大規模・複雑化した組込みシステムのための障害診断手法

モデルベースアプローチによる事後 V&V の提案

付録

（Ver. 2.0）

2016 年 3 月

独立行政法人情報処理推進機構
Information-technology Promotion Agency, Japan

はじめに

モデルベース開発を行うことで、各工程で実環境に依らずに仕様や動作確認ができ仕様ミスや不具合の早期発見・解消が可能になる。大規模・複雑化するシステム、ソフトウェアをモデル化して記述し、モデル上で環境や条件の変更を加えてシミュレーションすることで障害発生時にその原因の迅速な分析に資すると期待できる。

当 WG では、提案する事後 V&V フレームワークに沿った障害原因診断手順をモデル上で実際に確認し、それによって障害原因を推定するためのサンプルシステムとなるシミュレーターを開発している。本付録で、このサンプルシステムについて解説する。

シミュレーターを用いた実行によって、ハザード分析で得た障害原因の仮説シナリオをモデルで検証し、そのシナリオに沿った故障注入やオペレータの操作を再現すること、また、不具合の修正によって事故を回避できることをモデルで事前確認すること、などが実施できる。

2014 年度のタンク水位制御を行う化学プラントシミュレータに引き続き、今年度はロボット制御のシミュレーターを開発して下記仕様を実現している。

- 設計（モデル動作）と実機動作との差異の監視
- オペレータが介入するシステム動作の模擬
- 事故原因の改修による効果の確認

目次

図表目次

付録

A　二輪倒立ロボット

ここでは事後 V&V 要素技術の検証のためのサンプルシステムとして作成した、二輪倒立ロボットのモデルについて説明する。二輪倒立ロボットとは、ちょうど振り子を逆向きにした形のもので、ジャイロセンサーと二つのモーター、およびモーターに装着されている車輪と、値取得やモーター制御を行うコントローラーから構成される。二輪倒立ロボットは何もしないと倒れてしまうが、車体の傾きをジャイロセンサーから取得し、傾きを打ち消すために必要なパワーをコントローラーで計算、その結果をモーター動力として車輪に伝えることで自立を実現している。

このように、ハードウェア（センサーやモーター）とソフトウェア（コントローラー中ロジック）とが連携して動作しているロボットに対し、本事例ではさらにユーザー操作機能と環境の変化を追加する。変化する要素が複数存在するシステムにおいて障害注入することにより、本事例は事故の設定とその検証を行うための教材としての用途を目的としている。

本事例では、ET ロボコン等でも利用されている LEGO 社の Mindstorms NXT および EV3 を用いて二輪倒立ロボットを組み立て、上述の制御には MathWorks 社の MATLAB/Simulink を用いている。以下でその詳細を記述する。

A-1　ロボットについて

図 A-1-1　NXT(左)と EV3(右)全体図

(1) ロボットの構成

二輪倒立ロボットは、ジャイロセンサーと二つのモーター、及びモーターに装着される車輪とコントローラーから構成される。本事例では、さらに物体との距離を測る超音波センサー、輝度を認識する光センサーを追加している。これは後述する、ライントレース及び衝突回避を実現するためのものである。

これらの組み立てには LEGO 社 Mindstorms NXT 及び EV3 を用いている。組み立て設計図は ET ロボコン向け設計図を参考にしているが、後部のロボットを支えるモーター、所謂「尻尾」をはずした構成としている。組立て後のロボットを図 A-1-1 に示す。

(2) OS

NXT および EV3 にて動作させる OS を以下に示す。NXT の場合、OS は後述のツールボックスにて NXT に対して OS の適応が行われるため、別途 OS の準備は不要である。EV3 の場合、別途ツールを用いるため、LEGO 社のホームページ[1]から OS およびインストール用ツールを入手する。

