘッド異常） ERROR_002 （紙送り/紙排出機構異常） ・ 初期値は、NO_ERROR。 ・ エラーコードのデータスコープはシステム全体とする。
スキャンモード： （ScanMode）	・ スキャン制御システムのスキャンモードを定義する。 SCAN_INIT （SCAN 初期化モード） SCAN_IDLE （SCAN 待機モード） SCAN_SCAN （SCAN 中モード） SCAN_FEED （SCAN 紙送り中モード） ・ 初期値は、SCAN_INIT。 ・ スキャンモードのデータスコープはシステム全体とする。 ・ 書き込み権限はスキャン制御システムのみとする。
紙送りモード： （FeedMode）	・ スキャン制御システムと紙送り制御システム間の紙送りモードを定義する。 FEED_INIT（初期化モード） FEED_IDLE （アイドルモード（操作無し、紙送り完了）） FEED_PAPERFEED （紙送り中モード） ・ 初期値は、FEED_INIT。 ・ 紙送りモードのデータスコープはシステム全体とする。 ・ 書き込み権限はスキャン制御システムと紙送り制御システムとする。
スキャンバッファ情報： （ScannerBuffer_Info）	・ スキャン制御システムと USB 通信制御システム間のスキャンバッファの基本共有情報を定義する。 TRANS_FINISH （転送完了） STORING （格納中） FULL_UP （満タン） ・ 初期値は、TRANS_FINISH。 ・ スキャンバッファ情報のデータスコープはシステム全体とする。 ・ 書き込み権限はスキャン制御システムと USB 通信制御システムとする。
スキャンバッファステータス： （ScannerBuffer_Status）	・ スキャン制御システムと USB 通信制御システム間のスキャンバッファのステータスを定義する。 ・ 未使用時はゼロに初期化する。 ・ 初期値はゼロ。 ・ ELP（スキャン各行の行末パケット情報）:bit4 0：行末パケット以外（先頭および途中） 1：行末パケット（スキャン終了時の最終パケット含む） ・ START （START 情報） :bit3

	0：通常スキャン中 1：START ・ STOP （STOP 情報） :bit2 0：通常スキャン中 1：STOP ・ DIR （スキャン方向情報） :bit1 0：右向き 1：左向き ・ VALID （有効データ識別情報） :bit0 0：無効データ 1：有効データ ・ スキャンバッファステータスのデータスコープはシステム全体とする。 ・ 書き込み権限はスキャン制御システムと USB 通信制御システムとする。
スキャンバッファ数： （ScannerBuffer_Number）	・ スキャン制御システムと USB 通信システム間のスキャンバッファの送信用データ数（0～30）を定義する。 ・ 未使用時はゼロに初期化する。 ・ 初期値はゼロ。 ・ スキャンバッファ数のデータスコープはシステム全体とする。 ・ 書き込み権限はスキャン制御システムと USB 通信制御システムとする。
スキャンバッファデータ： （ScannerBuffer_Data）	・ スキャナン制御システムと USB 通信制御システム間のスキャンバッファのスキャンデータを定義する。 ・ 未使用時はゼロに初期化する。 ・ 初期値はゼロ。 ・ スキャンバッファデータのデータスコープはシステム全体とする。 ・ 書き込み権限はスキャン制御システムと USB 通信制御システムとする。
スキャンパケット数： （ScannerBuffer_Pnum）	・ スキャン制御システムと USB 通信制御システム間のスキャンバッファの現在スキャン行のパケット累計を定義する。 ・ 未使用時はゼロに初期化する。 ・ 初期値はゼロ。 ・ スキャンバッファ数のデータスコープはシステム全体とする。 ・ 書き込み権限はスキャン制御システムとする。

6　NXT Scanner の詳細設計

NXT Scanner の各機能モデルの詳細設計を説明します。

6.1　enumerated type の表現（R2008b 新機能）

各機能モデルでは、R2008b の新機能である enumerated type を使用します。NXT Scanner では enumerated type を以下のように使用しています。

- enumerated type を Data Store Memory ブロックに割り当て、共有データとして使用する
- さらに Data Store Memory に Simulink Data Object を割り当て、メモリ配置方法を指定できるようにする

enumerated typeの設定手順

enumerated type の設定手順は以下の通りです。

1. M ファイルにて enumerated type の登録
2. Data Store Memory ブロックのデータタイプを enumerated type に指定

設定の具体例

モード制御システムの制御モードをグローバル変数 ControlMode と定義します。この ControlMode のデータタイプとして ControlModeEnum という名前の enumerated type を登録します。Enumerated type を登録する M ファイルは下記の通りです。

ControlModeEnum.m

```
classdef(Enumeration) ControlModeEnum < Simulink.IntEnumType
    enumeration
        ERROR(-1) % エラーモード
        INITIALIZE(0)   % 各機構初期化
        IDLE(1)   % アイドルモード(操作無し)
        PAPERFEED(2) % 紙送りモード
        PAPEREXHAUST(3) % 紙排出モード
        SCAN(4) % スキャナ制御モード
    end
end
```

グローバル変数 ControlMode を図 6-1 のように Data Store Memory ブロックに割り当て、M ファイルで登録した ControlModeEnum をデータタイプに指定します。また、初期値についても enumerated type で指定します。

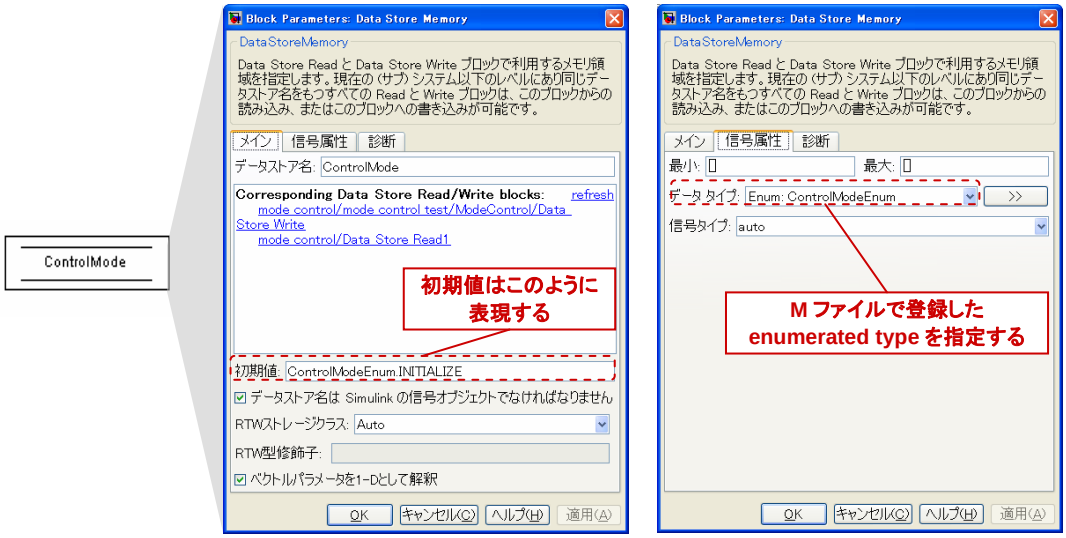


図 6-1　enumerated typeの設定方法

6.2 Simulink 関数の活用（R2008b 新機能）

紙送り制御モデルでは、R2008b の新機能である Simulink 関数機能を使用します。Simulink 関数を使用すると、従来では呼び出し元と同じ Stateflow でしか記述できなかった関数を Simulink のサブシステムとして記述する事ができます。

紙送り制御フローで使用している表 6-1 のすべての関数は、呼び出し元が Stateflow であるにもかかわらず、その定義が Simulink のサブシステムとして記述されています。Simulink 関数の登場により、このような記述が可能になりました。

Simulink 関数は、Stateflow 上で該当関数をダブルクリックし、Simulink のモデルウィンドウを開いて編集します。

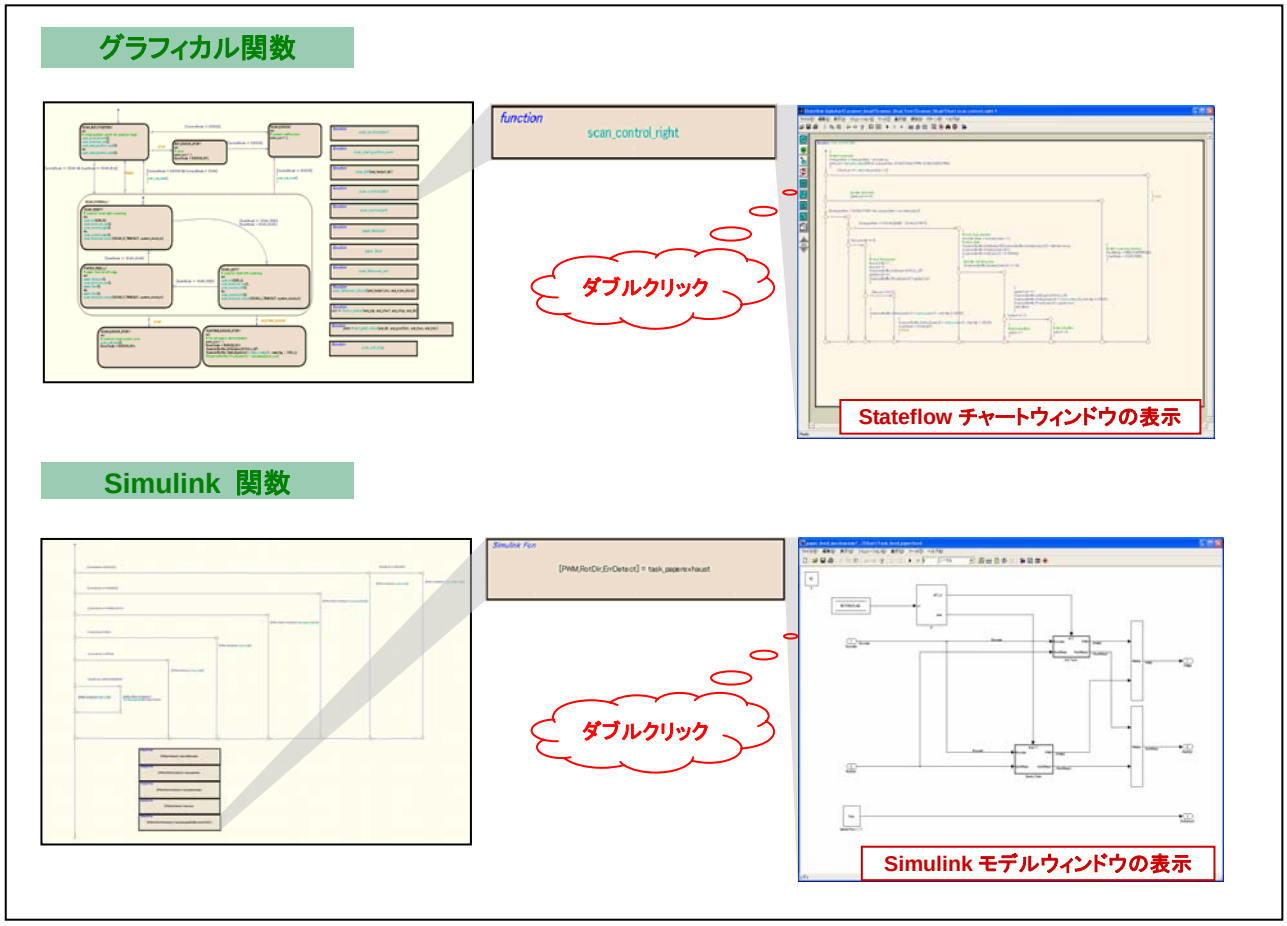


図 6-2　Simulink関数と従来のグラフィカル関数の違い

6.3　モード制御モデル

mode_control.mdl はモード制御システムの機能モデルです。図 6-3 はモード制御モデルのトップレイヤとライブラリ登録しているサブシステムの関係を表しています。

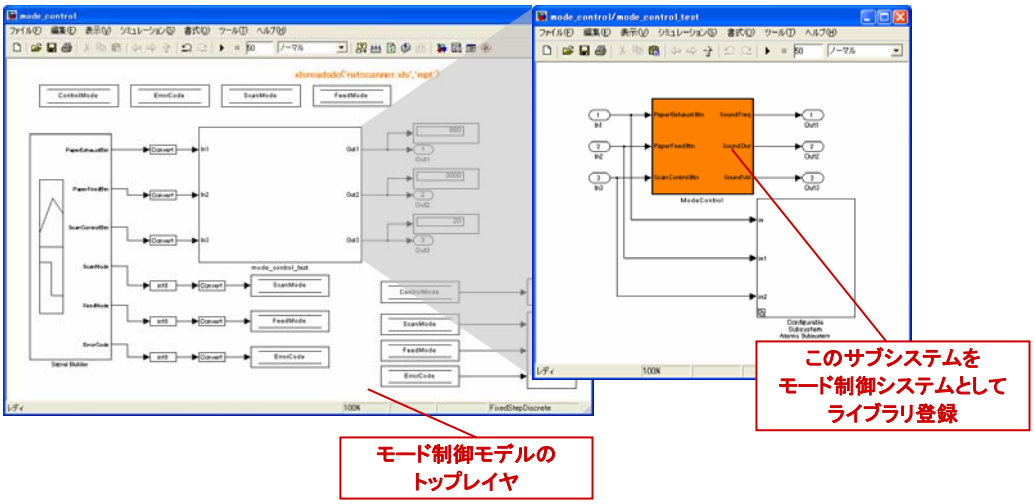


図 6-3　モード制御モデルとライブラリの関係

単体検証を行ったサブシステムをライブラリ登録するため、トップレイヤには検証用の入力信号、出力信号のモニタを配置しており、シミュレーションを実行することで検証を行うことが出来ます。

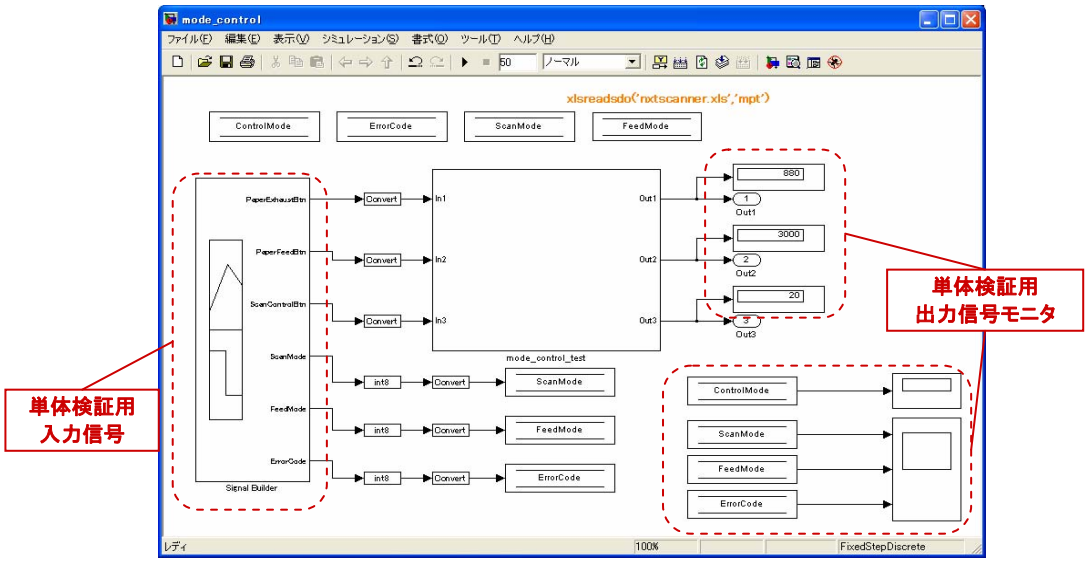


図 6-4　モード制御モデルのトップレイヤ

共有データ

タスク間で共有するデータとして図 6-5 の Data Store Memory ブロックを使用しています。ErrorCode、ScanMode、FeedMode は ControlMode と同様の手順で enumerated type をデータタイプに持つ Data Store Memory ブロックを使用しています。

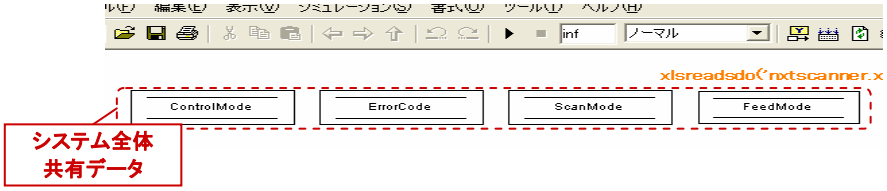


図 6-5　モード制御モデルの共有データ

アルゴリズム実装

モード制御モデルは図 6-6 で示すように、ライブラリ登録したサブシステムに実装しています。Stateflow に入力している ScanControl 入力信号は、図 6-7 で示すとおり、スキャン制御ボタンによるスタート/ストップ要求と全スキャンが終了した時の自発的なストップ要求から成り立っています。自発的なストップ要求は、共有データのスキャンモード（ScanMode）を監視し、最終データのスキャンが終了した時に発行されます。また、図 6-8 に示すように Stateflow を用いてシステム設計で作成したアルゴリズムを実現していることが確認できます。

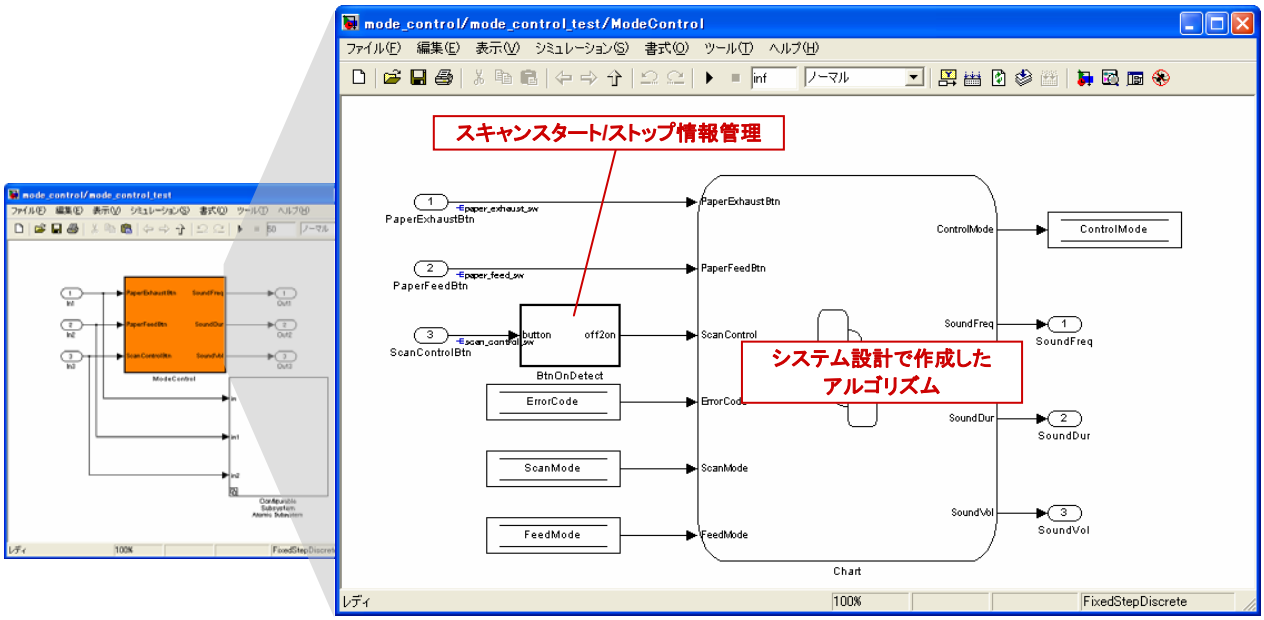


図 6-6　モード制御モデルとライブラリの関係

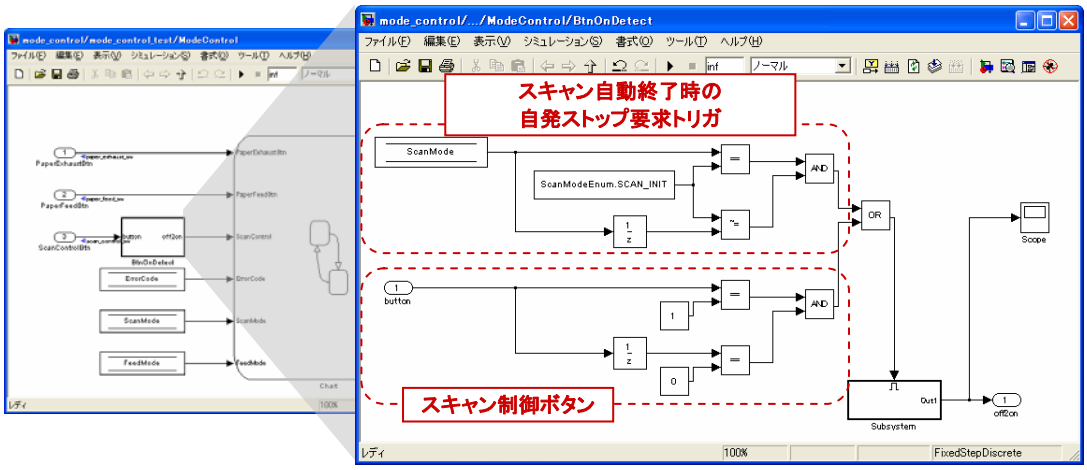


図 6-7　モード制御モデルのスキャンスタート/ストップ情報管理

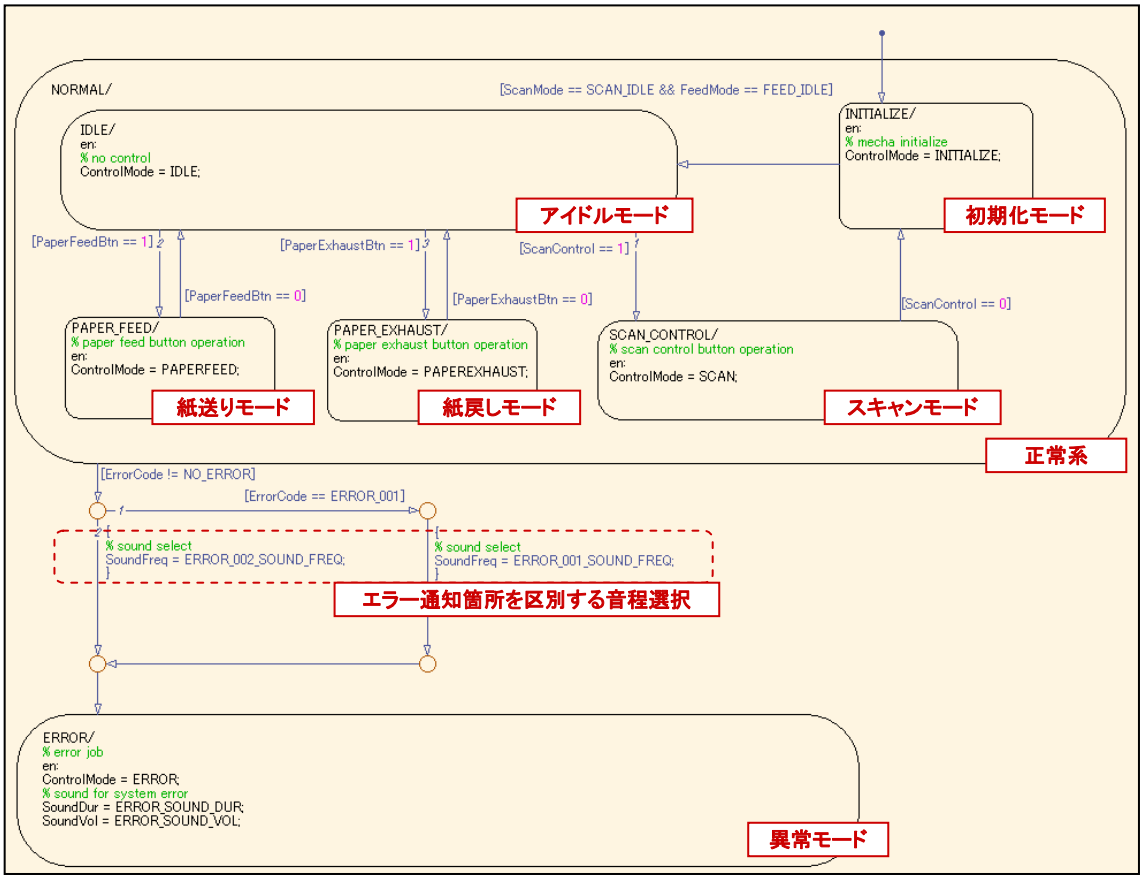


図 6-8　モード制御モデルの主アルゴリズム

6.4 紙送り制御モデル

paper_feed_mechanism.mdl は紙送り制御システムの機能モデルです。図 6-9 は紙送り制御モデルのトップレイヤとライブラリ登録しているサブシステムの関係を表しています。