NXT：NBC/NXC 1.28

EV3：BrickOS v1.08H

(3) PC との通信

PC との通信には NXT では Bluetooth、EV3 では無線 LAN をそれぞれ用いている。これは後述する開発環境の制約によるものである。モジュールの選定および設定の概要については、次節で述べる。

A-2　ロボットの要求仕様

(1) 自立制御

ロボットの電源がオンで、制御ソフトが動作しており、かつ自立制御が有効な場合に、ロボットは自身が倒れないように制御を行う。ジャイロセンサーから得る車体角速度をもとに、ロボットの傾きを打ち消すようモーターに対して回転要求を出す。あわせて、モーターのエンコーダーから得る車輪回転角を元に、ロボットが元の位置に戻るようモーターに対して回転要求を出す。上記二つの要求により、ロボットは自立制御有効時、元の位置で倒れないように自立する。

(2) ユーザーによる操作

ロボットの電源がオンで、制御ソフトが動作しており、かつ自立制御が有効である際に、ユーザー向け UI パネルから前後それぞれの移動、左右それぞれの旋回、およびこれらの停止の命令を受け付ける。前後それぞれの移動の命令を受けると、ロボットは指定された方向に 1000ms 移動する。また、左右それぞれの旋回の命令を受けると、ロボットは指定された方向に 3500ms 旋回する。これらは、左右のモーターに対しての目標回転角をそれぞれ変更することで実現する。

(3) ライントレース

ロボットの電源がオンで、制御ソフトが動作しており、自立制御が有効かつライントレースが有効である際に、ロボットは、地面の黒線を追いながら走行する。ロボット下部に設置された光センサーにより地面の輝度を計測し、そこから黒線か否かを判断する。黒線を検知した際は右に、検知できなかった場合は左に旋回することで、ロボットは黒線の右端を辿る。またライントレースが有効である場合、ロボットは常に前進する。ライントレースが有効である場合、上述のユーザーによる操作を受け付けない。

[1] http://www.lego.com/en-us/mindstorms/?domainredir=mindstorms.lego.com

(4) 障害物検知/回避

ロボットの電源がオンで、制御ソフトが動作しており、自立制御が有効かつ障害物検知が有効である際に、ロボットは前方に障害物を見つけると、その距離に応じて二つの動作をそれぞれ行う。ユーザーによる操作、およびライントレースによる前進命令があり、かつ 23cm 以内に物体を検知するとロボットは前進命令をキャンセルし、その位置にとどまるよう動作する。10cm 以内に物体を見つけると、ロボットは衝突回避のため一定速度で後退する。ロボット上部に設置された超音波センサーにより、常時前方との距離を測りながら、上記処理を行う。この 10cm 以内に物体を見つけた際の処理は、ユーザーによる操作、およびライントレースによる前後それぞれの移動命令より優先される。

(5) ロボットの情報出力

ロボットの電源がオンで、制御ソフトが動作している際に、ロボットは自身のディスプレイにセンサー値や PID 計算結果を逐次出力する。

(6) PC による監視

ロボットの電源がオンで、制御ソフトが動作している際に、ロボットは PC に対して常時センサー値を初めとする各種データを PC に転送する。PC 側では受信したデータの表示と制御ソフトの一部設定値の書き換えを行うことができる。また、受信したデータをもとにロボット側の異常を検知する機構を備える。この機構は、SPRT(逐次確率比検定)による分析を行うものである。

A-3　ロボットの動特性モデル

二輪ロボットの動特性は、Mathworks 社の NXTWay-GS [1] の記述を参考としている。動特性モデルから PID 計算に用いる各係数が得られる。NXT は本参考資料と動特性が同等のものであるが、EV3 の場合は NXT との質量の差、ツールボックスの動作速度の関係より、同一パラメータでは自立しない場合があった。そこで、新たに限界感度法等を用いて安定動作する係数値を求めている。

(1) 開発環境

本事例では、ロボットの制御プログラムの作成、および動作時の常時監視を同一環境で行う。これは、MATLAB/Simulink および Simulink toolbox for LEFO Mindstorms NXT/EV3 により実現している。本事例で用いたソフトウェア環境は以下のとおりである。

MATLAB/Simulink 2015b

- Simulink toolbox for LEGO Mindstorms NXT
- Simulink toolbox for LEGO Mindstorms EV3

各ツールボックスの役割の詳細を下記する。

(2) Simulink toolbox for LEGO Mindstorms NXT

Simulink に対して LEGO Mindstorms NXT 向けモデリング、及びエクスターナルモードによる実行を実現するツールボックスである。エクスターナルモードによる実行により、モデル実行時に PC

[1] http://www.mathworks.com/matlabcentral/fileexchange/19147-nxtway-gs--self-balancing-two-wheeled-robot--controller-design

による NXT の各種データの取得、及び PC からのパラメータ変更を可能としている。本ツールボックスの制約により、エクスターナルモードの実施には PC と NXT 間での Bluetooth 通信が必要となる。本事例では、PC 側端末として、ロジテック社製 LBT-UAN0C1 を用いている。

※PC 側端末は Bluetooth class1 v4.0 であるが、NXT 側は Bluetooth class2 V2.0 [1] である為、通信は class2 V2.0 で行われる。

(3) Simulink toolbox for LEGO Mindstorms EV3

Simulink に対して LEGO Mindstorms EV3 向けモデリング、及びエクスターナルモードによる実行を実現するツールボックスである。

エクスターナルモードによる実行により、モデル実行時に PC による EV3 の各種データの取得、及び PC からのパラメータ変更が可能となる。ツールボックスの制約により、エクスターナルモードの実施には PC と EV3 間での無線 LAN 通信が必要となる。この際、EV3 側の無線 LAN 対応モジュールが限定されている点に注意が必要である [2]。さらに、EV3 側の OSversion は 1.08H である必要がある [3]。

PC 側では無線 Lan モジュールとして buffalo WLI-UC-GNM2 を利用している。本ツールボックスにて無線 Lan を利用する場合は、アドホック接続が不可能で、DHCP による IP アドレス割り当てが必要であり、無線ルーターが別途必要となる。また、通信時の暗号化は EV3 の制約から WPA2 若しくは非暗号化による通信のみ可能である。今回は、Windows7 の SoftAP を用いて無線ルーター機能を実現した。

A-4　モデルの構成

ここでは事後 V&V 適応例として作成した、二輪倒立ロボットのモデルについて説明する。まずロボット中で動作する Simulink モデルを示す。次に、ロボットへの操作命令、およびロボットからの情報受信を行う PC 側モデル、並びに UI パネルの構成を示す。

A-4.1 ロボット側 Simulink モデル

Simulink を用いて作成した二輪ロボットのモデルの全体構成を図 A-4-1 に示す。

---

[1] http://www.generationrobots.com/media/Lego-Mindstorms-NXT-Education-Kit.pdf

[2] EDIMAX EW-7811Un の他、Planex GW-USVALUE2 にて動作を確認している。両者とも搭載チップは Realtek RTL8188CUS ではあるが、同チップを搭載している Planex GW-USNANO2 では本事例環境では動作不可であった。

[3] EV3 の OSversion が 1.07H 以下の場合、対応無線 LAN モジュールが販売終息品の 1 種類のみとなる。また、1.09H にて EV3 側で SSH, telnet 通信をブロックするように変更されている（リリースノート非公開のため推測である）。これが原因で、ping は通るがツールボックスと EV3 との通信ができない現象が生じる。