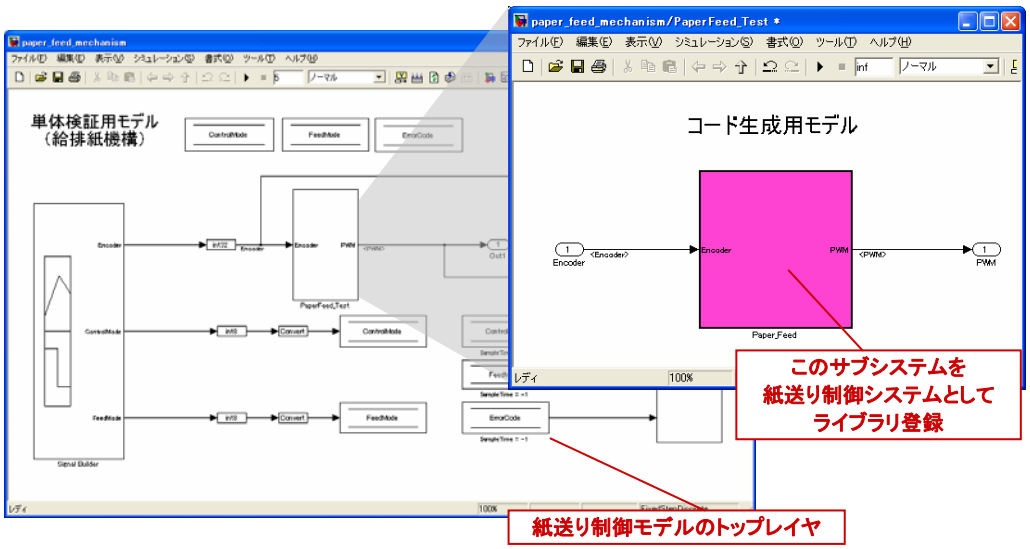


図 6-9　紙送り制御モデルとライブラリの関係

他の機能モデルのトップレイヤと同様に、トップレイヤには検証用の入力信号、出力信号のモニタを配置しており、シミュレーションを実行することで単体機能検証を行うことが出来ます。

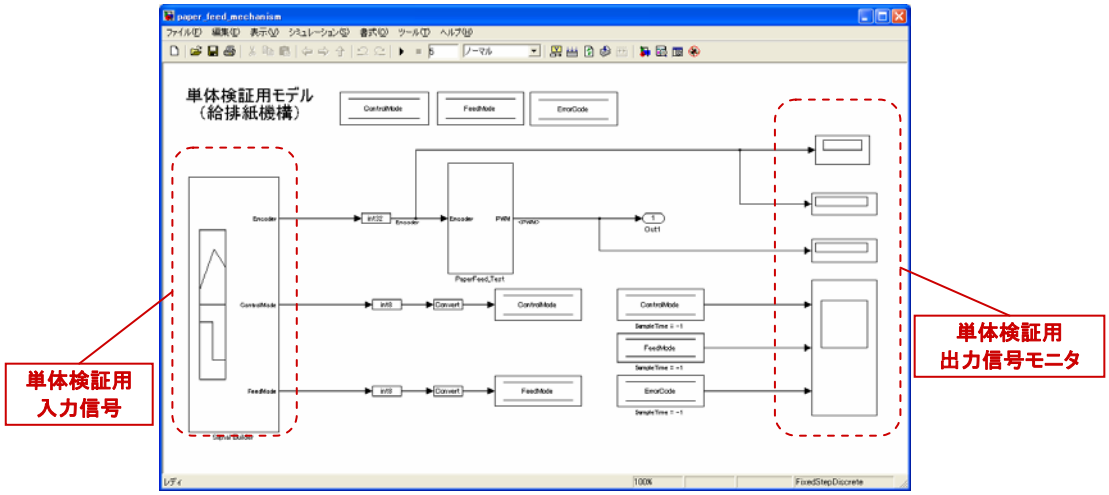


図 6-10　紙送り制御モデルのトップレイヤ

アルゴリズム実装

紙送り制御モデルは図 6-11 のように、Stateflow で記述した「制御フロー部」と、Simulink で記述した「エラー検出部」で構成されています。

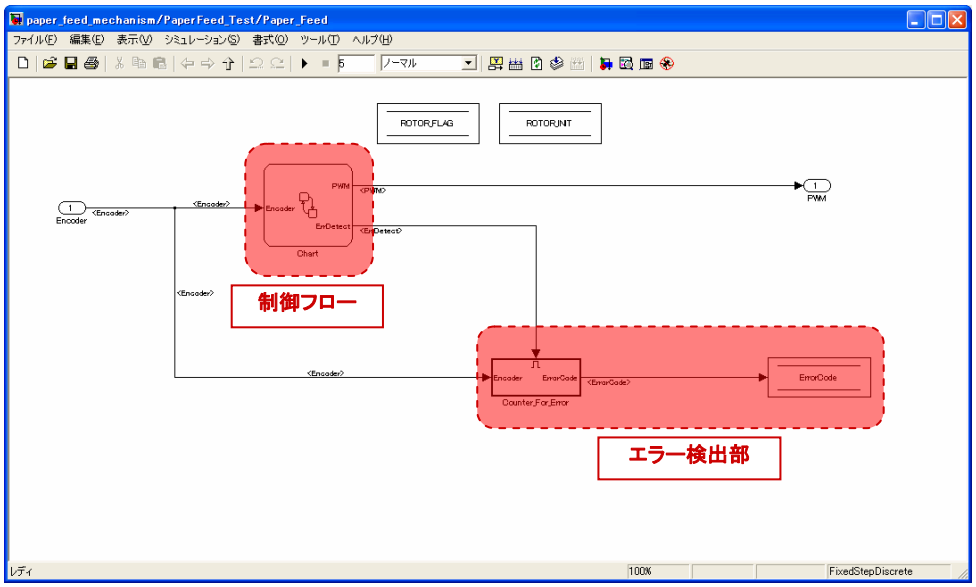


図 6-11　紙送り制御ライブラリモデルのトップレイヤ

制御フロー部

図 6-12 はフローチャート形式で設計された制御フロー部です。ここでは各制御システムから受け取った共有データ（制御モード、紙送りモード）に応じた分岐処理を行っています。下側には、このフローチャートから呼び出される Simulink 関数群が配置されています。各関数の概要を表 6-1 にまとめてあります。

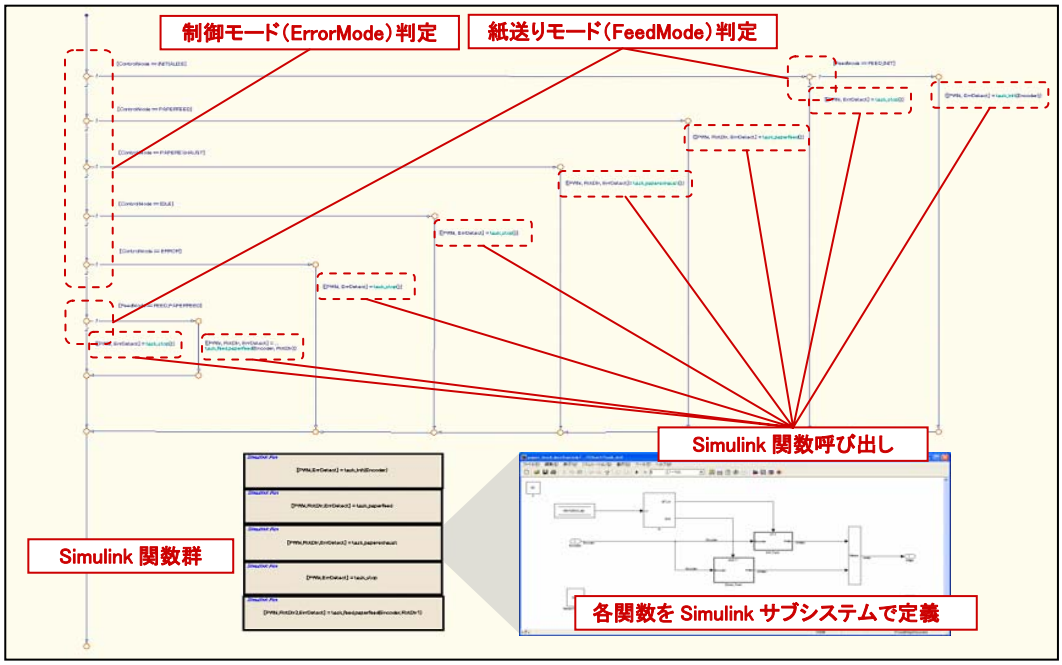


図 6-12　紙送り制御フロー部

表 6-1　紙送り制御フロー部のSimulink関数群

Simulink 関数名	引数	戻り値	概要
task_init	エンコーダ値（DC モータ 2（紙送り））	今回 PWM 値 有効/無効フラグ（Error 判定）	初期化関数 （電源投入直後、一定量の紙送りを強制的に行う）
task_paperfeed	-	今回 PWM 値 用紙進行方向 有効/無効フラグ（Error 判定）	紙送り実行関数（紙送りボタン操作）
task_paperexhaust	-	今回 PWM 値 用紙進行方向 有効/無効フラグ（Error 判定）	紙排出実行関数（紙排出ボタン操作）
task_stop		今回 PWM 値 有効/無効フラグ（Error 判定）	紙送り/排出停止関数
task_feed_paperfeed	エンコーダ値（DC モータ 2（紙送り）） 用紙進行方向	今回 PWM 値 用紙進行方向 有効/無効フラグ（Error 判定）	紙送り実行関数（スキャン制御実行中）

紙送り制御フロー

制御フロー部は制御モード、紙送りモードといった共有データを条件にした、いくつかの分岐処理から成ります。図 6-13 は、そのフローチャート図を示しています。

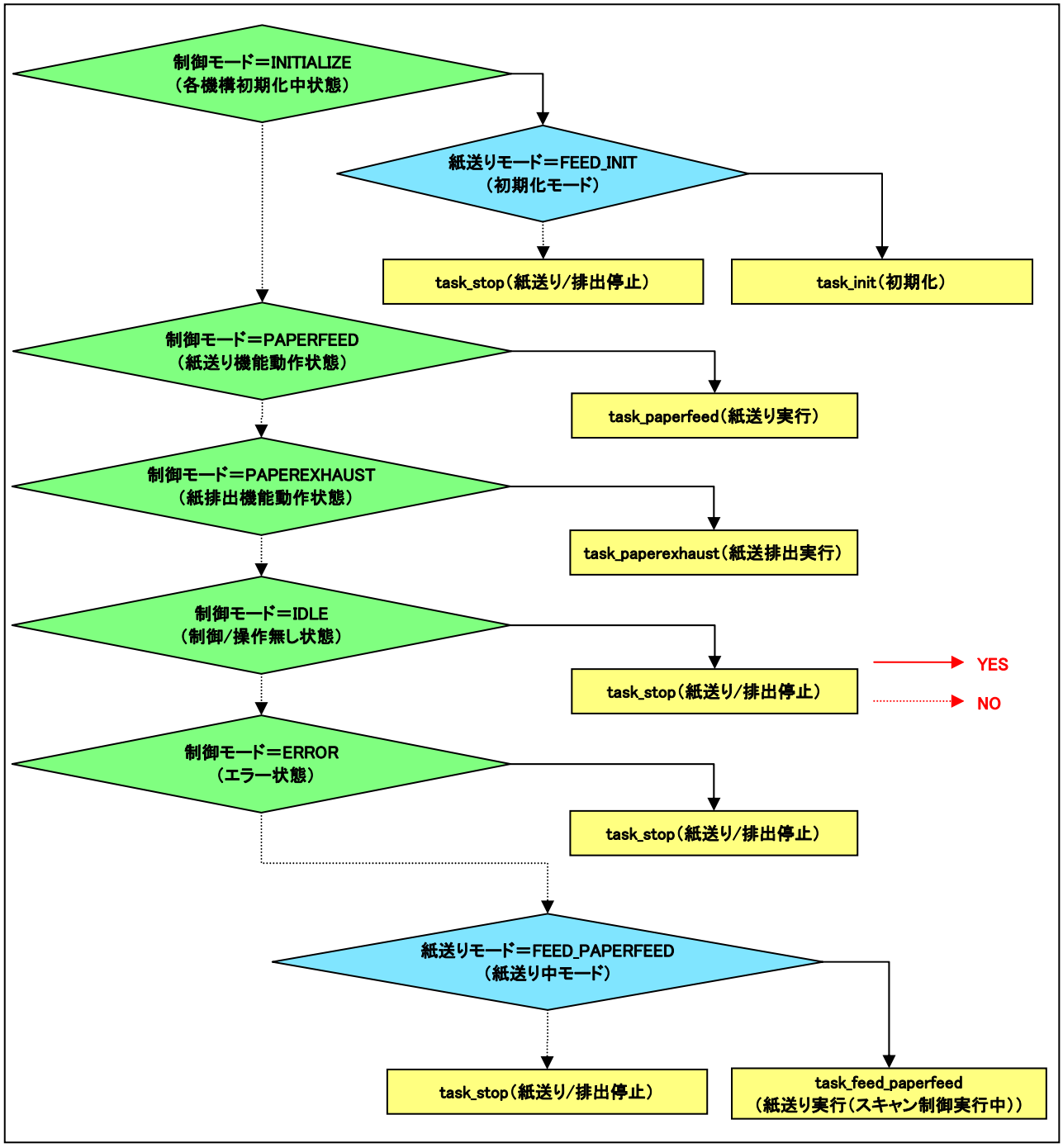


図 6-13　紙送り制御フロー部のフローチャート図

初期化処理（task_init）

紙送り制御では、紙送り用 DC モータ内部のギヤ間のバックラッシュを補償する目的で、電源投入直後に一定量の紙送りを強制的に行う（初期化する）仕様になっています。

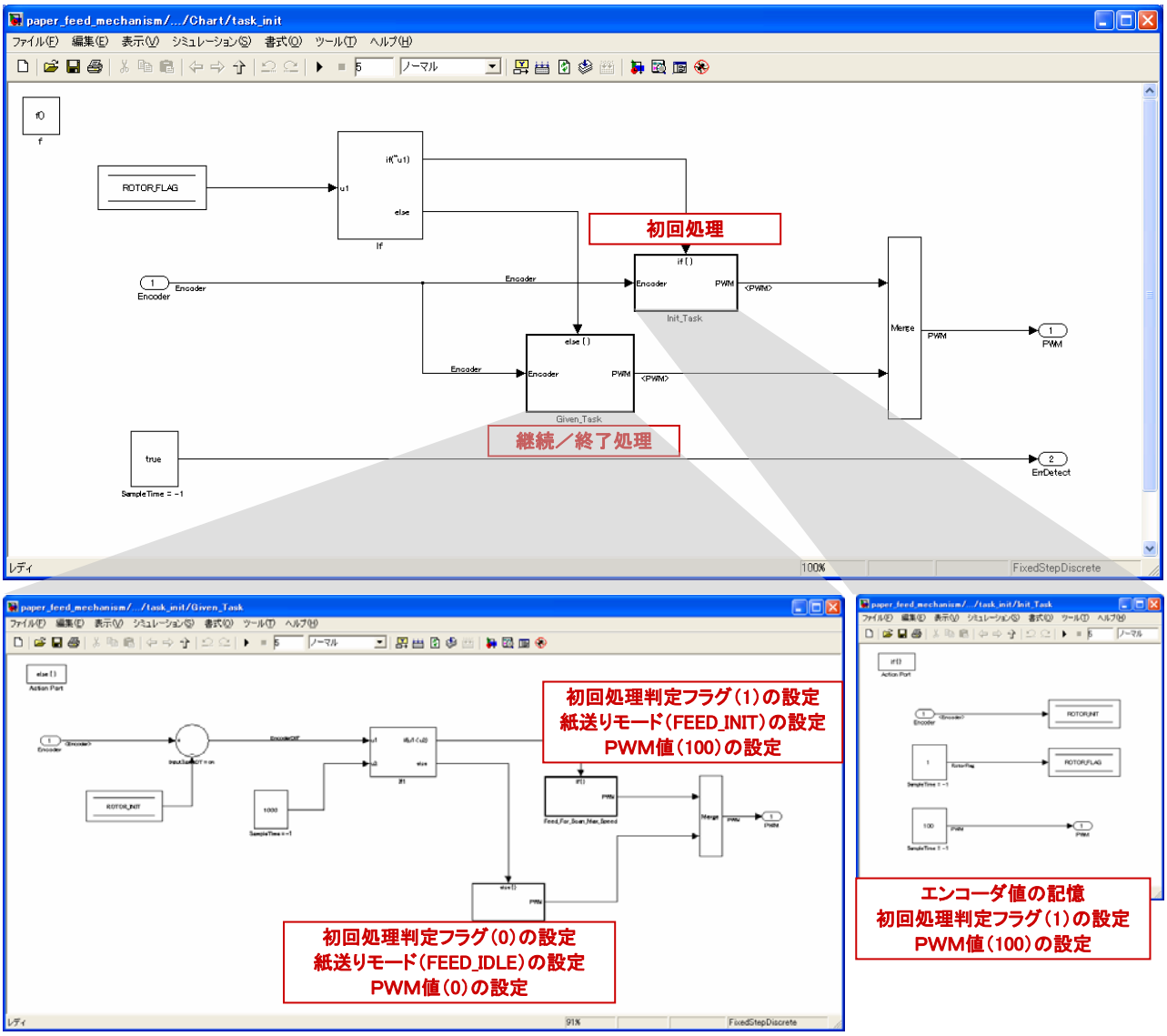


図 6-14　初期化処理（task_init関数）

初期化処理は、初回処理と、継続/終了処理の２つのサブシステムから成ります。これらの機能の切り分けは、Data Store Memory の値（フラグ）により実現しています。

初回処理では以下の処理を実行しています。

- 現在のエンコーダ値の記憶
- 初回処理判定フラグ（1）の設定（次回以降、継続/終了処理に切り替える）
- PWM 値を最高値（100）に設定

初回処理が 1 度だけ実行されると紙送りが開始され、以下の継続処理が実行されます。

- 初回処理判定フラグ（1）の設定
- 紙送りモード（FEED_INIT）の設定
- PWM 値を最高値（100）に設定

さらに、初回処理で記憶した値からのエンコーダ値の増分が目標回転角度（1000 度）に到達すると、終了処理として以下を実行し、紙送りが終了します。

- 初回処理判定フラグ（0）の設定
- 紙送りモード（FEED_IDLE）の設定
- PWM 値を最低値（0）に設定

初期化処理の終了は、紙送りモードの更新（FEED_INIT→FEED_IDLE）により、モード制御モデルに通知されます。この結果をもとに、モード制御モデルは制御モードの切り替えを行います。

紙送り実行処理（紙送りボタン操作）（task_paperfeed）

ユーザが紙送りボタンを押している間に呼び出されます。本関数では以下の処理を実行します。

- PWM 値を最高値（100）に設定
- バックラッシュの方向フラグ（0）の設定
  （バックラッシュが現在、紙送りの方向に発生している事を示す）
- エラー検出の有効/無効を示すフラグ（1）の設定
  （この情報は、信号線により、エラー検出部に伝達されます。）

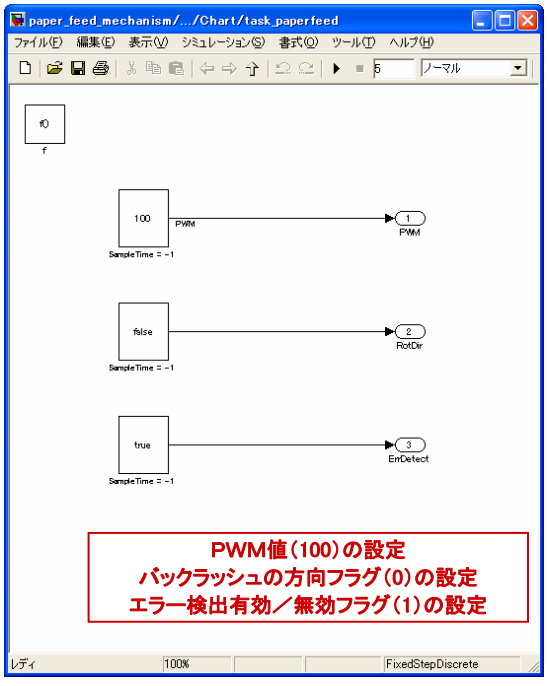


図 6-15　紙送り実行処理（紙送りボタン操作）(task_paperfeed関数)

紙排出実行処理（紙排出ボタン操作）（task_paperexhaust）

ユーザが紙排出ボタンを押している間に呼び出されます。本関数では以下の処理を実行します。

- PWM 値を最高値（-100）に設定
- バックラッシュの方向フラグ（1）の設定
  （バックラッシュが現在、紙排出方向に発生している事を示す）
- エラー検出の有効/無効を示すフラグ（1）の設定
  （この情報は、信号線により、エラー検出部に伝達されます。）

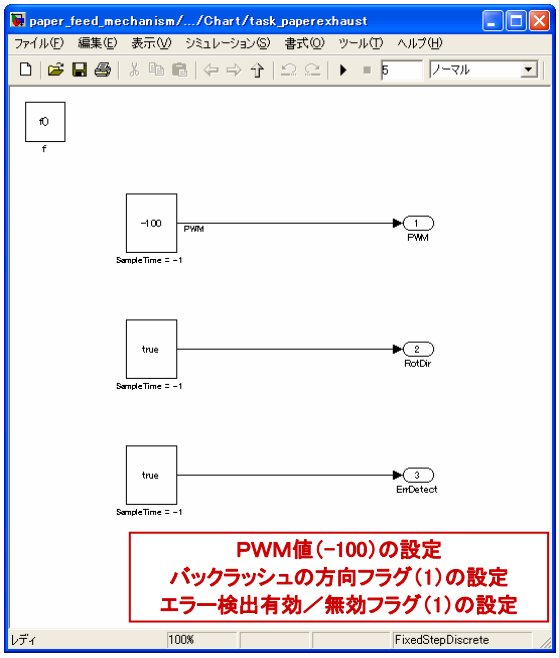


図 6-16　紙排出実行処理（紙排出ボタン操作）（task_paperexhaust関数）

紙送り／排出停止処理（task_stop）

紙送り、紙排出を実行していない時、スキャンヘッド移動時などに呼び出され、紙送り用 DC モータの回転を停止します。本関数では以下の処理を実行します。

- PWM 値を最低値（0）に設定
- エラー検出の有効/無効を示すフラグ（0）の設定
  （この情報は、信号線により、エラー検出部に伝達されます。）

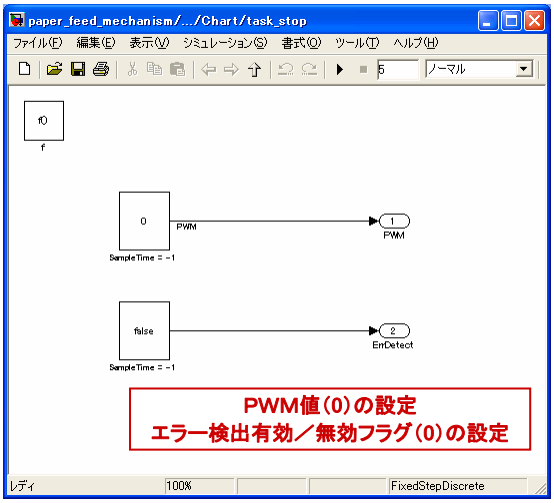


図 6-17　紙送り／排出停止処理(task_stop関数)

紙送り実行処理（スキャン制御実行中）（task_feed_paperfeed）

この関数は、スキャン制御実行中に発生する紙送りの処理を定義します。本関数は、前述の初期化処理（task_init）に類似した機能を持つため、モデリングについても初期化処理（task_init）の設計を一部流用しています。したがって、以下では主に、本関数の初期化処理（task_init）とは異なる仕様を中心に説明します。

初期化処理（task_init）との違いの 1 つは、バックラッシュを意識している点です。バックラッシュの方向フラグ（過去の用紙の進行方向の情報）を関数の引数として受け取り、そのフラグ値に応じてエンコーダの増分値を決定します。例えば、バックラッシュの方向フラグが 0 の場合には、直前に紙送り操作が行われバックラッシュの影響がないものと判断し、画像データ 1 行分の紙送りに必要な回転角度を 102 度とします。逆にこのフラグが 1 の場合には、直前に紙排出が行われていると考え、バックラッシュの影響を考慮し、必要な回転角度を、オフセットを加えた 102+50 = 152 度としています。

また、紙送り停止時に惰性によるモーター回転の影響を受けないよう、エンコーダ値に対応する PWM 値の出力を LookUp Table にて定義しています。LookUp Table ブロックのテーブルデータの算出には、参考文献[3]NXT GamePad のデータロギング機能による実機での計測を用いました。NXT GamePad は、Bluetooth 通信用 PC ユーティリティソフトウェアによるデータロギング機能を提供しています。

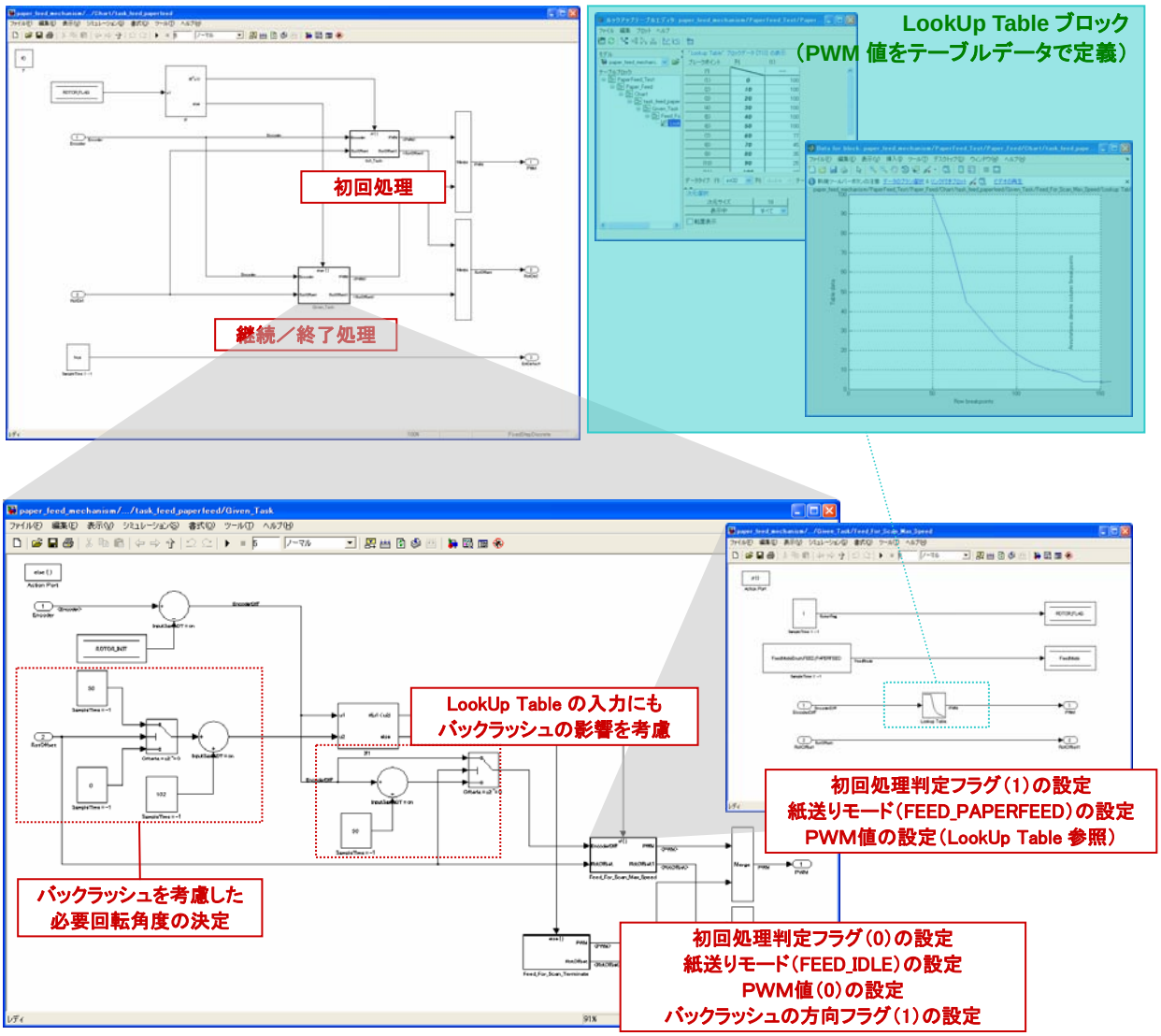


図 6-18　紙送り実行処理（スキャン制御実行中）（task_feed_paperfeed関数）

紙送り制御モデルでは、127 行のスキャンを行うため、スキャン制御実行中の紙送りは、127 回要求されます。1 行飛ばしになるように紙送りを行い正方形の画像データをスキャンします。結果、図 6-19 のように列方向にスキャンデータが欠損した画像になります。NXT Viewer では、この欠損画像を復元する画像補間を行います。

図 6-19　スキャン結果の一部を拡大(イメージ)

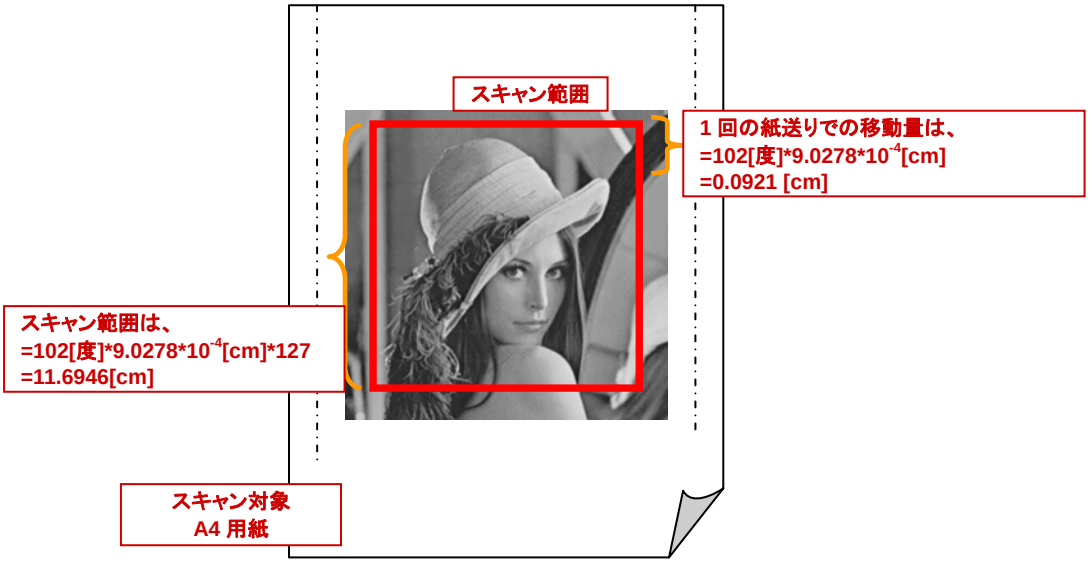


図 6-20　スキャン範囲（紙送り方向）

また、3章NXT Scannerの機構で求めた紙送り制御モータ回転角度 1 度あたりの移動距離 9.0278×10^{-4}[cm]を用いると図 6-20 の各値となります。従いまして、紙送り機構の解像度は、約 27dpi[dots per inch]と言えます。

エラー検出部

図 6-21 はエラー検出部です。エラー検出部のサブシステムは、制御フロー部から入力されるイネーブル信号（エラー判定を行うか否かを示すフラグ）により駆動されます。サブシステムがイネーブル状態の場合、10 ステップ毎にエンコーダ値の保持、前回値との比較を行い、2 つのエンコーダ値が等しい場合には、エラー状態と判断し、エラーコードの更新（NO_ERROR→ERROR_002）により、モード制御モデルに紙送り/紙排出機構異常が通知されます。

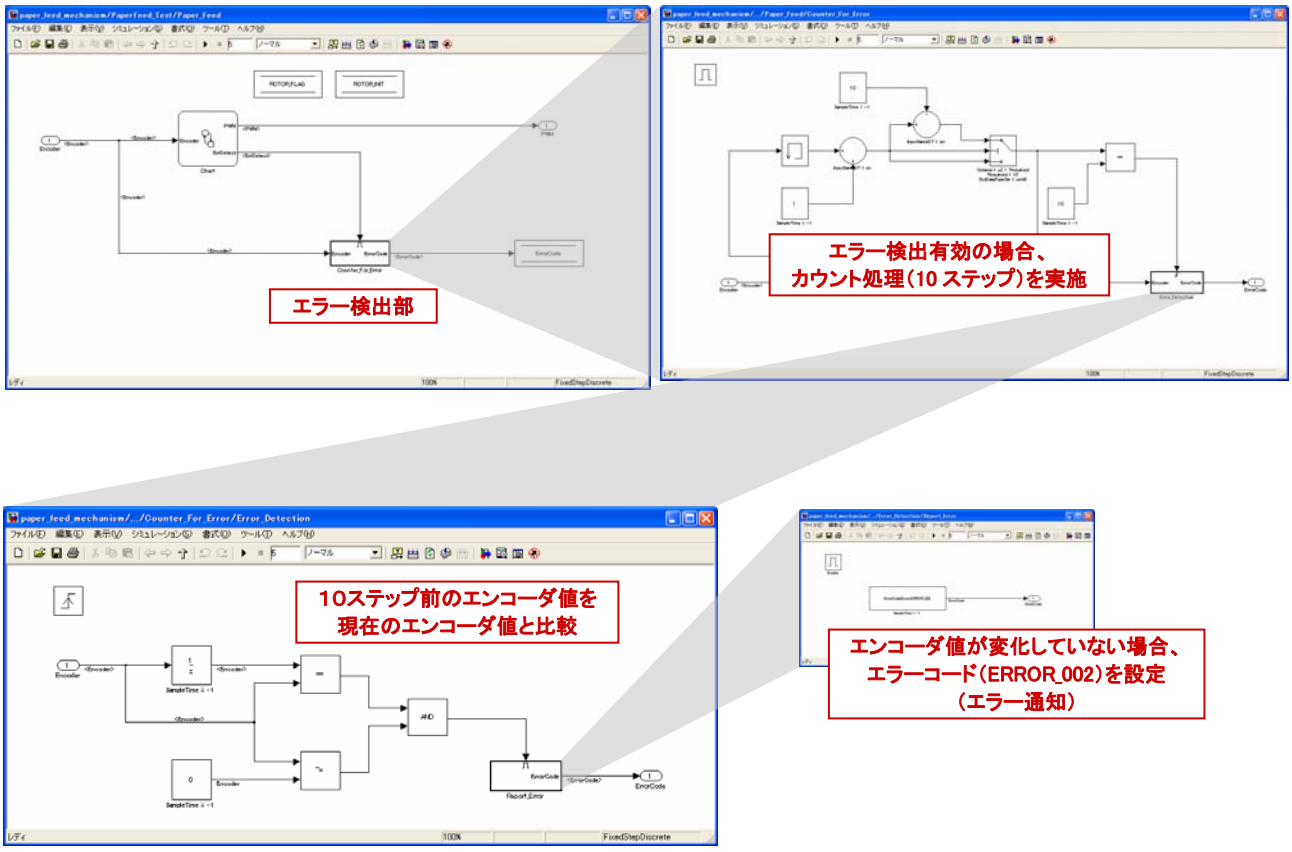


図 6-21　エラー検出部（サブシステム）

6.5　スキャン制御モデル

scanner_head.mdl はスキャン制御システムの機能モデルです。図 6-22 はスキャン制御モデルのトップレイヤとライブラリ登録しているサブシステムの関係を表しています。

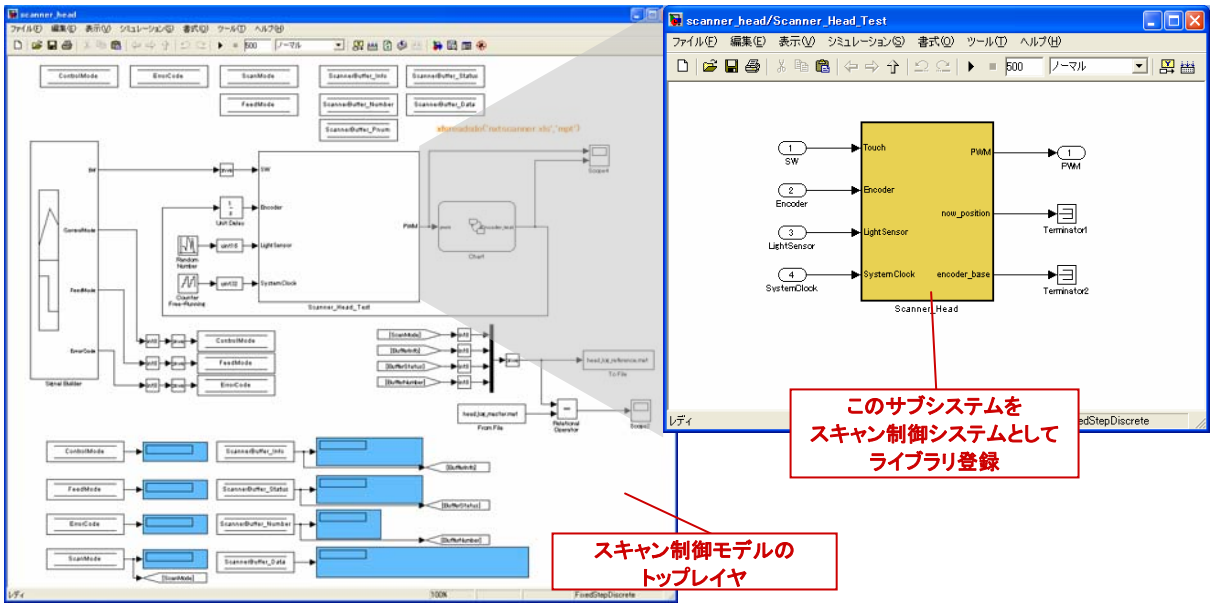


図 6-22　スキャン制御モデルとライブラリの関係

他の機能モデルのトップレイヤと同様に、トップレイヤには検証用の入力信号、出力信号のモニタを配置しており、シミュレーションを実行することで検証を行うことが出来ます。

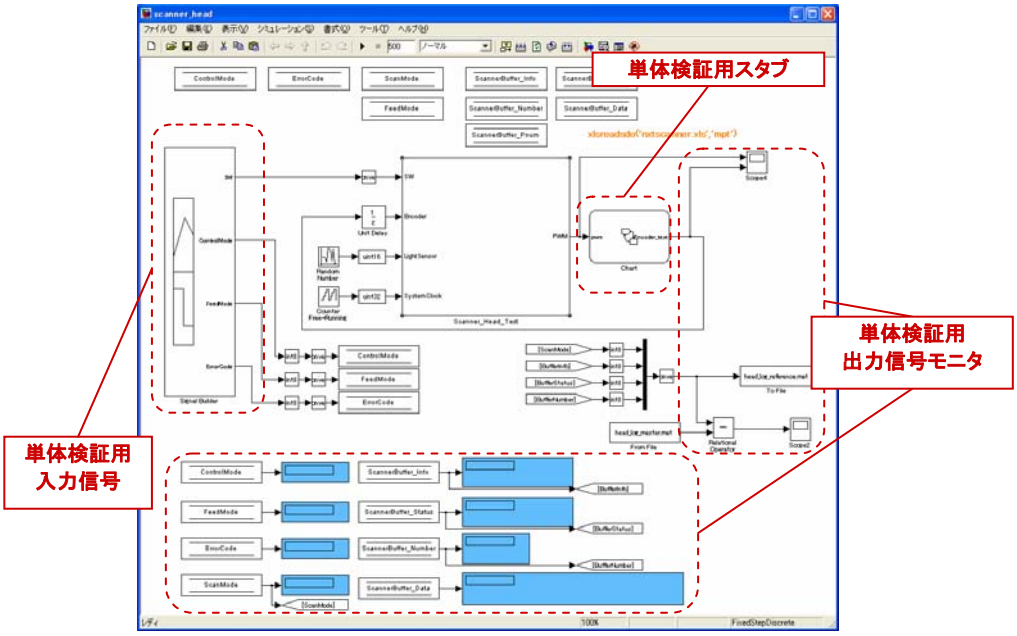


図 6-23　スキャン制御モデルのトップレイヤ

アルゴリズム実装

スキャン制御モデルは、Stateflow のみで構成されています。図 6-24 の左側はステートチャート形式で設計しています。右側は、ステートチャートから呼び出されるグラフィカル関数群です。グラフィカル関数内はフローチャート形式で設計しています。各グラフィカル関数の概要を表 6-2 にまとめてあります。

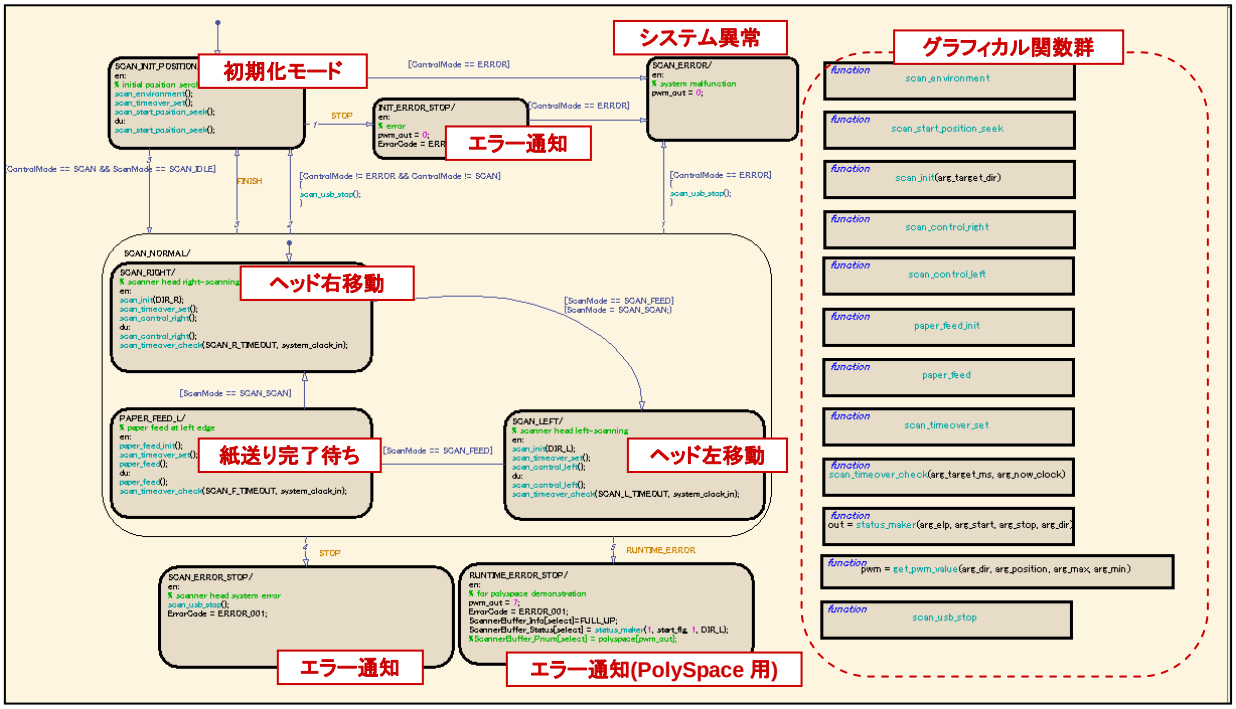


図 6-24　スキャン制御モデルの主アルゴリズム

表 6-2　スキャン制御モデルのグラフィカル関数群

グラフィカル関数名	引数	戻り値	概要
scan_environment	-	-	スキャナヘッド初期移動時の初期設定関数
scan_start_position_seek	-	-	スキャナヘッド初期移動用アルゴリズム関数
scan_init	目標進行方向	-	スキャン開始時の初期設定関数
scan_control_right	-	-	スキャナヘッド右移動用アルゴリズム関数
scan_control_left	-	-	スキャナヘッド左移動用アルゴリズム関数
paper_feed_init	-	-	スキャン紙送り完了待ち初期設定関数
paper_feed	-	-	スキャン紙送り完了待ち関数
scan_timeover_set	-	-	制限時間の監視を開始した時間を設定する関数
scan_timeover_check	指定制限時間 現在時間	-	制限時間を越えたかどうかを監視する関数
status_maker	ELP 情報 START 情報 STOP 情報 DIR 情報	送信ステータス	USB 送信用データのステータスパケット作成関数
get_pwm_value	進行方向 現在位置 目標最高速 PWM 目標最低速 PWM	今回 PWM 値	スキャナヘッド移動用今回 PWM 値の決定関数
scan_usb_stop	-	-	スキャンを強制終了する場合（ユーザによる中断要求、スキャンのエラー終了）の USB 送信最終データ作成関数

スキャンデータの取り込み

スキャンデータの取り込みは、スキャナヘッド右向き移動時に行うため、scan_control_right グラフィカル関数内で実施しています。図 6-25 はスキャナヘッド右向き時のアルゴリズムを実装したグラフィカル関数 scan_control_right の stateflow モデルです。

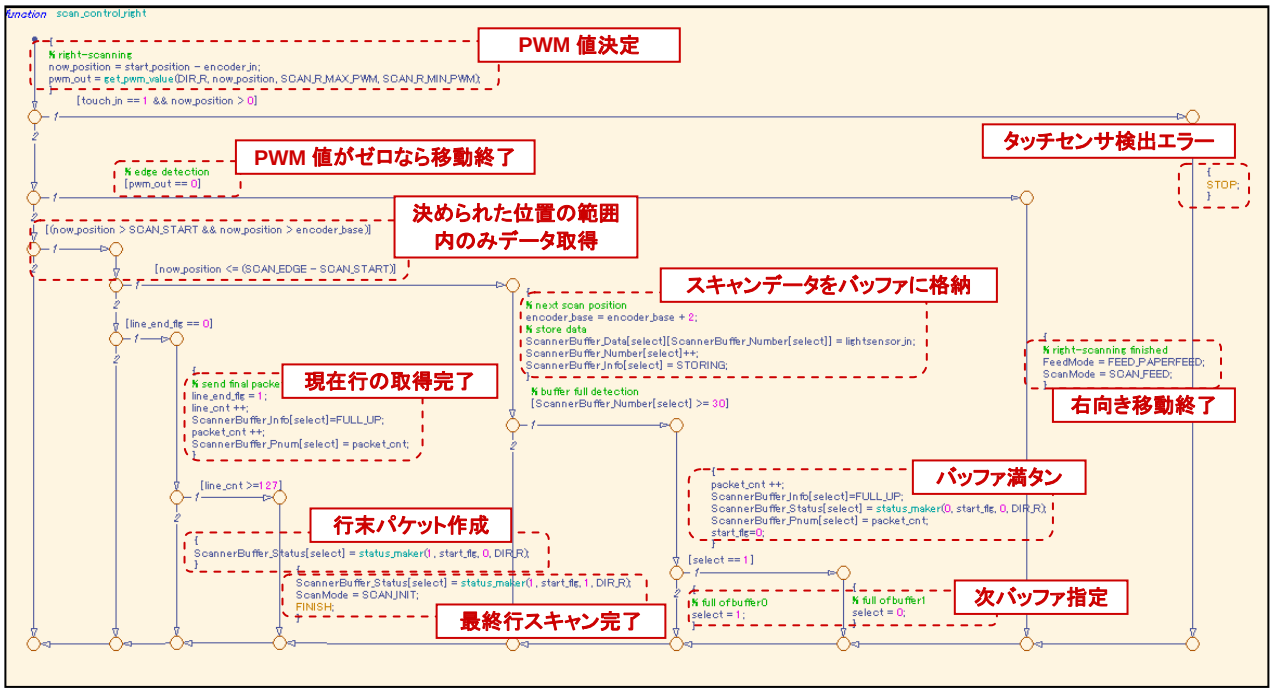


図 6-25　スキャナヘッド右向き移動アルゴリズム

用紙に対してスキャンする範囲は、図 6-26 の赤枠内となります。スキャナヘッド機構の稼動範囲はSCAN_EDGEで定義しています。単位は、ロータリエンコーダ値(度)です。このSCAN_EDGE分のスキャナヘッドの移動に対して両端SCAN_START分の範囲を除いた範囲を有効スキャン範囲としています。また、3章NXT Scannerの機構で求めたスキャナヘッド制御モータ回転角度 1 度あたりの移動距離 0.0215 [cm]を用いると図 6-26 の各値となります。従いまして、スキャナヘッド機構の解像度は、約 118dpi [dots per inch]と言えます。

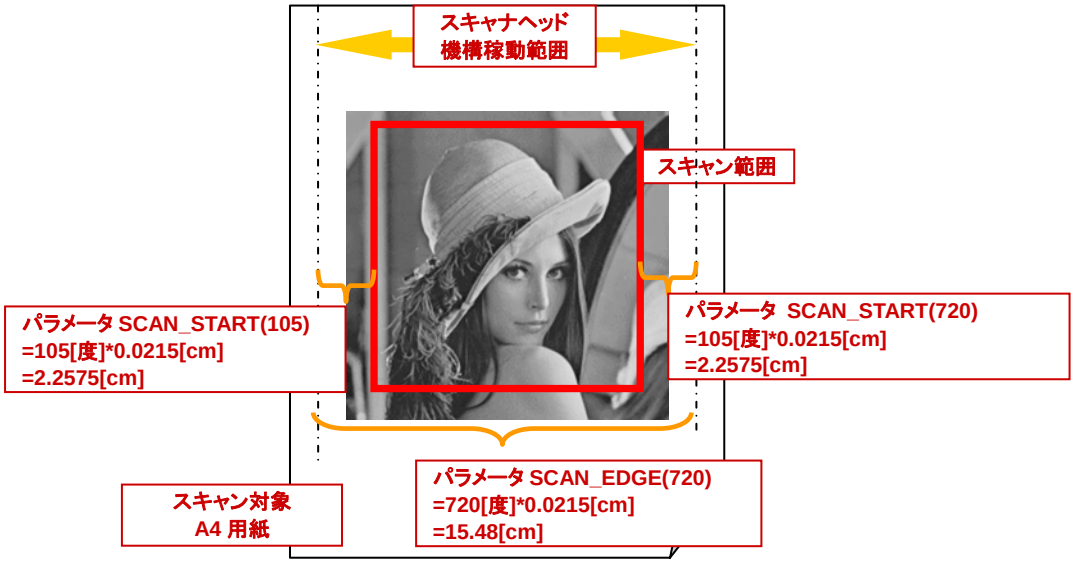


図 6-26　スキャン範囲（横方向）

スキャン位置の整列

1 行に 255 回のスキャンを行う時に重要となるのが、スキャンのタイミングです。スキャンのタイミングが行毎にばらついてしまうと、スキャン画像として列方向に整列できていない歪んだ画像となってしまいます。統合モデルでは、スキャン制御モデルを周期タスクに実装しますが、周期タスクにのみ依存したタイミングでスキャンを行うと、スキャナヘッド制御モータの移動量に行毎のばらつきがあるためにスキャン位置がばらついてしまいます。そのため NXT Scanner では、スキャナヘッド制御モータのロータリエンコーダ値をフィードバックして目標位置付近で定期的にスキャンする制御を設計しています。図 6-27 の赤枠内が次回の目標位置を設定している箇所となります。フィードバックするロータリエンコーダ値が 2 度増加する毎にスキャンを行っています。