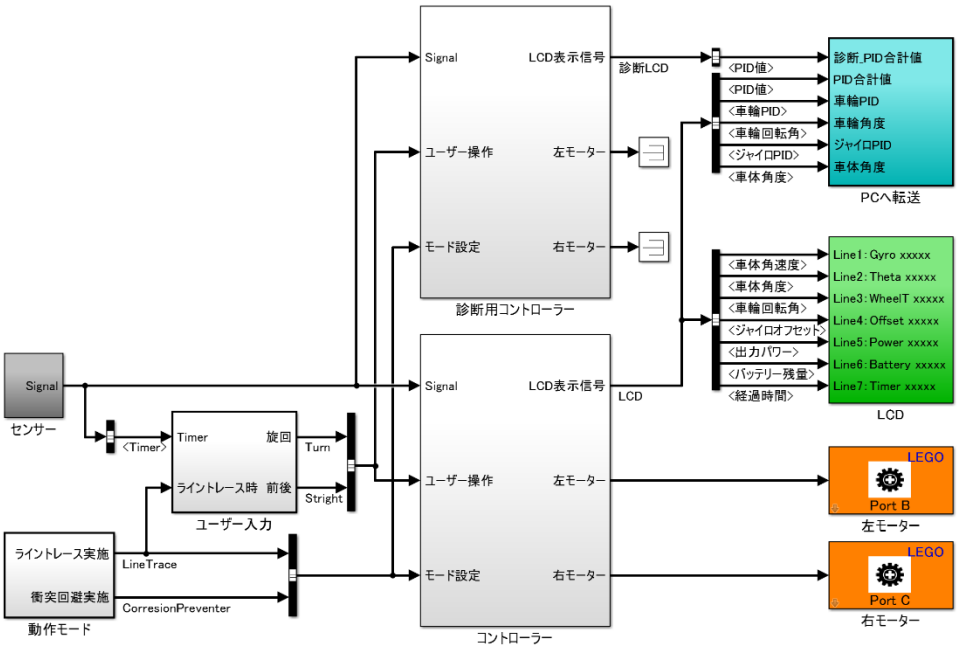

図 A-4-1　NXT 版二輪倒立ロボット Simulink モデル 全体構成

モデルは、ロボットのセンサー値を取得する「センサー」部と、センサー値から自立制御するための動力を計算する「コントローラー」部、及び計算された動力をロボットのモーターへ伝える「モーター」から構成される。さらにはロボットのディスプレイ画面及びPCへ情報を伝達するための「LCD」、「PC へ転送」部のほか、ユーザー操作を受ける外部入力が存在している。これ以外に図中には診断用コントローラーというサブシステムが存在している。これは後述する SPRT による分析を行うためのコントローラーであり、内部構造はコントローラー部と共通である。

なお、EV3 では NXT に比べて LCD 中の項目が 2 項目多い（図 A-4-2 赤枠参照）。これは EV3 の LCD では NXT の LCD に比べより多くの情報が表示可能であることから、表示対象の項目を増やしたものである。