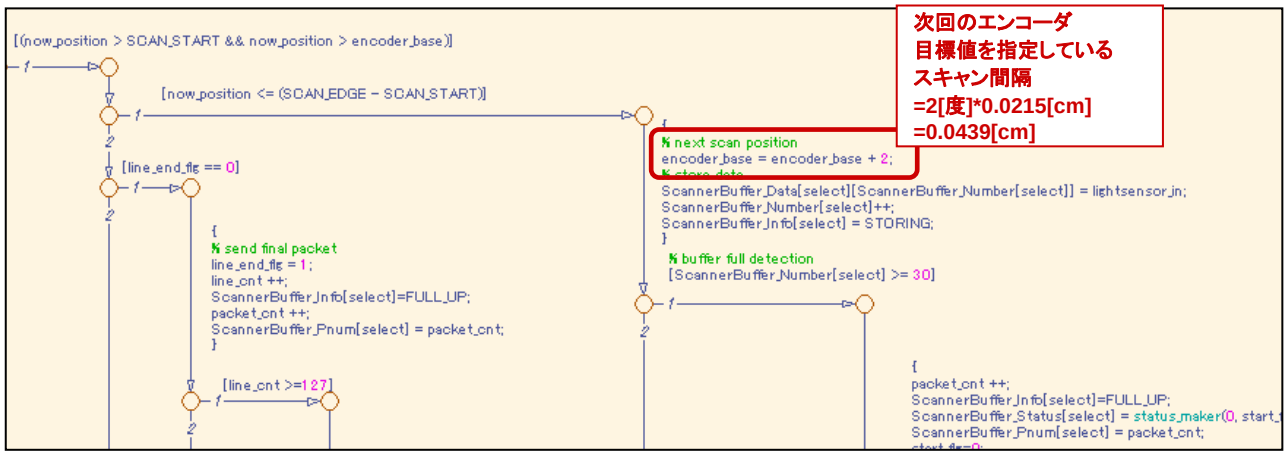


図 6-27　スキャン位置の制御（scan_control_rightグラフィカル関数内）

スキャンヘッドの停止位置制御

スキャン位置を整列させる制御を実装しても、毎回のスキャン動作でスキャナヘッド折り返し位置が慣性の影響でずれてしまうとスキャン画像が歪んでしまいます。従いまして、スキャナヘッドのロータリエンコーダ値が決められた角度(パラメータ SCAN_EDGE)移動したところで確実に停止させる制御が必要になります。

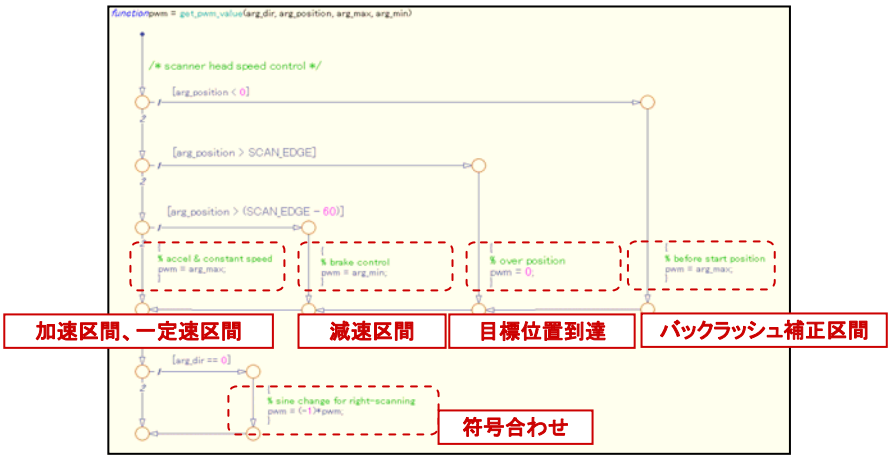


図 6-28　スキャナヘッドの位置制御（get_pwm_valueグラフィカル関数）

目標最低速 PWM を get_pwm_value グラフィカル関数に与えていますが、この値は、スキャナヘッド機構に影響する摩擦力やケーブルの負荷やモーターの駆動軸の位置などを考慮して事前に実験にてスキャナヘッド機構を動かす最低 PWM 値を確認しています。同様に目標位置到達時に、スキャナヘッド機構の慣性の影響を受けないブレーキ開始位置も事前の実験で確認しています。従いまして、仕様変更などにより最高速 PWM 値が変更されるときには、十分注意が必要です。

ギヤのバックラッシュ回避

スキャナヘッド機構を往復運動させる時に、ギヤのバックラッシュの影響を受けて現在位置と目標位置にずれが生じ、その結果スキャン画像が歪んでしまいます。この回避策として図 6-29 のようにスキャナヘッド移動開始持の初期化関数 scan_init グラフィカル関数にてスキャン開始時の現在位置にオフセット(パラメータ BACKLASH_ADJUST)を与えています。

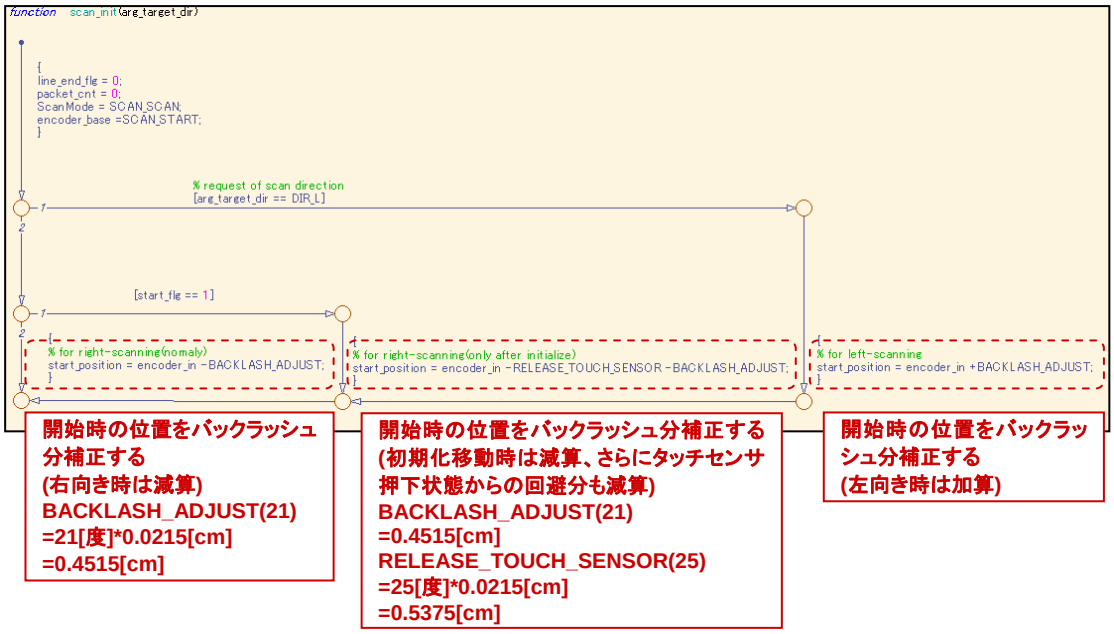


図 6-29　バックラッシュ回避制御(scan_initグラフィカル関数)

スキャン時間の短縮

スキャン時間を短縮するために、スキャナヘッドの左向き移動と、スキャン時の紙送り制御を平行して行っています。

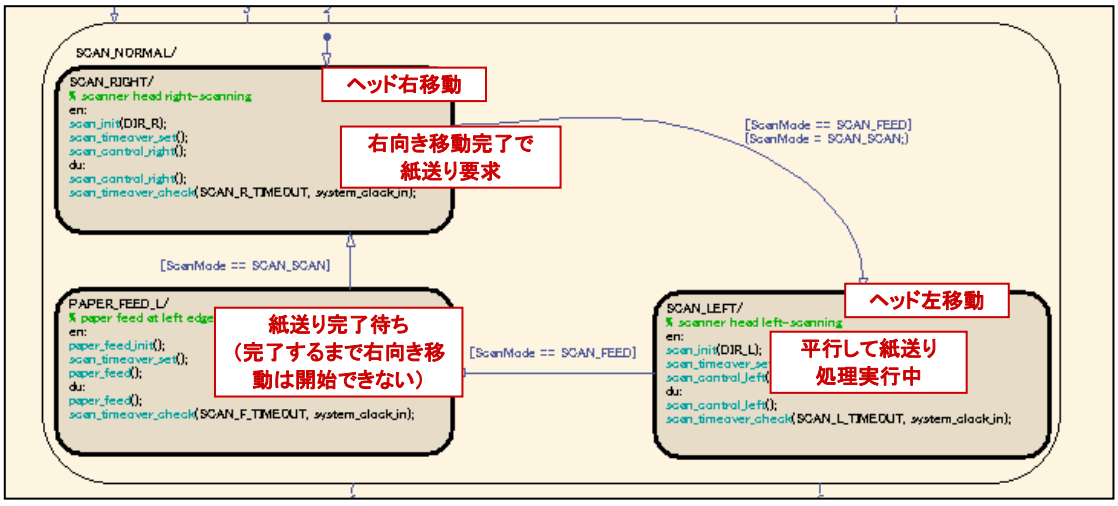


図 6-30　スキャナヘッド左向き移動と紙送り制御

6.6 USB 通信制御モデル

data_communication.mdl は USB 通信制御システムの機能モデルです。図 6-31 は USB 通信制御モデルのトップレイヤとライブラリ登録しているサブシステムの関係を表しています。

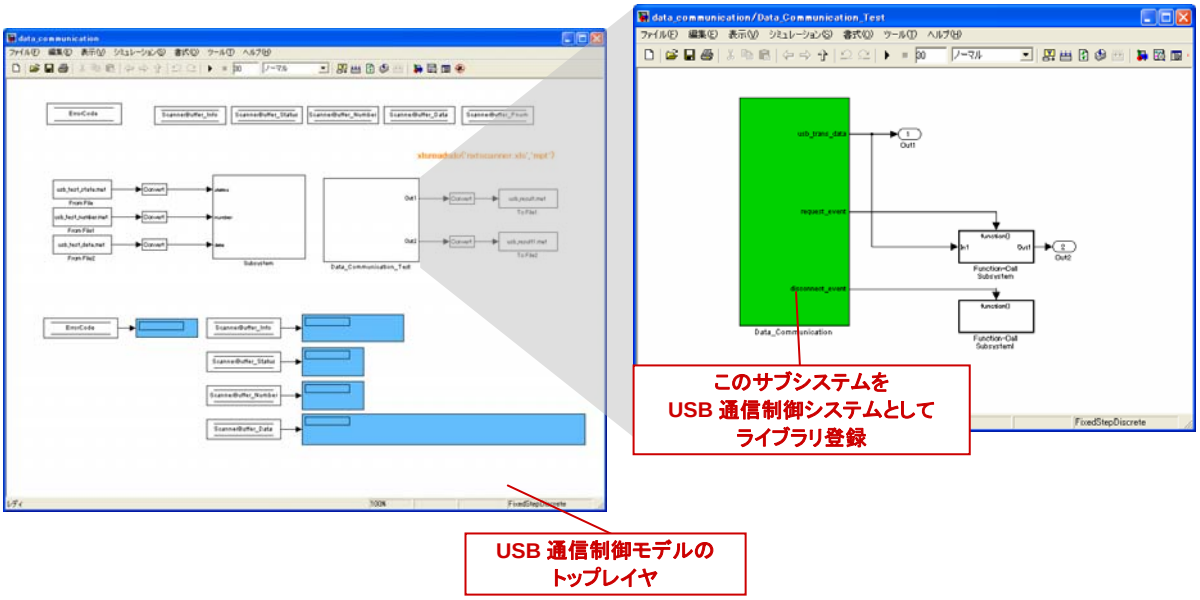


図 6-31　USB通信制御モデルとライブラリの関係

他の機能モデルのトップレイヤと同様に、トップレイヤには検証用の入力信号、出力信号のモニタを配置しており、シミュレーションを実行することで検証を行うことが出来ます。

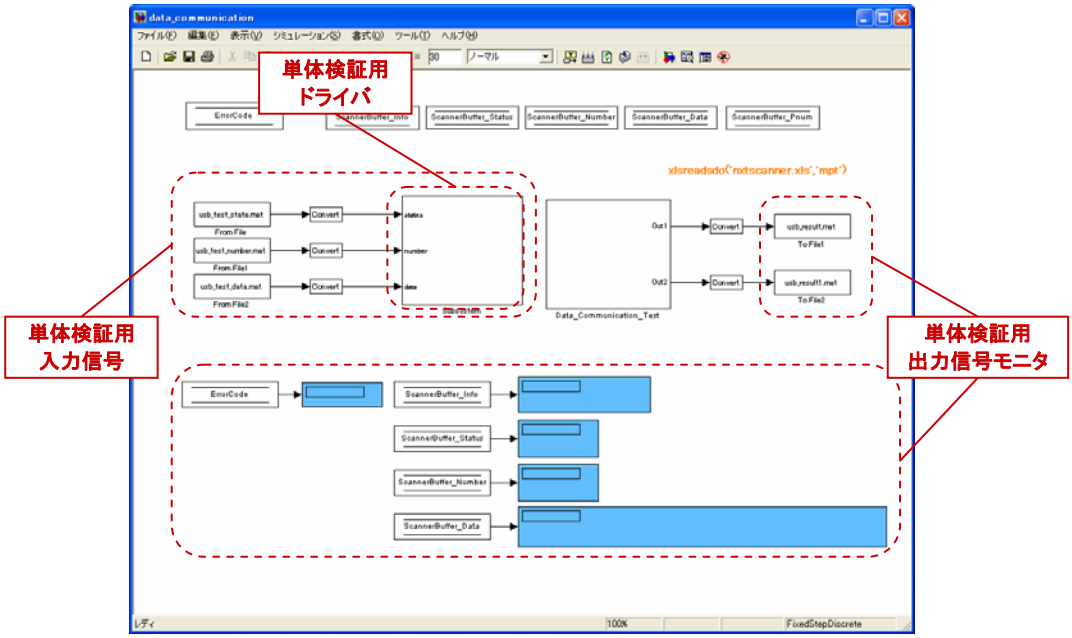


図 6-32　USB通信制御モデルのトップレイヤ

アルゴリズム実装

USB 通信制御モデルは図 6-33 で示すように、ライブラリ登録したサブシステムに実装しています。USB の送信アルゴリズムは、図 6-34 で示すように、Stateflow にてモデリングしています。USB 送信データは 64 バイトと定められているため、最終的に、ステータス（2 バイト）、データ数（2 バイト）、送信データ（60 バイト）を合成し 64 バイトデータを生成しています。

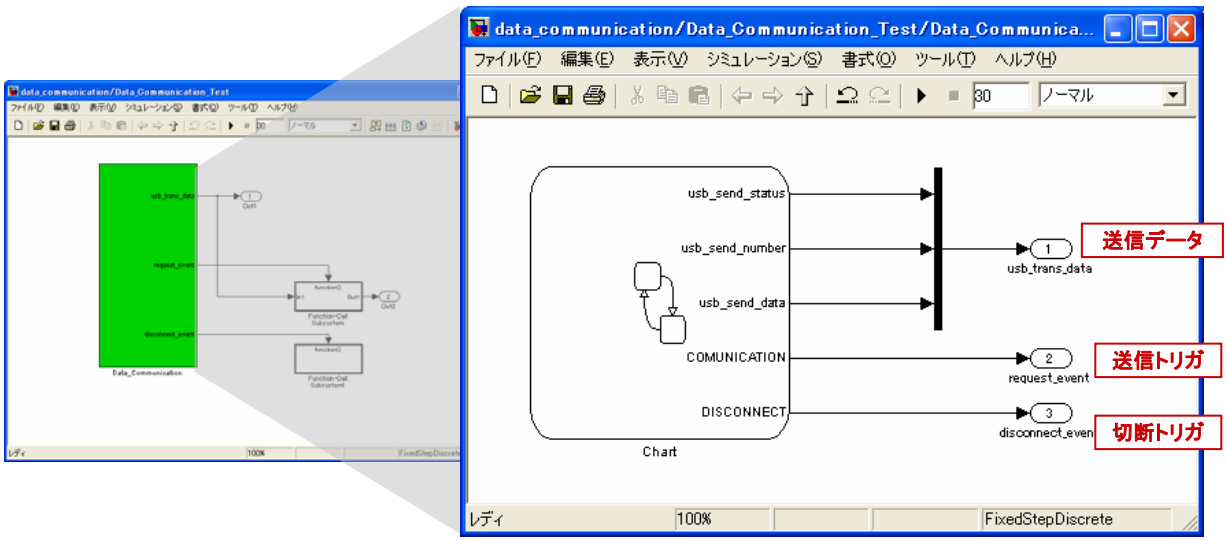


図 6-33　USB通信制御モデルとライブラリの関係

全データのスキャン終了時にホスト PC との USB 接続を切断するために、USB 通信モデルではスキャン制御モデルからの最終スキャンバッファを監視しています。最終スキャンバッファを送信後、1 サンプリング周期後に USB を切断します。1 サンプリング周期の遅れを実現するために State（CLOSE）でモデリングしています。

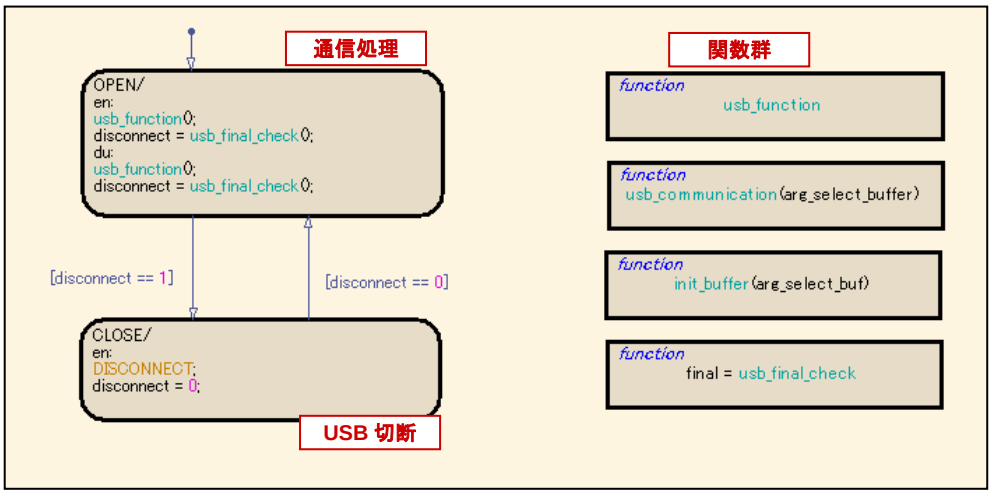


図 6-34　USB通信制御モデルの主アルゴリズム

USB 通信処理では、スキャン制御モデルが取得し作成したバッファを監視し、送信可能な状態になったときだけ送信を行います。送信可能な状態は、スキャンバッファ情報変数（ScannerBuffer_Info）を監視し、FULL_UP （満タン）になった時としています。ダブルバッファ構造であるため、通常スキャン時に、両方のバッファが満タンになってしまわないように実装タスク周期を検討する必要があります。ユーザーによるスキャン中断要求時には、タイミングによっては全バッファが送信可能状態になる場合がありえるため、必ず前回送信したバッファと異なるバッファから送信するようにしています。

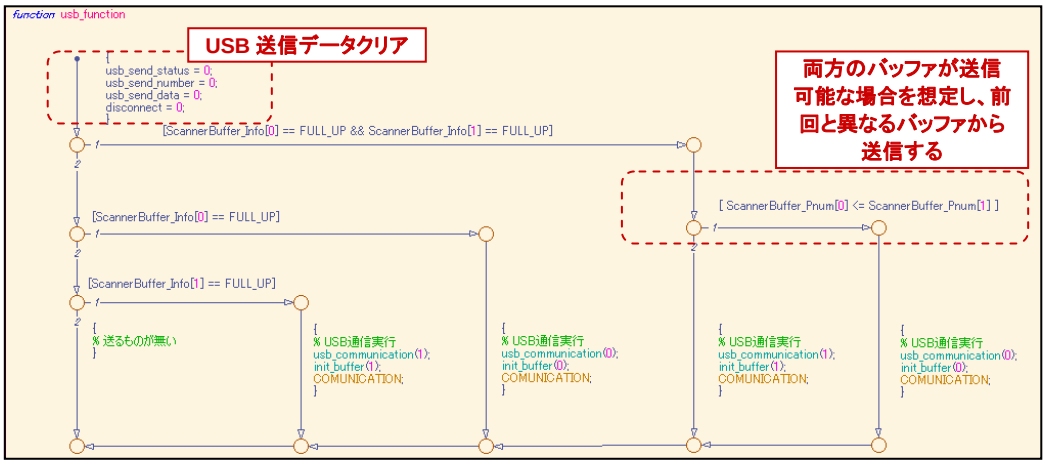


図 6-35　USB通信制御モデルの主アルゴリズム

7　機能モデルのシミュレーション

NXT Scanner 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。シミュレーション実施には、setup_nxtscanner.m を実行し、NXT Scanner の環境設定フォルダを MATLAB の PATH に追加してください。

7.1　シミュレーション方法

各機能モデルでは図 7-1 のようにモデルカバレージが 100%になるテスト信号を用意しています。機能モデルの単体検証には、このテスト信号を用いて検証を実施します。

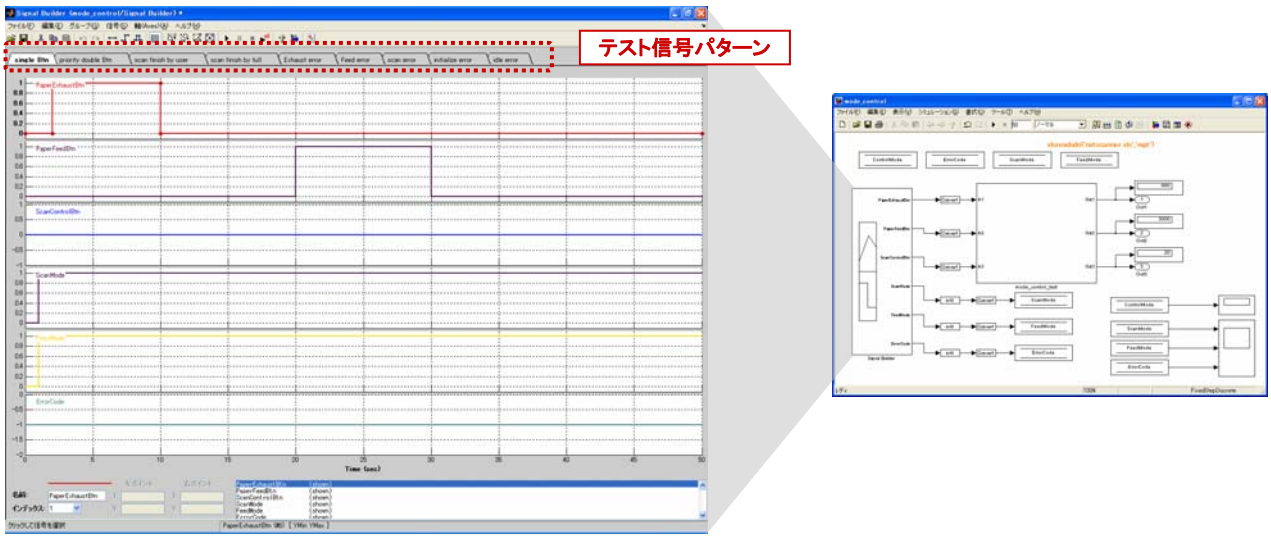


図 7-1　動作モードの切り替え

補足

R2008b の新機能である Enumerated Type や Stateflow の Simulink Function を使用した場合、Simulink® Design Verifier™はツールの機能制限により使用することが出来ません。この制限があるために NXT Scanner では、手動でカバレージ 100%のテスト信号を作成し、そのテスト信号で意図した状態遷移をしているかを目視で確認する方法で検証しています。

7.2 検証ツールの機能紹介

Simulink、Stateflow のモデルを検証するツールとして Simulink Verification and Validation と Simulink Design Verifier があります。

Simulink Verification and Validation

Simulink Verification and Validation は、以下の機能を持ちます。

- サブシステム単位でカバレッジ対象の指定が可能
- モデル中に、通過パス・未達パスの色別表示が可能
- 複数テストによるカバレッジの蓄積が可能

要求仕様に沿った設計とテストケースの開発およびテストカバレージの測定を行うツールです。開発の初期段階で、設計の不具合、不適切な要求仕様、不完全なテスト、および不要な設計要素を検出します。要求仕様書を設計モデル、コンポーネントテスト、および生成コードにトレースすることが可能です。また、モデルカバレージやモデリングスタンダードのチェックを通し、設計やテストの検証を行うことができます。

Simulink Design Verifier

Simulink Design Verifier は、以下の機能を持ちます。

- テストケースの自動生成
- プロパティ検証

Simulink Design Verifier は、形式手法によるテストケースの自動生成とモデルプロパティ検証を行うことが出来るツールです。

Simulink Verification and Validation のカバレッジ測定機能を使い、モデルのカバレッジ率 100%を実現するテストハーネスを自動生成する機能を提供しています。モデルのフルパス検証および自動生成テストハーネスを使って、モデルと生成コード（S-Function 化したもの）の等価性を検証することができます。

プロパティ検証は、設定した条件があらゆる入力信号に対して満たされていることを検証する機能です。具体的な使用例としては、図 7-2 のように、検証対象のサブシステム（ここではライブラリ登録したモード制御モデルのサブシステム）を検証するためのサブシステムを接続させます。図 7-2 では、R2008b では機能が対応していないため、mode_control_verification_lib.mdl に登録した、検証用サブシステムと空サブシステムを切り替えることが出来るようにモデリングしています。