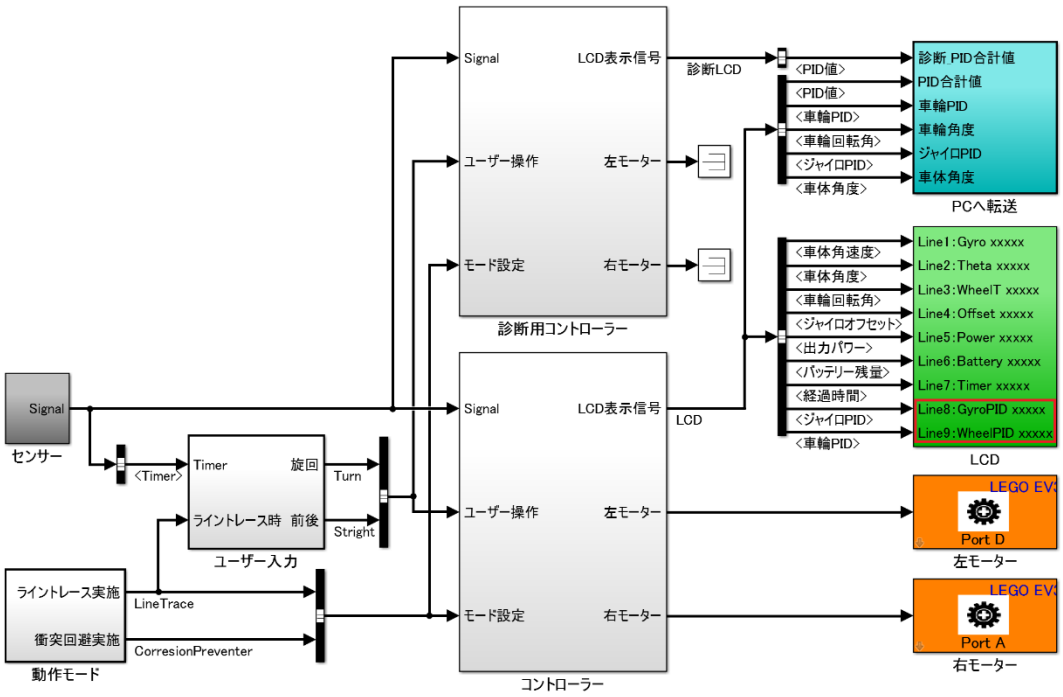

図 A-4-2　EV3 版二輪倒立ロボット Simulink モデル 全体構成

(1) モデルコンフィグレーションパラメータ

図 A-4-3 に示すように、本モデルでは NXT、EV3 共通してソルバーを「固定ステップ」「離散(連続状態なし)」、基本サンプル時間を auto としている。NXT ではツールボックスの制約から周期的なサンプル時間のタスクモードを「シングルタスク」としている。一方 EV3 では「マルチタスク」としている。また、EV3 では動作負荷が NXT に比べて重い現象が見られたため、確定的なデータ転送を「保障しない(最小遅延)」としている。