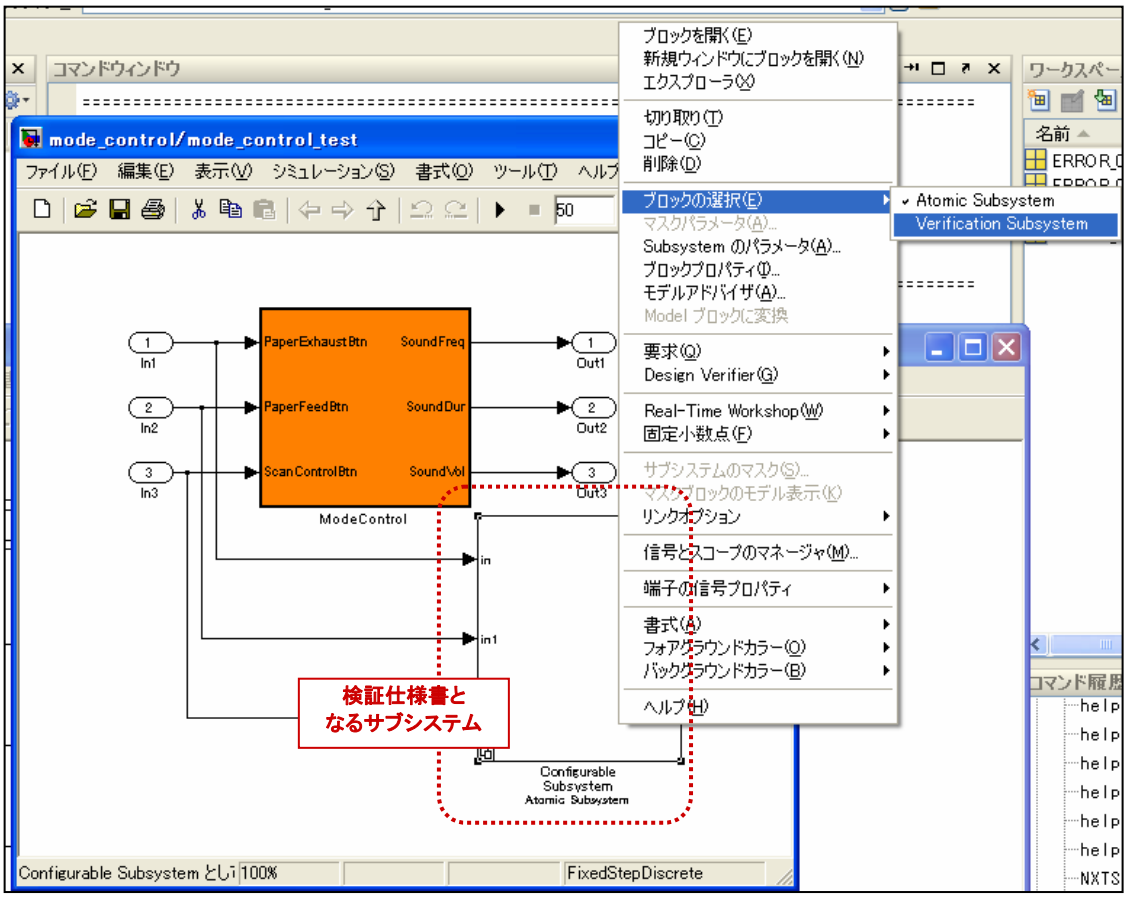


図 7-2　検証仕様書サブシステムの切り替え

検証仕様サブシステムのモデリングには様々な方法がありますが、参考例として真理値表を用いた検証仕様サブシステムを図 7-3 に示します。Simulink Design Verifier は、この検証仕様サブシステムに対する機能モデルの妥当性検証をおこないます。検証仕様モデルを満たさないケース（例、異常状態、出力データ範囲の超過）が見つかった場合には、Simulink Design Verifier は反例テストパターンを出力します。

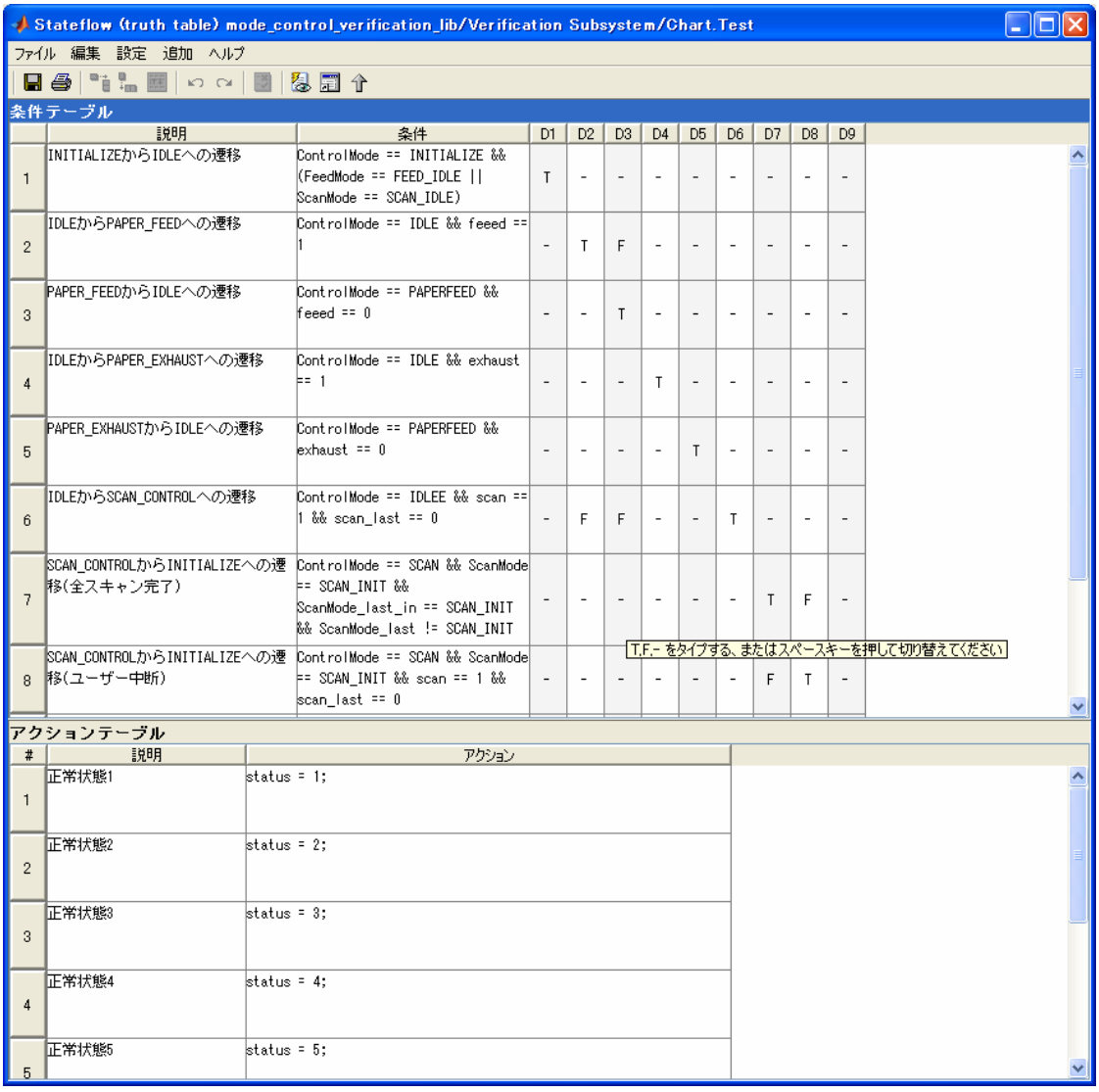


図 7-3　検証仕様モデルの例

8　NXT Scanner 統合モデル

各機能モデルを統合したモデルである nxtscanner_ctrl.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

制御プログラムを表 8-1 の 5 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	2 [ms] 周期	スキャン制御システム
task_ts2	10 [ms] 周期	紙送り制御システム
task_ts3	20 [ms] 周期	モード制御システム
task_ts4	60 [ms] 周期	USB 通信制御システム

8.2 モデル概要

nxtscanner_ctrl.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。

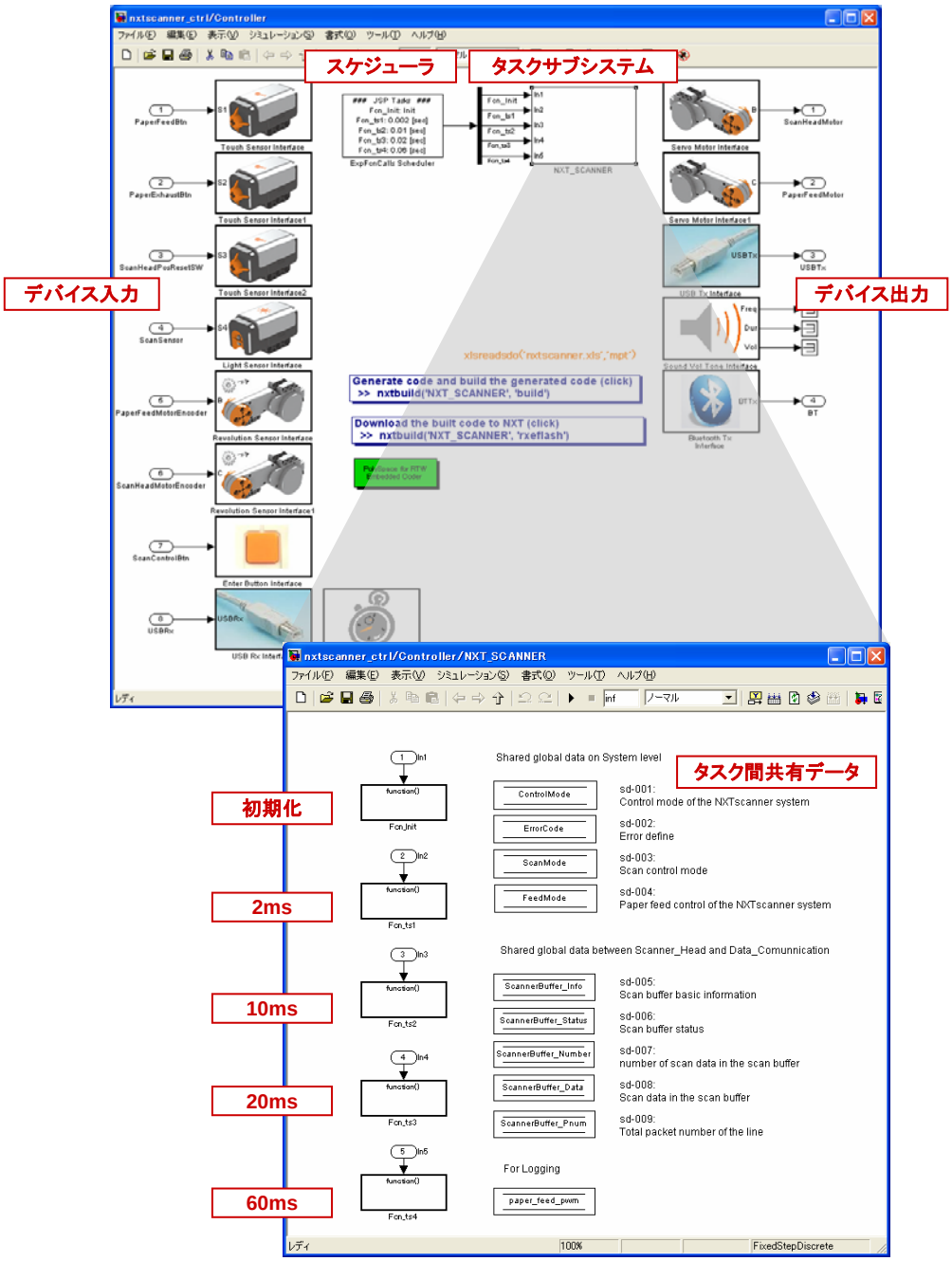


図 8-1　nxtscanner_ctrl.mdl

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。

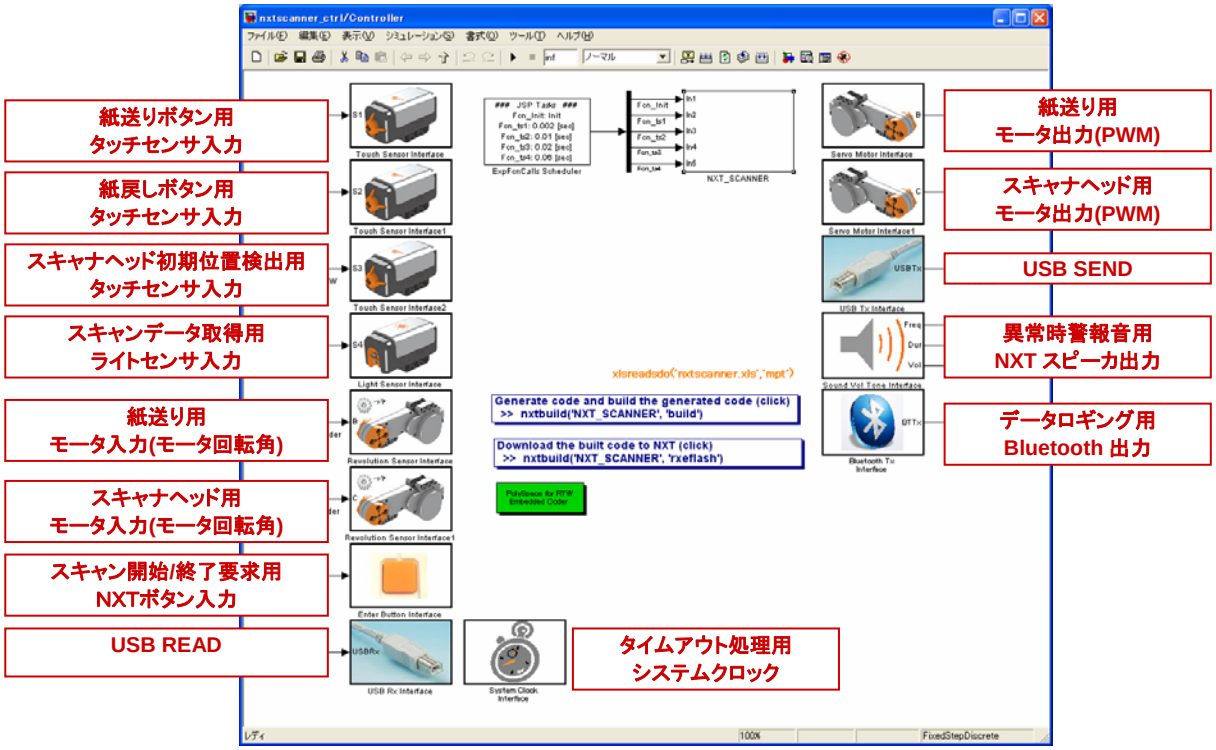


図 8-2　入出力インタフェース

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。プラットフォームは OSEK か JSP から選択することが可能です。

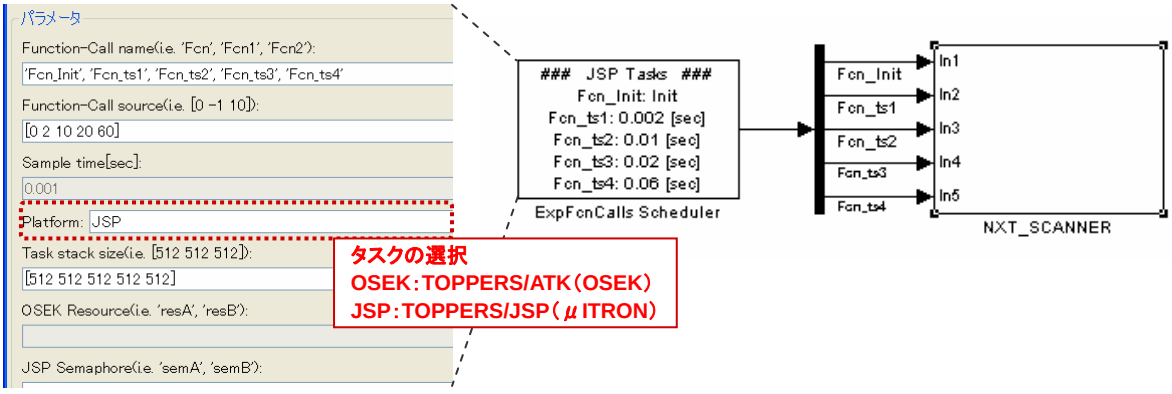


図 8-3　スケジューラ＆タスク

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 8-4　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 ブロックを使用しています。

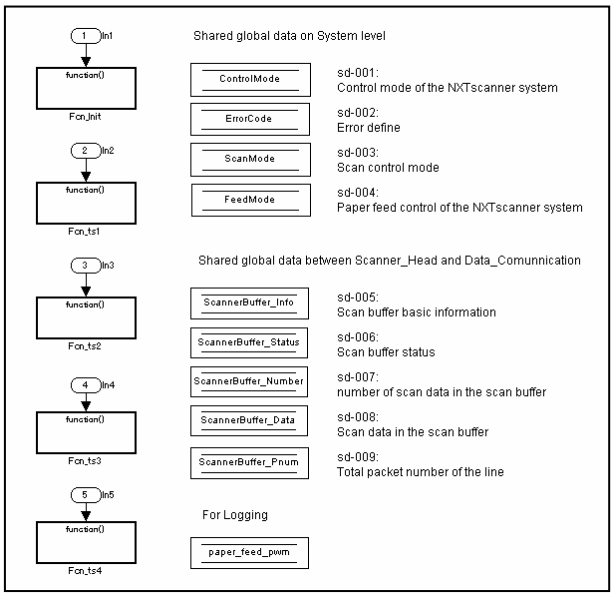


図 8-5　タスク間共有データ

8.3 初期化タスク：task_init

初期値設定を行うタスクです。使用する 2 つのモータの PWM をゼロに設定します。

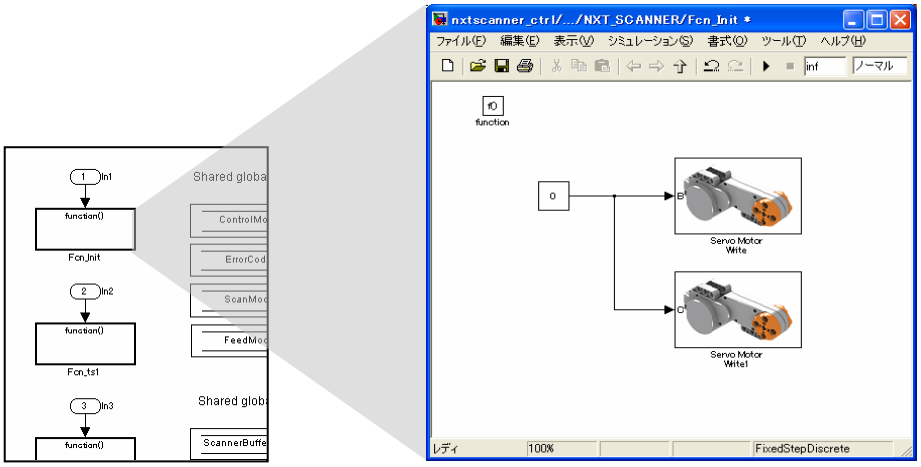


図 8-6 task_init

8.4 2ms タスク：task_ts1

スキャン制御モデルを含むタスクです。Bluetooth でデータロギングを行うことが出来るようにモデリングしています。

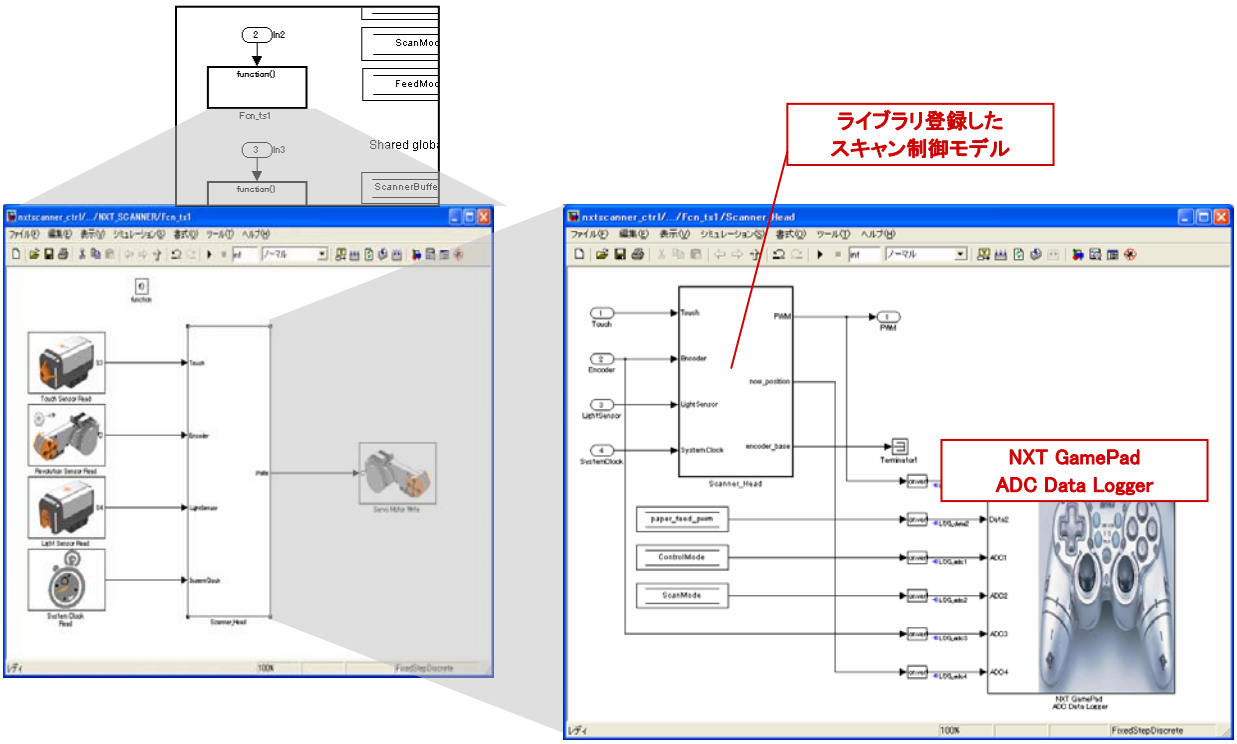


図 8-7 task_ts1

8.5 10ms タスク：task_ts2

紙送り制御モデルを含むタスクです。

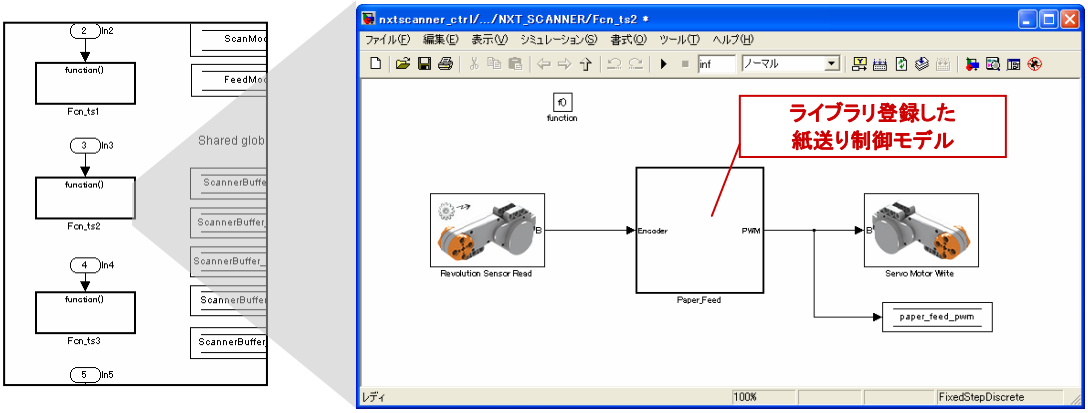


図 8-8　task_ts2

8.6 20ms タスク：task_ts3

モード制御モデルを含むタスクです。

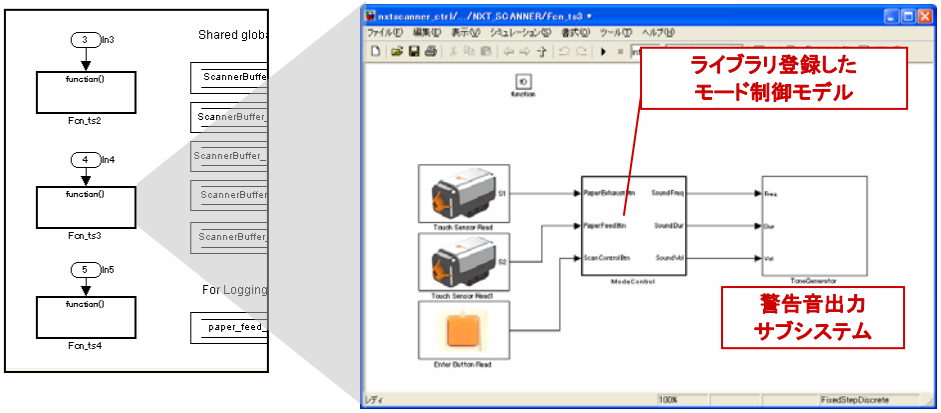


図 8-9　task_ts3

8.7　60ms タスク：task_ts4

USB 通信制御モデルを含むタスクです。

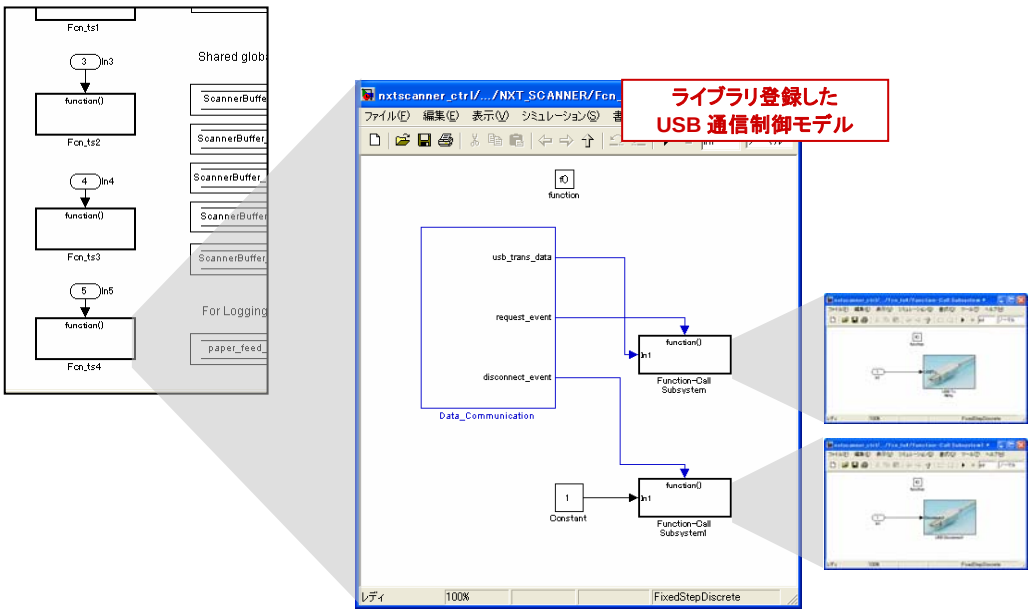


図 8-10　task_ts4

8.8 チューニングパラメータ

NXT Scanner の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
BACKLASH_ADJUST	バックラッシュ対策用補正
RELEASE_TOUCH_SENSOR	スキャン開始位置補正
SCAN_EDGE	スキャナヘッド機構の稼動範囲
SCAN_START	スキャン開始位置
SCAN_INITIALIZE_PWM	スキャナヘッド初期化移動時の PWM 値
SCAN_INITIALIZE_TIMEOUT	スキャナヘッド初期化移動タイムアウト時間[ms]
SCAN_R_TIMEOUT	スキャナヘッド右向き移動時のタイムアウト時間[ms]
SCAN_L_TIMEOUT	スキャナヘッド左向き移動時のタイムアウト時間[ms]
SCAN_F_TIMEOUT	スキャン中、紙送り完了待ちタイムアウト時間[ms]
SCAN_R_MAX_PWM	スキャナヘッド右向き移動時の最高速 PWM 値
SCAN_R_MIN_PWM	スキャナヘッド右向き移動時の最低速 PWM 値
SCAN_L_MAX_PWM	スキャナヘッド左向き移動時の最高速 PWM 値
SCAN_L_MIN_PWM	スキャナヘッド左向き移動時の最低速 PWM 値

9　コード生成と実装

nxtscanner_ctrl.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。

9.1　実装環境

表 9-1 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 9-1　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）
	RAM	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 / USB
ソフトウェア	RTOS	nxtOSEK / nxtJSP
	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）

9.2 コード生成・実装手順

図 9-1 に示す通り、nxtscanner_ctrl.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. [**xlsreadsdo(`nxtscanner.xls`,'mpt`)**] 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）が反映されます。Simulink データオブジェクトの詳細については参考文献[3]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. [**Download the built code to NXT**]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。

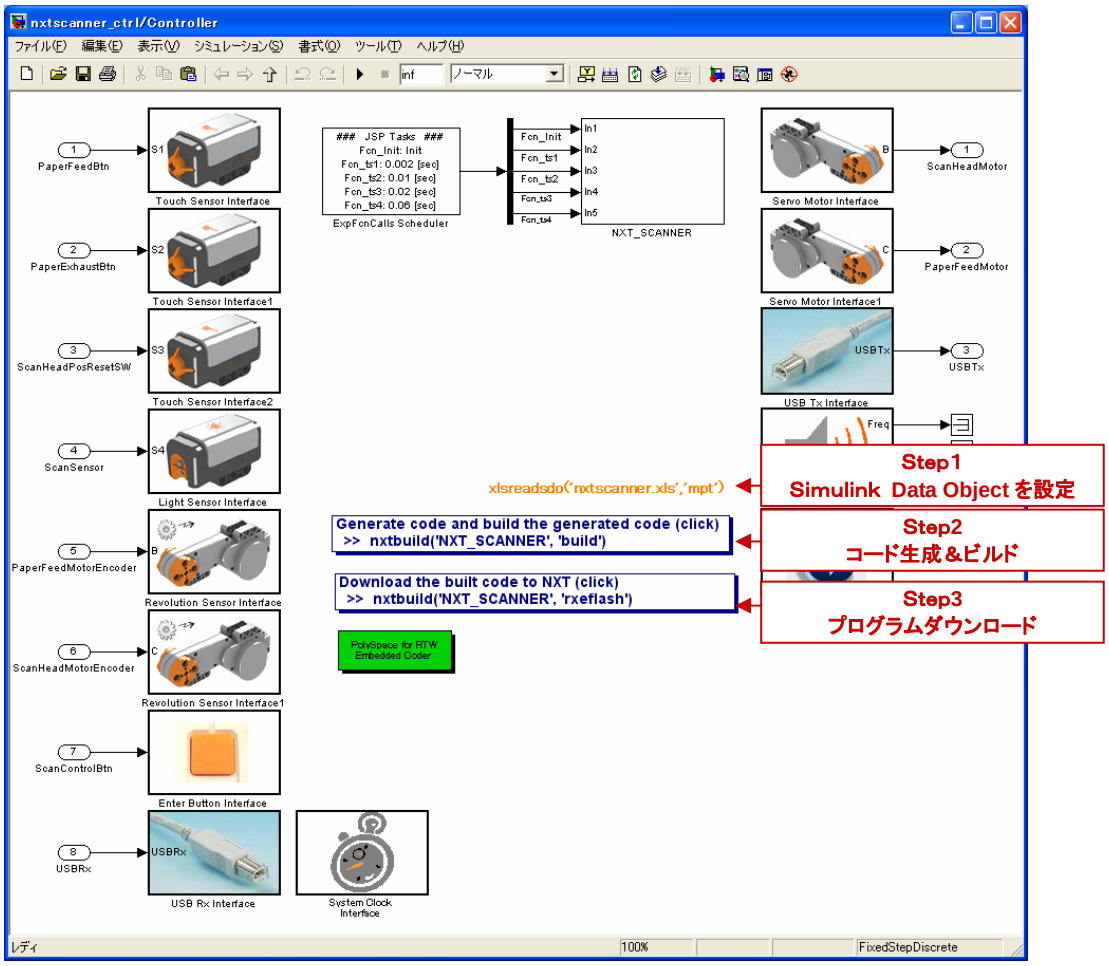


図 9-1　コード生成＆ビルド / プログラムダウンロード用アノテーション

コード生成結果については**付録**を参照してください。

10 生成コードの検証

PolySpace®を用いて、生成したコードの検証を行います。本資料では、PolySpace のインストールに関する説明は省略しており、NXT Scanner の検証に必要な設定に焦点を絞って説明しています。

10.1 PolySpace について

PolySpace は高い信頼性を要求されるソフトウェアの開発に向けた、実行時エラーの静的検証ツールです。PolySpace には二つの側面があり、実行時エラーを検出する事と、もう一歩進んで、ソースコードにエラーは存在しない、という確証を得る為の観点で使用することが出来ます。

PolySpace は演算実行の正しさ、または、正しくない場合はその状態を、4 色に分類して表示し、ユーザーにとって、コードにどの位の信頼性の度合いがあるかが、はっきりと分かるようになっています。

- green：正常
- red：ランタイムエラー
- gray：デッドコード
- orange：分類できない、要チェック

PolySpcae は、ハンドコーディングに対しての検証も可能ですが、オートコーディングに対して検証を行いモデルにフィードバックすることで、より高品質なモデリングが実現可能です。

10.2 PolySpace の設定

PolySpace の設定メニューは、図 10-1 に示すボタンで起動することが出来ます。

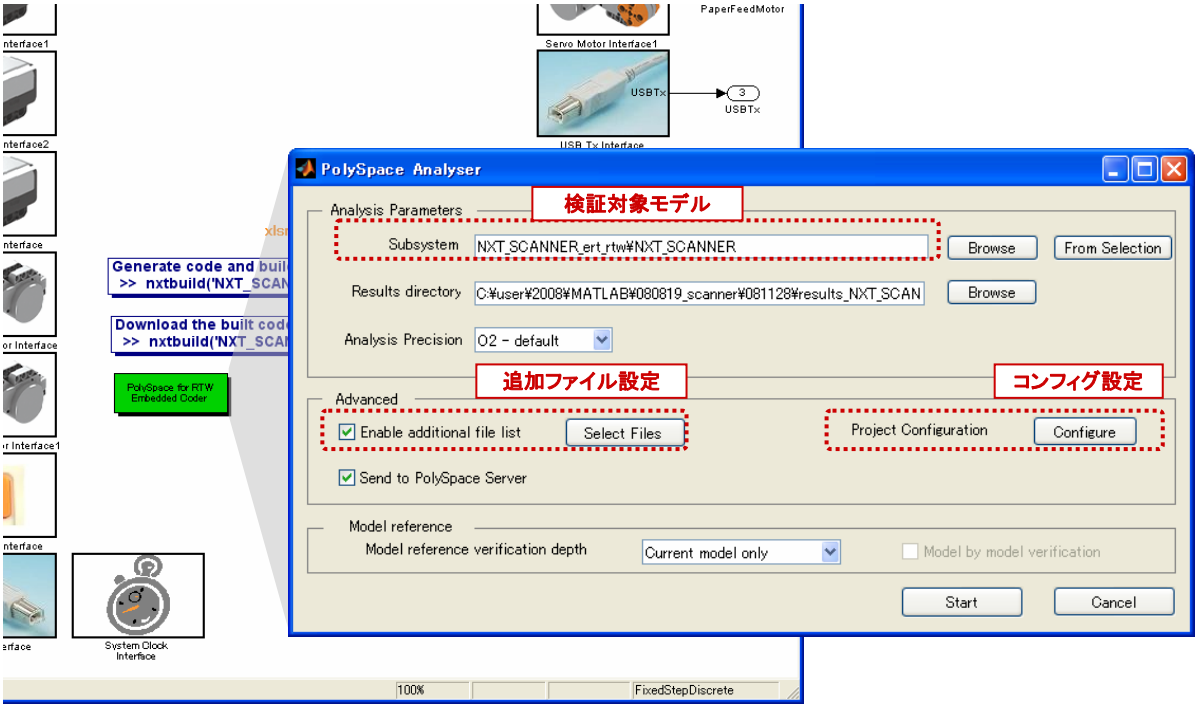


図 10-1　PolySpaceの起動

注意：

PolySpace 用の設定ファイルには、絶対パスを指定している箇所が複数存在しているので、ご使用の際はご注意ください。

検証対象モデル

PolySpace を起動したモデルが自動的に設定されます。NXT Scanner では、統合モデル nxt_scanner.mdl で検証を行い、NXT_SCANNER_ert_rtw フォルダに生成されるコードが検証対象となります。

コンフィグ設定

PolySpace のコンフィグ設定画面を起動します。画面イメージを、図 10-2 で示します。ここでは検証に必要な様々な設定を行います。NXT Scanner を検証するにあたりデフォルト設定から変更した箇所を赤枠で示しています。

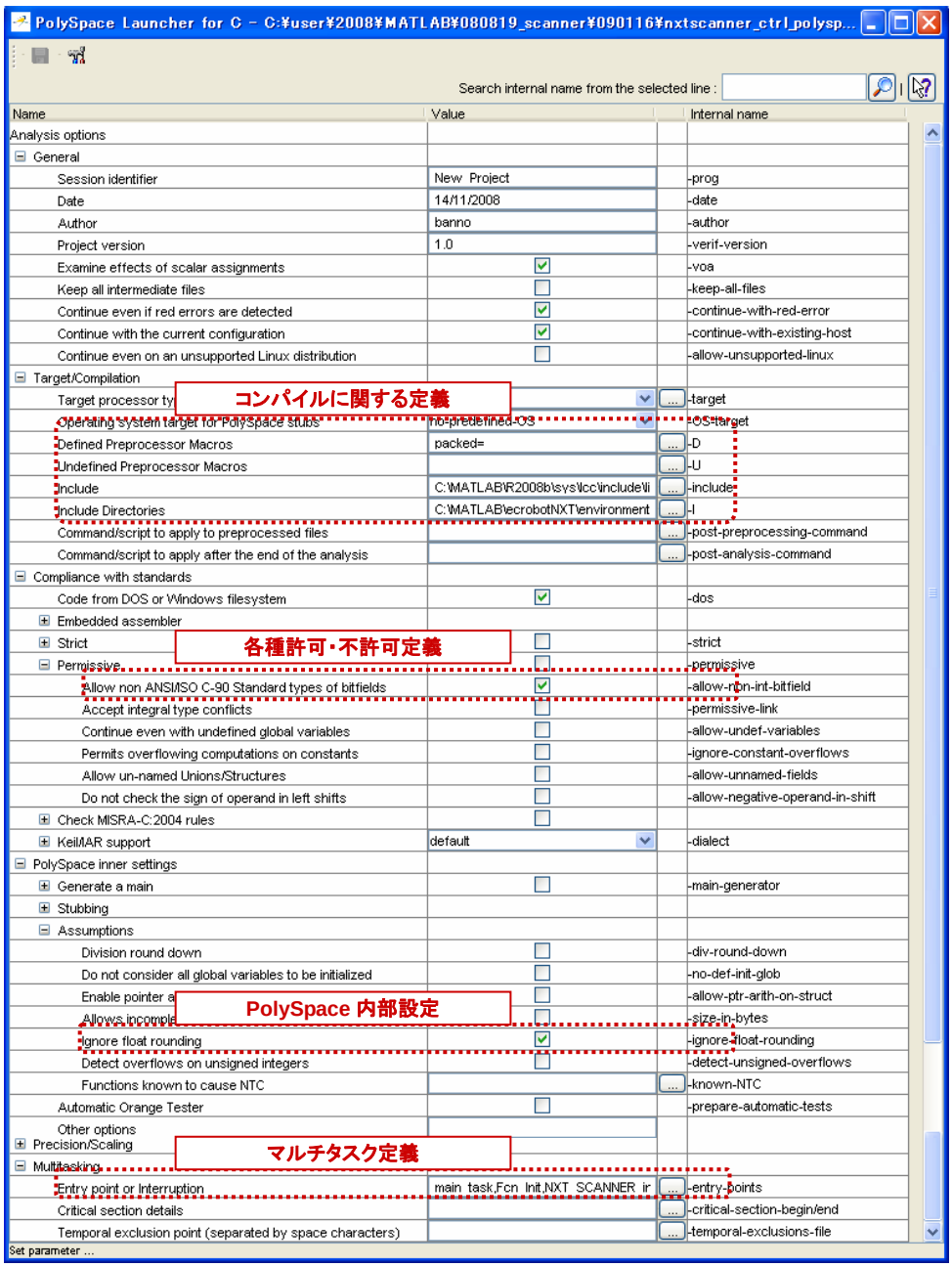


図 10-2　PolySpaceのConfigure設定画面

NXT_SCANNER_ert_rtw フォルダに生成されるコードだけでは、マルチタスクに関する情報がありません。従いまして、PolySpace での検証用に main 関数を作成したオリジナルファイル polyspace_main.c の Fcn_Init 関数、NXT_SCANNER_initialize 関数、main_task 関数をマルチタスクのエントリポイントとして指定します。

追加ファイル設定

コンフィグ設定のマルチタスク定義ファイル polyspace_main.c を登録しています。NXT Scanner は、スケジューラからのトリガで起動するタスクサブシステムを基本構造としています。polyspace_main.c の main_task 関数では、初期化タスク以外のタスク処理の関係を定義しています。PolySpace の検証ではタスク周期は任意の周期が考慮されます。

polyspace_main.c

```
#include "NXT_SCANNER.h"

extern int anyvalue(void);

int main(void){
    Fcn_Init();
    NXT_SCANNER_initialize();
    while(1){
        main_task();
    }
    return(0);
}
void main_task(void){
    while(anyvalue()){
        if(anyvalue()){
            Fcn_ts1();
        }
        if(anyvalue()){
            Fcn_ts2();
        }
        if(anyvalue()){
            Fcn_ts3();
        }
        if(anyvalue()){
            Fcn_ts4();
        }
    }
}
```

10.3 PolySpace のランタイムエラー検出

NXT Scanner では、PolySpace のランタイムエラー検出を確認するために、故意にモデル内にランタイムエラーが発生するモデリングを用意しています。スキャン制御モデルの Stateflow 内の”RUNTIME_ERROR”イベントで起動する State（RUNTIME_ERROR_STOP）内にそのコードを記載しています。PolySpace では、図 10-3 に示すように、ランタイムエラーを検出すると赤色で警告を行います。

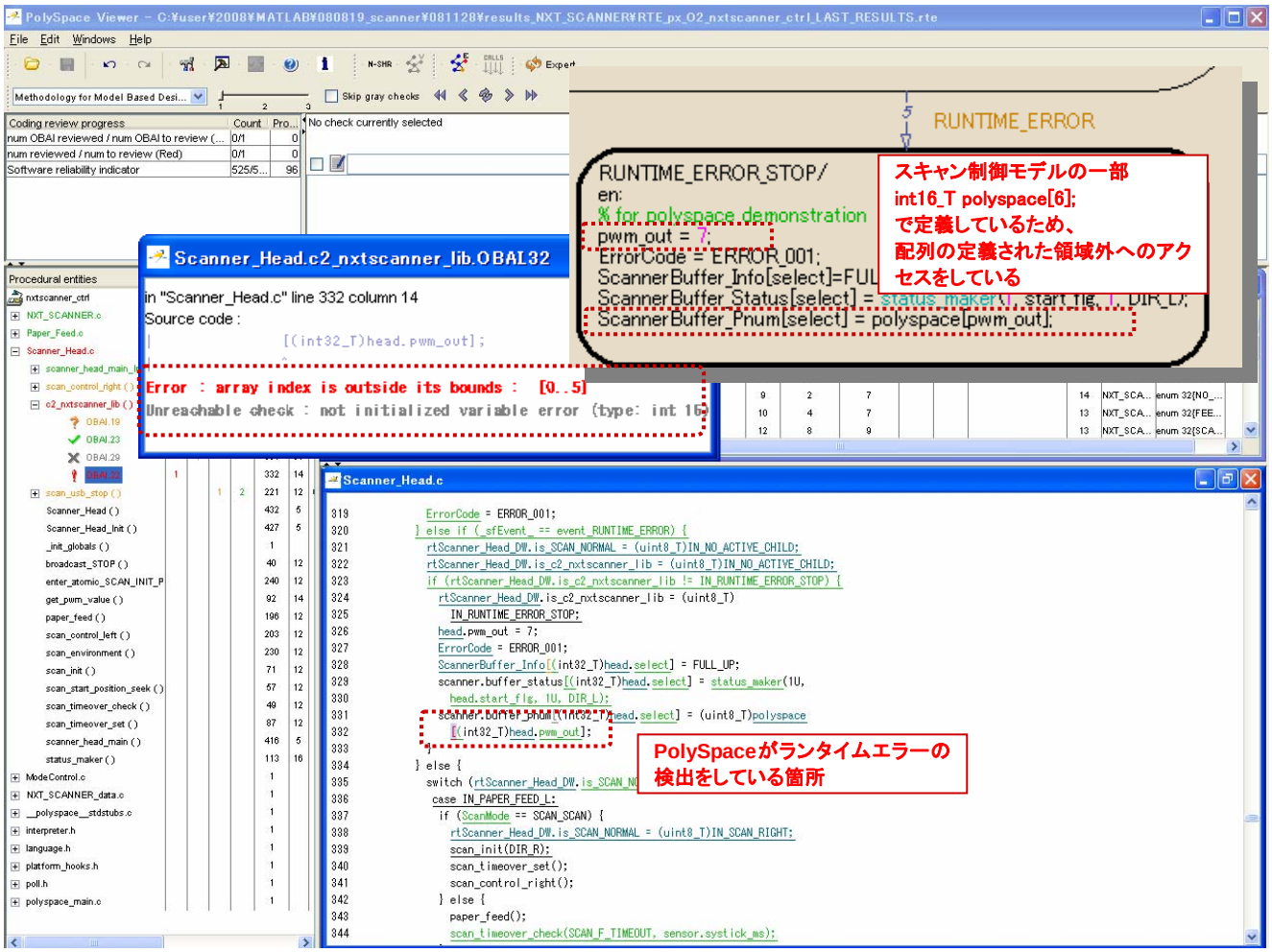


図 10-3　PolySpaceのランタイムエラー検出の例

参考

ランタイムエラーとは、コンパイル時には検出されない、プログラム実行中に発生するソフトウェアの致命的な不具合。具体的には PolySpace は以下のエラーを検出することが出来ます。

- 不正なポインタ参照（NULL、領域不足）
- 配列の定義された領域外へのアクセス
- 未初期化変数の参照
- 無効な数学演算（零除算、負数の平方根等）
- 整数・浮動小数点数のオーバーフロー、アンダーフロー
- 不正な型変換（小さな型へのキャスト）
- シフトサイズ違反
- 無限ループ（終了しないループ）
- コントロールを返さない関数コール
- デッドコード

10.4 PolySpace 検証結果

ランタイムエラーを発生させている箇所を削除した生成コードで PolySpace を実行します。図 10-4 は PolySpace の検証結果画面です。red と gray の指摘は存在していません。orange の指摘が 18 箇所あります。orange 指摘は、red、gray、green のどれにも分類が出来なかったという意味であり、個々をレビューにてチェックします。

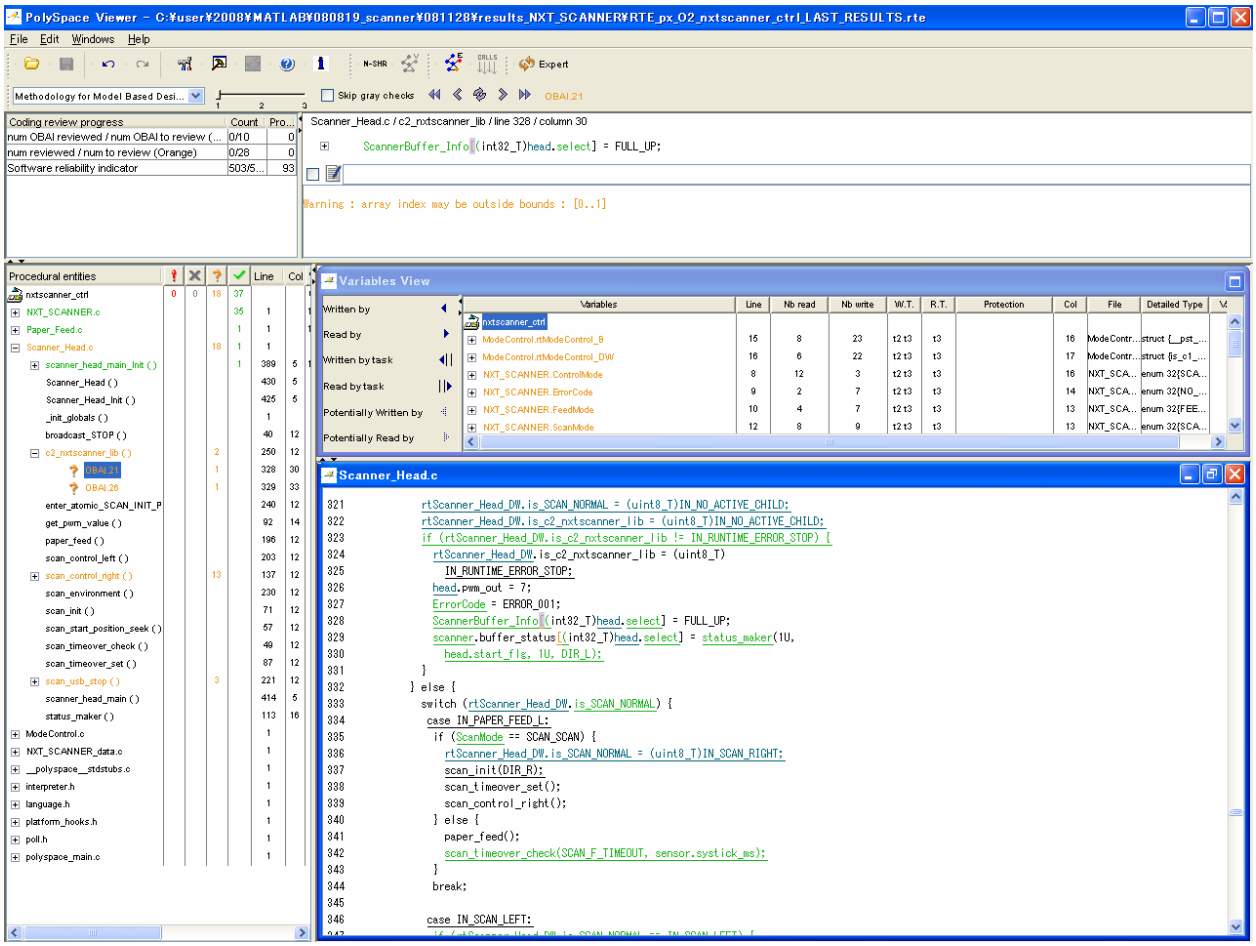


図 10-4　PolySpaceの検証結果

11 NXT Viewer について

NXT Viewer は、NXT Scanner 専用に作成された GUI ツールで、主に以下の２つの機能を持ちます。

- 画像表示：NXT Scanner のスキャンデータを USB 受信して、そのデータを１行毎にリアルタイムで表示、データを逐次結合し、加工して画像データを生成
- 画像補正：画像データを補正する

NXT Viewer は、GUIDE と呼ばれる MATLAB の GUI 作成機能に準じて作成されており、上記機能は M-言語により実現しています。特に、画像補正処理に関しては MATLAB のオプション製品である Image Processing Toolbox を利用しているのが特長です。

11.1 NXT Viewer の使い方

ここでは、NXT Viewer の使い方について説明します。

図 11-1　NXT Viewer画面イメージ

表 11-1 は、図 11-1 で示す NXT Viewer 画面上の各部機能をまとめた表です。

表 11-1　NXT Vewer各部の説明

番号	コンポーネント名	コンポーネントタイプ	機能
①	–	座標軸 （描画エリア）	スキャン実行中：スキャン中のイメージがリアルタイムで描画されます。 スキャン実行中以外：MAT ファイルからロードしたイメージが描画されます。 （詳細は11.3章を参照） 扱うイメージのサイズは M 行 255 列（M =< 255）
②	SCAN	プッシュボタン	確認ダイアログを表示します。OK が選択された場合にデータ待ちとなり、スキャンを開始します。 スキャンしたイメージが 255 行に達するか、スキャン停止ボタンが押された場合、スキャンを終了します。
③	LOAD	プッシュボタン	MAT ファイルの選択ダイアログを表示します。 MAT ファイルが選択された後、スキャンデータを読み込み描画エリアに描画します。
④	SAVE	プッシュボタン	MAT ファイルの保存ダイアログを表示します。 MAT ファイルが選択された後、スキャンデータを MAT ファイルに保存します。
⑤	IP	プッシュボタン	描画エリア（スキャンされた、または MAT ファイルからロードされたイメージ）に対し、画像補正を実行します。補正されたイメージは、描画エリアとは異なる画面に表示されます。 （詳細は11.4章を参照）
⑥	EXIT	プッシュボタン	確認ダイアログを表示し、OK が選択された場合に NXT Viewer を終了し画面を閉じます。
⑦	–	スタティックテキスト	現在のステータス等、各種情報を表示します。
⑧	–	エディットテキスト	明るさのレベルを表示します。
⑨	–	スライダー	明るさのレベルを指定します。（0～100）
⑩	–	エディットテキスト	コントラストのレベルを表示します。
⑪	–	スライダー	コントラストのレベルを指定します。（0～100）

11.2 USB 通信用コマンド（nxtusb）

NXT Viewer では、NXT Scanner からスキャンしたデータを受信するために、USB 通信用のコマンド（nxtusb）を使用しています。nxtusb は、自身の持つプライベート関数を通じて、Mindstorms NXT 提供の USB ドライバ（fantom）とインタフェースをとり、MATLAB 上での Mindstorms NXT との USB 通信を実現しています。上記プライベート関数の作成には、MATLAB の MEX 機能（MATLAB から実行可能な関数をプログラミング言語で表現する機能）が用いられています。

NXTScannerViewer.m

```
 （省略）
% nxtusbオブジェクトの作成
nuObj = nxtusb;
   :
% nxtusbによるUSBデータ受信
[len, buf] = read(nuObj, 'uint16', 32);
   :
% nxtusbによるUSB通信の終了処理
delete(nuObj);
clear nuObj
 （省略）
```

11.3 画像表示の概要

NXT Viewer では、NXT Scanner から送られてきた 10bit の生スキャンデータをグレースケールで表示します。そのためデータの劣化がなるべく発生しないように uint8（0-255）のデータに変換しています。変換式は実験により算出しています。

NXTScannerViewer.m

```
（省略）
% 簡易輝度補正
data1 = uint8(747 - data1);
scanData(row, 1:length(data1)) = data1;
imshow(scanData), drawnow
（省略）
```

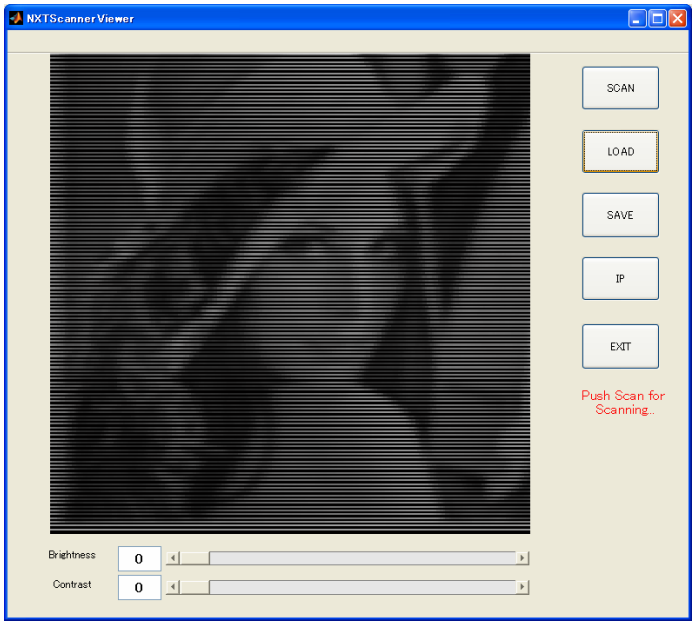


図 11-2　NXT Viewer画像表示

11.4 画像補正の概要

NXT Viewer は画像補正を以下の 3 段階のプロセスを経て行います。

1. カラーコントラスト補正 （Figure1）
2. 行間補間 （Figure2）
3. 画像先鋭化 （Figure3）

カラーコントラスト補正

画像表示時に輝度調整を行ったデータに対して、グレースケールの最適化を行います。M ファイル内の赤太字は、Image Processing Toolbox のコマンドです。

- Imadjust：イメージの強度値を調整
- Imshow：Handle Graphics figure のイメージを表示

NXTScannerViewer.m

```
（省略）
% カラーコントラスト補正
brLvl = get(handles.slider1,'Value');
cnLvl = get(handles.slider2, 'Value');
scanData = scanData + brLvl;
scanData = scanData * ((100 + cnLvl) / 100);
[xs,ys] = size(scanData);
mx = max(scanData(1:2:end));
mn = min(scanData(1:2:end));
scanData = imadjust(scanData,[double(mn)/255 double(mx)/255],[0 1],1);
（省略）
figure, imshow(scanData)
```

図 11-3　NXT Viewerカラーコントラスト補正画像

行間補間

NXT Scanner では、1 行飛ばしでスキャンしているので、ここで行間を補う補正をします。

- imresize：Lanczos3 補間方法によるイメージのリサイズ
- edgetaper：イメージのエッジ領域にテーパリング処理を適用
- Imshow：Handle Graphics figure のイメージを表示

NXTScannerViewer.m

```
（省略）
% 行間補間
I1 = scanData(1:2:end,:);
I1 = imresize(I1,[xs,ys],'lanczos3');
scanData2 = I1;
k = ones(3,6);
k = k/(size(k,1)*size(k,2));
scanData2 = edgetaper(scanData2,k);
（省略）
figure, imshow(scanData2)
```

図 11-4　NXT Viewer行間補間画像

画像先鋭化

ぼやけている画像を先鋭化する補正を行います。

- edge：イメージ内のエッジの検出
- strel：形態学的処理を行うための構造化要素を作成
- imdilate：イメージの形態学的膨張処理
- deconvblind：ブラインドデコンボリューションを使ってイメージのブレを除去

NXTScannerViewer.m

```
（省略）
% 画像先鋭化
a = edge(scanData2,'sobel',0.10);
se = strel('rect',[2 5]);
b = imdilate(a,se);
b = ~b;
b([1:3 end-[0:2]],:) = 0;
b(:,[1:3 end-[0:2]]) = 0;
[J,PSF] = deconvblind(scanData2,k,22,uint8(0),double(b),uint8(255));
（省略）
figure, imshow(J)
```

図 11-5　NXT Viewer先鋭化画像

12 実験結果

下記 URL に NXT Scanner の実験動画が公開されています。NXT Scanner のスキャンの様子と NXT Viewer の画像補正の様子をご覧になれます。

http://jp.youtube.com/watch?v=hadVwPJw3_0

図 12-1　NXT Scanner実験動画

13 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。

- NXT Scanner スキャン時間の短縮
- NXT Scanner 機構本体の改良（スキャナヘッド部をタイヤで移動させるなど）
- NXT Viewer 画像補正アルゴリズムの改良

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。

NXT_SCANNER.c

```
#include "NXT_SCANNER.h"
#include "NXT_SCANNER_private.h"

#define IN_CLOSE                       (1)
#define IN_OPEN                        (2)

uint8_T _sfEvent_;
ControlModeEnum ControlMode;
ErrorCodeEnum ErrorCode;
FeedModeEnum FeedMode;
ScannerBufferInfoEnum ScannerBuffer_Info[2];
ScanModeEnum ScanMode;
head_type head;
logger_type logger;
scanner_type scanner;
sensor_type sensor;
usb_type usb;
BlockIO rtB;
D_Work rtDWork;
PrevZCSigStates rtPrevZCSigState;
void Fcn_Init(void)
{
  ecrobot_set_motor_mode_speed(NXT_PORT_B, 1, 0);
  ecrobot_set_motor_mode_speed(NXT_PORT_C, 1, 0);
}

void Fcn_ts1_Init(void)
{
  Scanner_Head_Init();
}

void Fcn_ts1(void)
{
  sensor.scanner_head_reset_sw = ecrobot_get_touch_sensor(NXT_PORT_S3);
  sensor.scanner_head_rev = ecrobot_get_motor_rev(NXT_PORT_C);
  sensor.light_data = ecrobot_get_light_sensor(NXT_PORT_S4);
  sensor.systick_ms = ecrobot_get_systick_ms();
  Scanner_Head();
  ecrobot_set_motor_mode_speed(NXT_PORT_C, 1, head.pwm_out);
  logger.LOG_data1 = head.pwm_out;
  logger.LOG_data2 = rtDWork.paper_feed_pwm;
  logger.LOG_adc1 = (int16_T)ControlMode;
  logger.LOG_adc2 = (int16_T)ScanMode;
  logger.LOG_adc3 = (int16_T)sensor.scanner_head_rev;
  logger.LOG_adc4 = (int16_T)head.now_position;
  ecrobot_bt_adc_data_logger(logger.LOG_data1, logger.LOG_data2, logger.LOG_adc1,
    logger.LOG_adc2, logger.LOG_adc3, logger.LOG_adc4);
}

void Fcn_ts2_Init(void)
{
  Paper_Feed_Init();
}

void Fcn_ts2_Disable(void)
{
  Paper_Feed_Disable();
}
```

```
void Fcn_ts2_Start(void)
{
  Paper_Feed_Start();
}

void Fcn_ts2(void)
{
  rtB.RevolutionSensor_B = ecrobot_get_motor_rev(NXT_PORT_B);
  Paper_Feed();
  rtDWork.paper_feed_pwm = rtB.PWM;
  ecrobot_set_motor_mode_speed(NXT_PORT_B, 1, rtB.PWM);
}

void Fcn_ts3_Init(void)
{
  ModeControl_Init();
}

void Fcn_ts3(void)
{
  {
    boolean_T rtb_RelationalOperator_k;
    sensor.paper_exhaust_sw = ecrobot_get_touch_sensor(NXT_PORT_S1);
    sensor.paper_feed_sw = ecrobot_get_touch_sensor(NXT_PORT_S2);
    sensor.scan_control_sw = ecrobot_is_ENTER_button_pressed();
    ModeControl();
    rtb_RelationalOperator_k = (rtModeControl_B.SoundVol > 0U);
    if (rtb_RelationalOperator_k && (rtPrevZCSigState.EnabledSubsystem_Trig_ZCE
        != POS_ZCSIG)) {
      ecrobot_sound_tone(rtModeControl_B.SoundFreq, rtModeControl_B.SoundDur,
                         rtModeControl_B.SoundVol);
    }

    rtPrevZCSigState.EnabledSubsystem_Trig_ZCE = rtb_RelationalOperator_k ?
      POS_ZCSIG : ZERO_ZCSIG;
  }
}

void usb_send(void)
{
  uint8_T rtb_TmpHiddenBufferAtUSBTxWrite[64];

  {
    int32_T i;
    for (i = 0; i < 2; i++) {
      rtb_TmpHiddenBufferAtUSBTxWrite[i] = usb.send_status[i];
    }

    for (i = 0; i < 2; i++) {
      rtb_TmpHiddenBufferAtUSBTxWrite[i + 2] = usb.send_number[i];
    }

    for (i = 0; i < 60; i++) {
      rtb_TmpHiddenBufferAtUSBTxWrite[i + 4] = usb.send_data[i];
    }

    ecrobot_send_usb(rtb_TmpHiddenBufferAtUSBTxWrite, 0, MAX_USB_DATA_LEN);
  }
}
```

```
static void usb_communication(uint8_T sf_arg_select_buffer);
static void init_buffer(uint8_T sf_arg_select_buf);
static void usb_function(void);
static uint8_T usb_final_check(void);
static void usb_communication(uint8_T sf_arg_select_buffer)
{
  uint8_T sf_index;
  usb.send_status[1] = 0U;
  usb.send_status[0] = (uint8_T)scanner.buffer_status[sf_arg_select_buffer];
  usb.send_number[1] = 0U;
  usb.send_number[0] = scanner.buffer_number[sf_arg_select_buffer];
  for (sf_index = 0U; sf_index < 30; sf_index++) {
    usb.send_data[(sf_index << 1) + 1] = (uint8_T)((scanner.buffer_data
      [(sf_index << 1) + sf_arg_select_buffer] & 0xFF00) >> 8);
    usb.send_data[sf_index << 1] = (uint8_T)(scanner.buffer_data[(sf_index << 1)
      + sf_arg_select_buffer] & 0x00FF);
  }
}

static void init_buffer(uint8_T sf_arg_select_buf)
{
  uint8_T sf_index;
  ScannerBuffer_Info[sf_arg_select_buf] = TRANS_FINISH;
  scanner.buffer_status[sf_arg_select_buf] = 0U;
  scanner.buffer_number[sf_arg_select_buf] = 0U;
  scanner.buffer_pnum[sf_arg_select_buf] = 0U;
  for (sf_index = 0U; sf_index < 30; sf_index++) {
    scanner.buffer_data[sf_arg_select_buf + (sf_index << 1)] = 0U;
  }
}

static void usb_function(void)
{
  int32_T i;
  for (i = 0; i < 2; i++) {
    usb.send_status[i] = 0U;
    usb.send_number[i] = 0U;
  }

  for (i = 0; i < 60; i++) {
    usb.send_data[i] = 0U;
  }

  rtDWork.disconnect = 0U;
  if ((ScannerBuffer_Info[0] == FULL_UP) && (ScannerBuffer_Info[1] == FULL_UP))
  {
    if (scanner.buffer_pnum[0] <= scanner.buffer_pnum[1]) {
      usb_communication(0U);
      init_buffer(0U);
      usb_send();
    } else {
      usb_communication(1U);
      init_buffer(1U);
      usb_send();
    }
  } else if (ScannerBuffer_Info[0] == FULL_UP) {
    usb_communication(0U);
    init_buffer(0U);
    usb_send();
  } else {
    if (ScannerBuffer_Info[1] == FULL_UP) {
      usb_communication(1U);
      init_buffer(1U);
      usb_send();
    }
  }
}
```

```
void usb_communication_main_Init(void)
{
  {
    int32_T i;
    rtDWork.is_active_c4_nxtscanner_lib = 0U;
    rtDWork.is_c4_nxtscanner_lib = 0U;
    rtDWork.disconnect = 0U;
    for (i = 0; i < 2; i++) {
      usb.send_status[i] = 0U;
      usb.send_number[i] = 0U;
    }

    for (i = 0; i < 60; i++) {
      usb.send_data[i] = 0U;
    }
  }
}

void usb_communication_main(void)
{
  if (rtDWork.is_active_c4_nxtscanner_lib == 0) {
    rtDWork.is_active_c4_nxtscanner_lib = 1U;
    rtDWork.is_c4_nxtscanner_lib = (uint8_T)IN_OPEN;
    usb_function();
    rtDWork.disconnect = usb_final_check();
  } else {
    switch (rtDWork.is_c4_nxtscanner_lib) {
     case IN_CLOSE:
      if (rtDWork.disconnect == 0) {
        rtDWork.is_c4_nxtscanner_lib = (uint8_T)IN_OPEN;
        usb_function();
        rtDWork.disconnect = usb_final_check();
      }
      break;

     case IN_OPEN:
      if (rtDWork.disconnect == 1) {
        rtDWork.is_c4_nxtscanner_lib = (uint8_T)IN_CLOSE;
        if (((uint8_T)1U) == 1) {
          ecrobot_disconnect_usb();
        }

        rtDWork.disconnect = 0U;
      } else {
        usb_function();
        rtDWork.disconnect = usb_final_check();
      }
      break;

     default:
      rtDWork.is_c4_nxtscanner_lib = (uint8_T)IN_OPEN;
      usb_function();
      rtDWork.disconnect = usb_final_check();
      break;
    }
  }
}

void Fcn_ts4_Init(void)
{
  usb_communication_main_Init();
}

void Fcn_ts4(void)
{
  usb_communication_main();
}
```

```
void NXT_SCANNER_initialize(void)
{
  {
    rtB.ErrorCode_i = NO_ERROR;
  }

  ControlMode = ERROR;
  ErrorCode = NO_ERROR;
  FeedMode = FEED_INIT;

  {
    int_T i;
    for (i = 0; i < 2; i++) {
      ScannerBuffer_Info[i] = TRANS_FINISH;
    }
  }

  ScanMode = SCAN_INIT;
  ModeControl_initialize();
  Fcn_ts2_Start();
  ErrorCode = NO_ERROR;
  rtPrevZCSigState.EnabledSubsystem_Trig_ZCE = POS_ZCSIG;
  rtPrevZCSigState.Error_Detection_Trig_ZCE = POS_ZCSIG;
  _sfEvent_ = CALL_EVENT;
  Fcn_ts1_Init();
  Fcn_ts2_Init();
  Fcn_ts3_Init();
  Fcn_ts4_Init();
}
```

ModeControl.c

```
#include "ModeControl.h"

#include "NXT_SCANNER.h"
#include "NXT_SCANNER_private.h"

#define IN_ERROR                       (1)
#define IN_IDLE                        (1)
#define IN_INITIALIZE                   (2)
#define IN_NORMAL                      (2)
#define IN_NO_ACTIVE_CHILD_e0           (0)
#define IN_PAPER_EXHAUST                (3)
#define IN_PAPER_FEED                  (4)
#define IN_SCAN_CONTROL                 (5)

rtB_ModeControl rtModeControl_B;
rtDW_ModeControl rtModeControl_DW;
void ModeControl_Init(void)
{
  rtModeControl_DW.is_NORMAL = 0U;
  rtModeControl_DW.is_active_c1_nxtscanner_lib = 0U;
  rtModeControl_DW.is_c1_nxtscanner_lib = 0U;
  rtModeControl_B.ControlMode_d = ERROR;
  rtModeControl_B.SoundFreq = 0U;
  rtModeControl_B.SoundDur = 0U;
  rtModeControl_B.SoundVol = 0U;
}

void ModeControl(void)
{
  {
    ScanModeEnum rtb_DataStoreRead2_k;
    rtb_DataStoreRead2_k = ScanMode;
    if (((ScanMode == SCAN_INIT) && (SCAN_INIT !=
         rtModeControl_DW.UnitDelay1_DSTATE)) || ((sensor.scan_control_sw ==
         ((uint8_T)1U)) && (rtModeControl_DW.UnitDelay2_DSTATE == ((uint8_T)0U))))
    {
      if (rtModeControl_DW.UnitDelay2_DSTATE_m >= ((uint8_T)1U)) {
        rtModeControl_B.Switch = ((uint8_T)0U);
      } else {
        rtModeControl_B.Switch = ((uint8_T)1U);
      }

      rtModeControl_DW.UnitDelay2_DSTATE_m = rtModeControl_B.Switch;
    }

    rtModeControl_DW.UnitDelay1_DSTATE = rtb_DataStoreRead2_k;
    rtModeControl_DW.UnitDelay2_DSTATE = sensor.scan_control_sw;
    if (rtModeControl_DW.is_active_c1_nxtscanner_lib == 0) {
      rtModeControl_DW.is_active_c1_nxtscanner_lib = 1U;
      rtModeControl_DW.is_c1_nxtscanner_lib = (uint8_T)IN_NORMAL;
      rtModeControl_DW.is_NORMAL = (uint8_T)IN_INITIALIZE;
      rtModeControl_B.ControlMode_d = INITIALIZE;
    } else {
      switch (rtModeControl_DW.is_c1_nxtscanner_lib) {
       case IN_ERROR:
        break;

       case IN_NORMAL:
        if (ErrorCode != NO_ERROR) {
          if (ErrorCode == ERROR_001) {
            rtModeControl_B.SoundFreq = 880U;
          } else {
            rtModeControl_B.SoundFreq = 220U;
          }
          rtModeControl_DW.is_NORMAL = (uint8_T)IN_NO_ACTIVE_CHILD_e0;
          rtModeControl_DW.is_c1_nxtscanner_lib = (uint8_T)IN_ERROR;
          rtModeControl_B.ControlMode_d = ERROR;
          rtModeControl_B.SoundDur = 3000U;
          rtModeControl_B.SoundVol = 20U;
```

```
      } else {
        switch (rtModeControl_DW.is_NORMAL) {
         case IN_IDLE:
          if (rtModeControl_B.Switch == 1) {
            rtModeControl_DW.is_NORMAL = (uint8_T)IN_SCAN_CONTROL;
            rtModeControl_B.ControlMode_d = SCAN;
          } else if (sensor.paper_feed_sw == 1) {
            rtModeControl_DW.is_NORMAL = (uint8_T)IN_PAPER_FEED;
            rtModeControl_B.ControlMode_d = PAPERFEED;
          } else {
            if (sensor.paper_exhaust_sw == 1) {
              rtModeControl_DW.is_NORMAL = (uint8_T)IN_PAPER_EXHAUST;
              rtModeControl_B.ControlMode_d = PAPEREXHAUST;
            }
          }
          break;

         case IN_INITIALIZE:
          if ((ScanMode == SCAN_IDLE) && (FeedMode == FEED_IDLE)) {
            rtModeControl_DW.is_NORMAL = (uint8_T)IN_IDLE;
            rtModeControl_B.ControlMode_d = IDLE;
          }
          break;

         case IN_PAPER_EXHAUST:
          if (sensor.paper_exhaust_sw == 0) {
            rtModeControl_DW.is_NORMAL = (uint8_T)IN_IDLE;
            rtModeControl_B.ControlMode_d = IDLE;
          }
          break;

         case IN_PAPER_FEED:
          if (sensor.paper_feed_sw == 0) {
            rtModeControl_DW.is_NORMAL = (uint8_T)IN_IDLE;
            rtModeControl_B.ControlMode_d = IDLE;
          }
          break;

         case IN_SCAN_CONTROL:
          if (rtModeControl_B.Switch == 0) {
            rtModeControl_DW.is_NORMAL = (uint8_T)IN_INITIALIZE;
            rtModeControl_B.ControlMode_d = INITIALIZE;
          }
          break;

         default:
          rtModeControl_DW.is_NORMAL = (uint8_T)IN_INITIALIZE;
          rtModeControl_B.ControlMode_d = INITIALIZE;
          break;
        }
       }
       break;

      default:
       rtModeControl_DW.is_c1_nxtscanner_lib = (uint8_T)IN_NORMAL;
       rtModeControl_DW.is_NORMAL = (uint8_T)IN_INITIALIZE;
       rtModeControl_B.ControlMode_d = INITIALIZE;
       break;
     }
   }

   ControlMode = rtModeControl_B.ControlMode_d;
  }
}

void ModeControl_initialize(void)
{
  {
   rtModeControl_B.ControlMode_d = ERROR;
  }
}
```

Scanner_Head.c

```
#include "Scanner_Head.h"

#include "NXT_SCANNER.h"
#include "NXT_SCANNER_private.h"

#define DIR_L                        (1U)
#define DIR_R                        (0U)
#define IN_INIT_ERROR_STOP             (1)
#define IN_NO_ACTIVE_CHILD             (0)
#define IN_PAPER_FEED_L               (1)
#define IN_RUNTIME_ERROR_STOP          (2)
#define IN_SCAN_ERROR                 (3)
#define IN_SCAN_ERROR_STOP             (4)
#define IN_SCAN_INIT_POSITION          (5)
#define IN_SCAN_LEFT                  (2)
#define IN_SCAN_NORMAL                (6)
#define IN_SCAN_RIGHT                 (3)
#define event_FINISH                  (1U)
#define event_RUNTIME_ERROR            (2U)
#define event_STOP                   (0U)

rtDW_Scanner_Head rtScanner_Head_DW;
static void broadcast_STOP(void);
static void scan_timeover_check(uint32_T sf_arg_target_ms, uint32_T
  sf_arg_now_clock);
static void scan_start_position_seek(void);
static void scan_init(uint8_T sf_arg_target_dir);
static void scan_timeover_set(void);
static int8_T get_pwm_value(uint8_T sf_arg_dir, int32_T sf_arg_position, uint8_T
  sf_arg_max, uint8_T sf_arg_min);
static uint16_T status_maker(uint8_T sf_arg_elp, uint8_T sf_arg_start, uint8_T
  sf_arg_stop, uint8_T sf_arg_dir);
static void scan_control_right(void);
static void paper_feed(void);
static void scan_control_left(void);
static void scan_usb_stop(void);
static void scan_environment(void);
static void enter_atomic_SCAN_INIT_POSITION(void);
static void c2_nxtscanner_lib(void);
static void broadcast_STOP(void)
{
  uint8_T sf_previousEvent;
  sf_previousEvent = _sfEvent_;
  _sfEvent_ = event_STOP;
  c2_nxtscanner_lib();
  _sfEvent_ = sf_previousEvent;
}

static void scan_timeover_check(uint32_T sf_arg_target_ms, uint32_T
  sf_arg_now_clock)
{
  if (sf_arg_target_ms < sf_arg_now_clock - head.start_time) {
    broadcast_STOP();
  }
}

static void scan_start_position_seek(void)
{
  if (sensor.scanner_head_reset_sw == 1) {
    head.pwm_out = 0;
    ScanMode = SCAN_IDLE;
    head.scan_init_end_flg = 1U;
  } else if (head.scan_init_end_flg == 1) {
    broadcast_STOP();
  } else {
    head.pwm_out = SCAN_INITIALIZE_PWM;
    scan_timeover_check(SCAN_INITIALIZE_TIMEOUT, sensor.systick_ms);
  }
}
```

```
static void scan_init(uint8_T sf_arg_target_dir)
{
  head.line_end_flg = 0U;
  head.packet_cnt = 0U;
  ScanMode = SCAN_SCAN;
  head.encoder_base = (int32_T)SCAN_START;
  if (sf_arg_target_dir == DIR_L) {
    head.start_position = sensor.scanner_head_rev + BACKLASH_ADJUST;
  } else if (head.start_flg == 1) {
    head.start_position = (sensor.scanner_head_rev - RELEASE_TOUCH_SENSOR) -
      BACKLASH_ADJUST;
  } else {
    head.start_position = sensor.scanner_head_rev - BACKLASH_ADJUST;
  }
}

static void scan_timeover_set(void)
{
  head.start_time = sensor.systick_ms;
}

static int8_T get_pwm_value(uint8_T sf_arg_dir, int32_T sf_arg_position, uint8_T
  sf_arg_max, uint8_T sf_arg_min)
{
  int8_T sf_pwm;
  if (sf_arg_position < 0) {
    sf_pwm = (int8_T)sf_arg_max;
  } else if (sf_arg_position > SCAN_EDGE) {
    sf_pwm = 0;
  } else if (sf_arg_position > SCAN_EDGE - 60) {
    sf_pwm = (int8_T)sf_arg_min;
  } else {
    sf_pwm = (int8_T)sf_arg_max;
  }

  if (sf_arg_dir == 0) {
    return (int8_T)(-sf_pwm);
  }

  return sf_pwm;
}

static uint16_T status_maker(uint8_T sf_arg_elp, uint8_T sf_arg_start, uint8_T
  sf_arg_stop, uint8_T sf_arg_dir)
{
  uint16_T sf_out;
  sf_out = 1U;
  if (sf_arg_elp == 1) {
    sf_out = 17U;
  }

  if (sf_arg_start == 1) {
    sf_out += 8;
  }

  if (sf_arg_stop == 1) {
    sf_out += 4;
  }

  if (sf_arg_dir == 1) {
    sf_out += 2;
  }

  return sf_out;
}
```

```
static void scan_control_right(void)
{
  uint8_T sf_previousEvent;
  head.now_position = head.start_position - sensor.scanner_head_rev;
  head.pwm_out = get_pwm_value(DIR_R, head.now_position, SCAN_R_MAX_PWM,
    SCAN_R_MIN_PWM);
  if ((sensor.scanner_head_reset_sw == 1) && (head.now_position > 0)) {
    broadcast_STOP();
  } else if (head.pwm_out == 0) {
    FeedMode = FEED_PAPERFEED;
    ScanMode = SCAN_FEED;
  } else {
    if ((head.now_position > SCAN_START) && (head.now_position >
         head.encoder_base)) {
      if (head.now_position <= SCAN_EDGE - SCAN_START) {
        head.encoder_base = head.encoder_base + 2;
        scanner.buffer_data[head.select + (scanner.buffer_number[(int32_T)
          head.select] << 1)] = sensor.light_data;
        scanner.buffer_number[(int32_T)head.select] = (uint8_T)
          (scanner.buffer_number[(int32_T)head.select] + 1);
        ScannerBuffer_Info[(int32_T)head.select] = STORING;
        if (scanner.buffer_number[(int32_T)head.select] >= 30) {
          head.packet_cnt = (uint8_T)(head.packet_cnt + 1);
          ScannerBuffer_Info[(int32_T)head.select] = FULL_UP;
          scanner.buffer_status[(int32_T)head.select] = status_maker(0U,
            head.start_flg, 0U, DIR_R);
          scanner.buffer_pnum[(int32_T)head.select] = head.packet_cnt;
          head.start_flg = 0U;
          if (head.select == 1) {
            head.select = 0U;
          } else {
            head.select = 1U;
          }
        }
      } else {
        if (head.line_end_flg == 0) {
          head.line_end_flg = 1U;
          head.line_cnt = head.line_cnt + 1U;
          ScannerBuffer_Info[(int32_T)head.select] = FULL_UP;
          head.packet_cnt = (uint8_T)(head.packet_cnt + 1);
          scanner.buffer_pnum[(int32_T)head.select] = head.packet_cnt;
          if (head.line_cnt >= 127U) {
            scanner.buffer_status[(int32_T)head.select] = status_maker(1U,
              head.start_flg, 1U, DIR_R);
            ScanMode = SCAN_INIT;
            sf_previousEvent = _sfEvent_;
            _sfEvent_ = event_FINISH;
            c2_nxtscanner_lib();
            _sfEvent_ = sf_previousEvent;
          } else {
            scanner.buffer_status[(int32_T)head.select] = status_maker(1U,
              head.start_flg, 0U, DIR_R);
          }
        }
      }
    }
  }
}

static void paper_feed(void)
{
  if (FeedMode == FEED_IDLE) {
    ScanMode = SCAN_SCAN;
  }
}
```

```
static void scan_control_left(void)
{
  uint8_T sf_previousEvent;
  head.now_position = sensor.scanner_head_rev - head.start_position;
  head.pwm_out = get_pwm_value(DIR_L, head.now_position, SCAN_L_MAX_PWM,
    SCAN_L_MIN_PWM);
  if ((sensor.scanner_head_reset_sw == 1) && (head.now_position > 0)) {
    sf_previousEvent = _sfEvent_;
    _sfEvent_ = event_RUNTIME_ERROR;
    c2_nxtscanner_lib();
    _sfEvent_ = sf_previousEvent;
  } else {
    if (head.pwm_out == 0) {
      ScanMode = SCAN_FEED;
    }
  }
}

static void scan_usb_stop(void)
{
  head.pwm_out = 0;
  ScannerBuffer_Info[(int32_T)head.select] = FULL_UP;
  scanner.buffer_status[(int32_T)head.select] = status_maker(1U, head.start_flg,
    1U, DIR_R);
  scanner.buffer_pnum[(int32_T)head.select] = (uint8_T)(head.packet_cnt + 1);
}

static void scan_environment(void)
{
  head.start_flg = 1U;
  head.select = 0U;
  head.line_cnt = 0U;
  head.line_end_flg = 0U;
  head.packet_cnt = 0U;
  head.scan_init_end_flg = 0U;
}

static void enter_atomic_SCAN_INIT_POSITION(void)
{
  if (rtScanner_Head_DW.is_c2_nxtscanner_lib != IN_SCAN_INIT_POSITION) {
    rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_SCAN_INIT_POSITION;
    scan_environment();
    scan_timeover_set();
    scan_start_position_seek();
  }
}

static void c2_nxtscanner_lib(void)
{
  if (rtScanner_Head_DW.is_active_c2_nxtscanner_lib == 0) {
    rtScanner_Head_DW.is_active_c2_nxtscanner_lib = 1U;
    if (rtScanner_Head_DW.is_c2_nxtscanner_lib != IN_SCAN_INIT_POSITION) {
      rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_SCAN_INIT_POSITION;
      scan_environment();
      scan_timeover_set();
      scan_start_position_seek();
    }
  } else {
    switch (rtScanner_Head_DW.is_c2_nxtscanner_lib) {
     case IN_INIT_ERROR_STOP:
      if (ControlMode == ERROR) {
        rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_SCAN_ERROR;
        head.pwm_out = 0;
      }
      break;

     case IN_RUNTIME_ERROR_STOP:
      break;

     case IN_SCAN_ERROR:
      break;
```

```
    case IN_SCAN_ERROR_STOP:
     break;

    case IN_SCAN_INIT_POSITION:
     if (_sfEvent_ == event_STOP) {
       rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_INIT_ERROR_STOP;
       head.pwm_out = 0;
       ErrorCode = ERROR_001;
     } else if (ControlMode == ERROR) {
       rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_SCAN_ERROR;
       head.pwm_out = 0;
     } else if ((ControlMode == SCAN) && (ScanMode == SCAN_IDLE)) {
       rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_SCAN_NORMAL;
       if (rtScanner_Head_DW.is_SCAN_NORMAL != IN_SCAN_RIGHT) {
         rtScanner_Head_DW.is_SCAN_NORMAL = (uint8_T)IN_SCAN_RIGHT;
         scan_init(DIR_R);
         scan_timeover_set();
         scan_control_right();
       }
     } else {
       scan_start_position_seek();
     }
     break;

    case IN_SCAN_NORMAL:
     if (rtScanner_Head_DW.is_c2_nxtscanner_lib == IN_SCAN_NORMAL) {
       if (ControlMode == ERROR) {
         scan_usb_stop();
         rtScanner_Head_DW.is_SCAN_NORMAL = (uint8_T)IN_NO_ACTIVE_CHILD;
         rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_SCAN_ERROR;
         head.pwm_out = 0;
       } else if ((ControlMode != ERROR) && (ControlMode != SCAN)) {
         scan_usb_stop();
         rtScanner_Head_DW.is_SCAN_NORMAL = (uint8_T)IN_NO_ACTIVE_CHILD;
         rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_NO_ACTIVE_CHILD;
         enter_atomic_SCAN_INIT_POSITION();
       } else if (_sfEvent_ == event_FINISH) {
         rtScanner_Head_DW.is_SCAN_NORMAL = (uint8_T)IN_NO_ACTIVE_CHILD;
         rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_NO_ACTIVE_CHILD;
         enter_atomic_SCAN_INIT_POSITION();
       } else if (_sfEvent_ == event_STOP) {
         rtScanner_Head_DW.is_SCAN_NORMAL = (uint8_T)IN_NO_ACTIVE_CHILD;
         rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_SCAN_ERROR_STOP;
         scan_usb_stop();
         ErrorCode = ERROR_001;
       } else if (_sfEvent_ == event_RUNTIME_ERROR) {
         rtScanner_Head_DW.is_SCAN_NORMAL = (uint8_T)IN_NO_ACTIVE_CHILD;
         rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_NO_ACTIVE_CHILD;
         if (rtScanner_Head_DW.is_c2_nxtscanner_lib != IN_RUNTIME_ERROR_STOP) {
           rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)
             IN_RUNTIME_ERROR_STOP;
           head.pwm_out = 7;
           ErrorCode = ERROR_001;
           ScannerBuffer_Info[(int32_T)head.select] = FULL_UP;
           scanner.buffer_status[(int32_T)head.select] = status_maker(1U,
             head.start_flg, 1U, DIR_L);
         }
       } else {
         switch (rtScanner_Head_DW.is_SCAN_NORMAL) {
          case IN_PAPER_FEED_L:
           if (ScanMode == SCAN_SCAN) {
             rtScanner_Head_DW.is_SCAN_NORMAL = (uint8_T)IN_SCAN_RIGHT;
             scan_init(DIR_R);
             scan_timeover_set();
             scan_control_right();
           } else {
             paper_feed();
             scan_timeover_check(SCAN_F_TIMEOUT, sensor.systick_ms);
           }
           break;
```

```
          case IN_SCAN_LEFT:
           if (rtScanner_Head_DW.is_SCAN_NORMAL == IN_SCAN_LEFT) {
             if (ScanMode == SCAN_FEED) {
               rtScanner_Head_DW.is_SCAN_NORMAL = (uint8_T)IN_PAPER_FEED_L;
               scan_timeover_set();
               paper_feed();
             } else {
               scan_control_left();
               scan_timeover_check(SCAN_L_TIMEOUT, sensor.systick_ms);
             }
           }
           break;

          case IN_SCAN_RIGHT:
           if (rtScanner_Head_DW.is_SCAN_NORMAL == IN_SCAN_RIGHT) {
             if (ScanMode == SCAN_FEED) {
               ScanMode = SCAN_SCAN;
               rtScanner_Head_DW.is_SCAN_NORMAL = (uint8_T)IN_SCAN_LEFT;
               scan_init(DIR_L);
               scan_timeover_set();
               scan_control_left();
             } else {
               scan_control_right();
               scan_timeover_check(SCAN_R_TIMEOUT, sensor.systick_ms);
             }
           }
           break;

          default:
           rtScanner_Head_DW.is_SCAN_NORMAL = (uint8_T)IN_NO_ACTIVE_CHILD;
           break;
         }
       }
     }
     break;

    default:
     rtScanner_Head_DW.is_c2_nxtscanner_lib = (uint8_T)IN_NO_ACTIVE_CHILD;
     break;
   }
  }
}

void scanner_head_main_Init(void)
{
  {
    int32_T i;
    rtScanner_Head_DW.is_SCAN_NORMAL = 0U;
    rtScanner_Head_DW.is_active_c2_nxtscanner_lib = 0U;
    rtScanner_Head_DW.is_c2_nxtscanner_lib = 0U;
    head.start_position = 0;
    head.select = 0U;
    head.start_flg = 0U;
    head.line_cnt = 0U;
    head.line_end_flg = 0U;
    head.packet_cnt = 0U;
    head.scan_init_end_flg = 0U;
    head.start_time = 0U;
    for (i = 0; i < 6; i++) {
      polyspace[i] = 0;
    }

    head.pwm_out = 0;
    head.now_position = 0;
    head.encoder_base = 0;
  }
}
```

```
void scanner_head_main(void)
{
  {
    uint8_T sf_previousEvent;
    sf_previousEvent = _sfEvent_;
    _sfEvent_ = CALL_EVENT;
    c2_nxtscanner_lib();
    _sfEvent_ = sf_previousEvent;
  }
}

void Scanner_Head_Init(void)
{
  scanner_head_main_Init();
}

void Scanner_Head(void)
{
  scanner_head_main();
}
```

Paper_Feed.c

```
#include "Paper_Feed.h"

#include "NXT_SCANNER.h"
#include "NXT_SCANNER_private.h"

void task_stop_Start(rtB_task_stop *localB)
{
  localB->PWM = 0;
  localB->Constant2 = FALSE;
}

void task_stop(rtB_task_stop *localB)
{
  localB->PWM = 0;
  localB->Constant2 = FALSE;
}

void Paper_Feed_Init(void)
{
  rtDWork.RotDir = false;
}

void Paper_Feed_Disable(void)
{
  rtDWork.Counter_For_Error_MODE = SUBSYS_DISABLED;
}

void Paper_Feed_Start(void)
{
  rtB.PWM_f = 100;
  rtB.Constant1_c = FALSE;
  rtB.Constant2_m = TRUE;
  rtB.PWM_o = -100;
  rtB.Constant1 = TRUE;
  rtB.Constant2_i = TRUE;
  task_stop_Start(&rtB.task_stop_n);
  rtB.PWM_c = 100;
  rtB.PWM_c = 100;
  rtB.Constant2_l = TRUE;
  rtB.PWM_l = 100;
  rtB.PWM_l = 100;
  rtB.PWM_l = 100;
  rtB.Constant2 = TRUE;
  rtDWork.Memory1_PreviousInput = ((uint8_T)0U);
  rtDWork.UnitDelay_DSTATE = 0;
  rtB.ErrorCode_i = NO_ERROR;
}

void Paper_Feed(void)
{
  int32_T rtb_Switch1;

  {
    boolean_T rtb_ErrDetect;
    uint8_T rtb_Sum1_d;
    int32_T rtb_Switch;
    int32_T rtb_EncoderDiff_o;
    if (ControlMode == INITIALIZE) {
      if (FeedMode == FEED_INIT) {
        rtB.SFunction_o6 = rtB.RevolutionSensor_B;
        if (!rtDWork.ROTOR_FLAG) {
          rtDWork.ROTOR_INIT = rtB.SFunction_o6;
          rtDWork.ROTOR_FLAG = TRUE;
          rtB.PWM_l = 100;
        } else {
          if (rtB.SFunction_o6 - rtDWork.ROTOR_INIT < 1000) {
            rtDWork.ROTOR_FLAG = TRUE;
```

```
          FeedMode = FEED_INIT;
          rtB.PWM_l = 100;
        } else {
          rtDWork.ROTOR_FLAG = FALSE;
          FeedMode = FEED_IDLE;
          rtB.PWM_l = 0;
        }
      }

      rtB.Constant2 = TRUE;
      rtB.PWM = rtB.PWM_l;
      rtb_ErrDetect = rtB.Constant2;
    } else {
      task_stop(&rtB.task_stop_n);
      rtB.PWM = rtB.task_stop_n.PWM;
      rtb_ErrDetect = rtB.task_stop_n.Constant2;
    }
  } else if (ControlMode == PAPERFEED) {
    rtB.PWM_f = 100;
    rtB.Constant1_c = FALSE;
    rtB.Constant2_m = TRUE;
    rtB.PWM = rtB.PWM_f;
    rtDWork.RotDir = rtB.Constant1_c;
    rtb_ErrDetect = rtB.Constant2_m;
  } else if (ControlMode == PAPEREXHAUST) {
    rtB.PWM_o = -100;
    rtB.Constant1 = TRUE;
    rtB.Constant2_i = TRUE;
    rtB.PWM = rtB.PWM_o;
    rtDWork.RotDir = rtB.Constant1;
    rtb_ErrDetect = rtB.Constant2_i;
  } else if (ControlMode == IDLE) {
    task_stop(&rtB.task_stop_n);
    rtB.PWM = rtB.task_stop_n.PWM;
    rtb_ErrDetect = rtB.task_stop_n.Constant2;
  } else if (ControlMode == ERROR) {
    task_stop(&rtB.task_stop_n);
    rtB.PWM = rtB.task_stop_n.PWM;
    rtb_ErrDetect = rtB.task_stop_n.Constant2;
  } else if (FeedMode == FEED_PAPERFEED) {
    rtB.SFunction_o4 = rtB.RevolutionSensor_B;
    rtB.SFunction_o5 = rtDWork.RotDir;
    if (!rtDWork.ROTOR_FLAG) {
      rtDWork.ROTOR_INIT = rtB.SFunction_o4;
      rtDWork.ROTOR_FLAG = TRUE;
      rtB.PWM_c = 100;
      rtB.RotOffset = rtB.SFunction_o5;
    } else {
      rtb_EncoderDiff_o = rtB.SFunction_o4 - rtDWork.ROTOR_INIT;
      if (rtB.SFunction_o5) {
        rtb_Switch1 = rtb_EncoderDiff_o;
      } else {
        rtb_Switch1 = rtb_EncoderDiff_o + 50;
      }

      if (rtB.SFunction_o5) {
        rtb_Switch = 50;
      } else {
        rtb_Switch = 0;
      }

      if (rtb_EncoderDiff_o < rtb_Switch + 102) {
        rtDWork.ROTOR_FLAG = TRUE;
        FeedMode = FEED_PAPERFEED;
        {
          uint32_T iLeft;
          if (rtb_Switch1 <= 0 ) {
            iLeft = 0;
          } else if (rtb_Switch1 >= 150 ) {
            iLeft = 15U;
```

```
          } else {
            iLeft = (uint32_T)( rtb_Switch1 ) / 10;

            {
              uint32_T remainder;
              remainder = (uint32_T)( rtb_Switch1 ) % 10;
              if (( 10 - remainder ) <= remainder ) {
                iLeft++;
              }
            }
          }

          rtB.PWM_c = (rtConstP.LookupTable_YData[iLeft]);
        }

        rtB.RotOffset = rtB.SFunction_o5;
      } else {
        rtDWork.ROTOR_FLAG = FALSE;
        FeedMode = FEED_IDLE;
        rtB.PWM_c = 0;
        rtB.RotOffset = FALSE;
      }
    }

    rtB.Constant2_l = TRUE;
    rtB.PWM = rtB.PWM_c;
    rtDWork.RotDir = rtB.RotOffset;
    rtb_ErrDetect = rtB.Constant2_l;
  } else {
    task_stop(&rtB.task_stop_n);
    rtB.PWM = rtB.task_stop_n.PWM;
    rtb_ErrDetect = rtB.task_stop_n.Constant2;
  }

  if (rtb_ErrDetect) {
    if (rtDWork.Counter_For_Error_MODE == SUBSYS_DISABLED) {
      rtDWork.Memory1_PreviousInput = ((uint8_T)0U);
      rtDWork.UnitDelay_DSTATE = 0;
      rtDWork.Counter_For_Error_MODE = SUBSYS_ENABLED;
    }

    rtb_Sum1_d = (uint8_T)(uint32_T)(rtDWork.Memory1_PreviousInput + ((uint8_T)
      1U));
    if (rtb_Sum1_d > ((uint8_T)10U)) {
      rtb_Sum1_d -= ((uint8_T)10U);
    }

    rtb_ErrDetect = (rtb_Sum1_d == ((uint8_T)10U));
    if (rtb_ErrDetect && (rtPrevZCSigState.Error_Detection_Trig_ZCE !=
                          POS_ZCSIG)) {
      if ((rtDWork.UnitDelay_DSTATE == rtB.RevolutionSensor_B) &&
          (rtB.RevolutionSensor_B != 0)) {
        rtB.ErrorCode_i = ERROR_002;
      }

      rtDWork.UnitDelay_DSTATE = rtB.RevolutionSensor_B;
    }

    rtPrevZCSigState.Error_Detection_Trig_ZCE = rtb_ErrDetect ? POS_ZCSIG :
      ZERO_ZCSIG;
    rtDWork.Memory1_PreviousInput = rtb_Sum1_d;
  } else {
    if (rtDWork.Counter_For_Error_MODE == SUBSYS_ENABLED) {
      rtDWork.Counter_For_Error_MODE = SUBSYS_DISABLED;
    }
  }

  ErrorCode = rtB.ErrorCode_i;
 }
}
```

参考文献

[1] Embedded Coder Robot NXT
http://www.mathworks.com/matlabcentral/fileexchange/13399

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

[3] NXT GamePad
http://lejos-osek.sourceforge.net/utilities.htm

[4] Excel Interface API for Simulink Data Object
http://www.mathworks.com/matlabcentral/fileexchange/20316

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

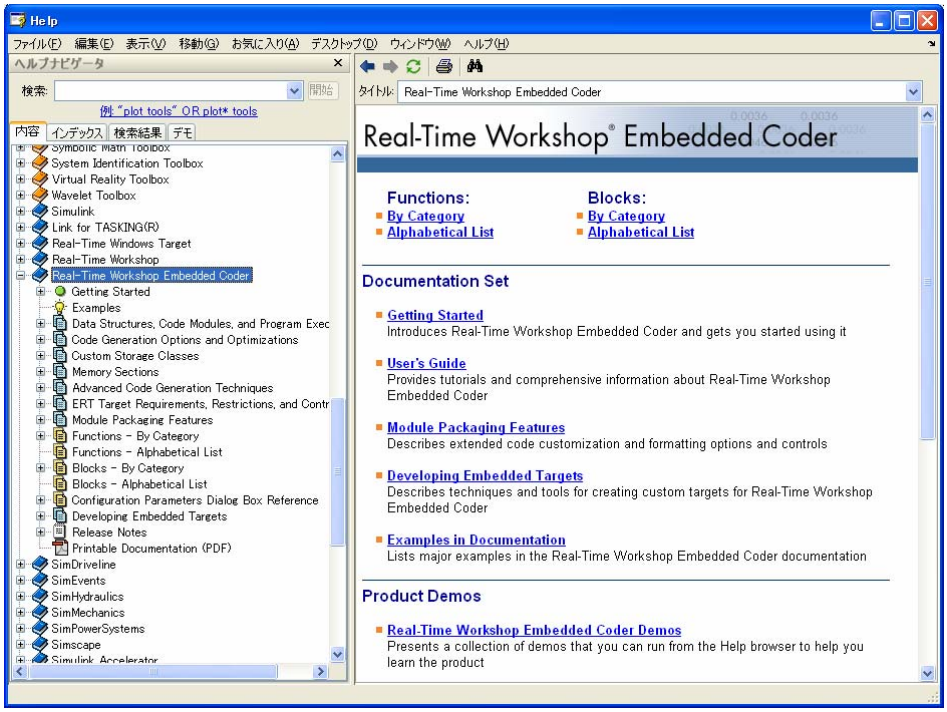
```
>> help 関数・コマンド名
       or
>> doc 関数・コマンド名
```

■ 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

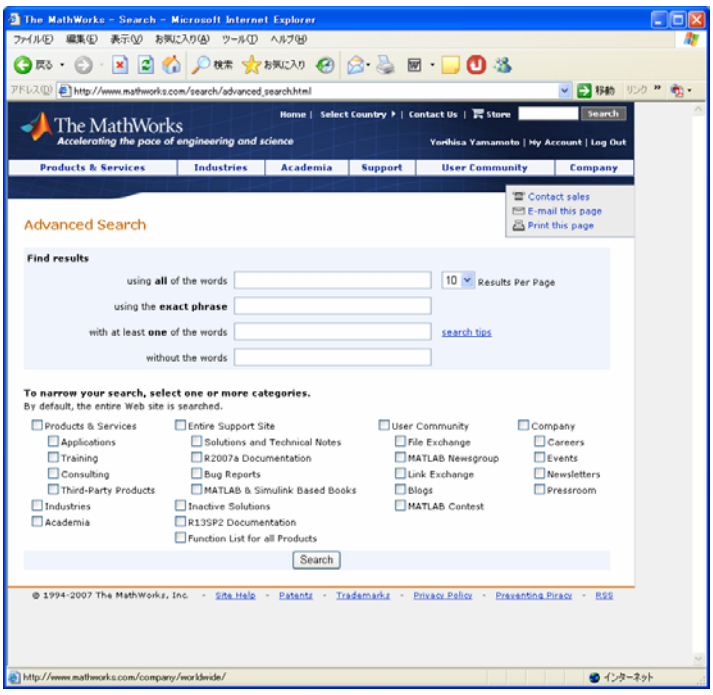
http://www.cybernet.co.jp/matlab/

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。

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