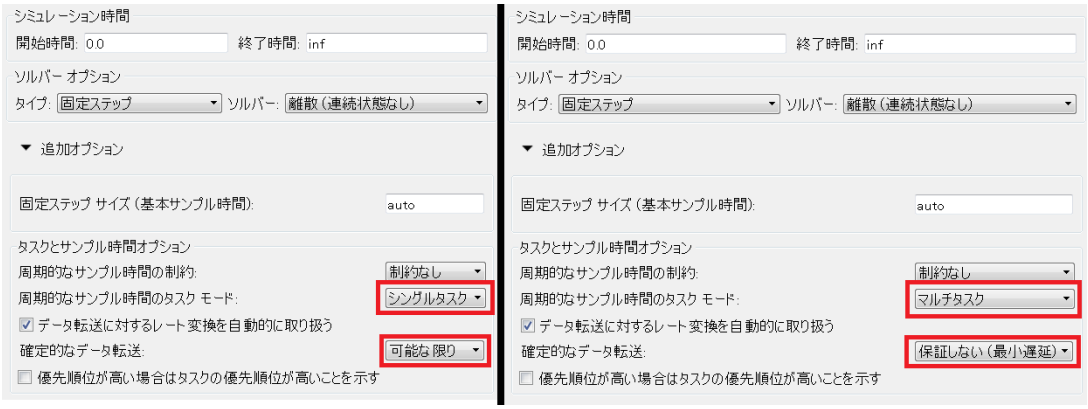

図 A-4-3　NXT(左)/EV3(右)モデルコンフィグレーションパラメータ

(2) センサー部の構成

センサー部は図 A-4-4 に示すように、ロボットに存在する各種センサーに対応するブロックから構成される。

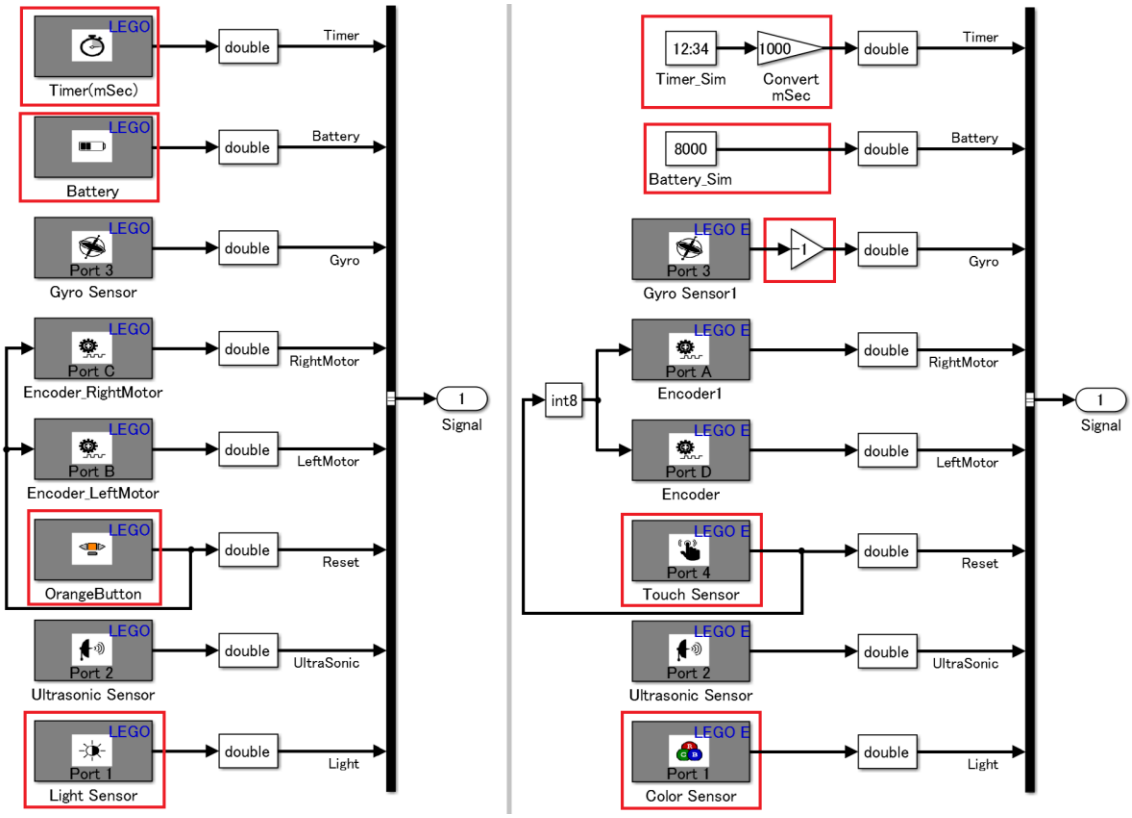

図 A-4-4　NXT(左)/EV3(右)センサーモデル

NXT と EV3 では対応するセンサーが異なるため、それぞれ異なるブロックから構成されている。特に NXT では取得できる時間、電源電圧が EV3 では取得できないため、ここではそれぞれモデルのシステム時間、固定値を用いている。その他、EV3 では動作の開始/終了のスイッチとしてタッチセンサーを用いている点、NXT での光センサーが EV3 ではカラーセンサーに変わっている違いがある。また、ジャイロセンサーの取り付け方向の問題から、EV3 ではジャイロセンサー値に-1 をかけている。

自立制御に用いるジャイロセンサーとエンコーダー、及びライントレースに用いる光センサーは 10ms で動作するよう設定している。ただし、物体検知に用いる超音波センサーは、その特性から 10ms では誤動作する可能性がある。超音波センサーは仕様として 250cm（2.5m）まで物体を検知する。超音波センサーはセンサーから出た超音波が帰ってくる時間からその距離を算出するため、2.5m 地点の物体を検知するには、超音波は倍の 5m を移動する必要がある。音速を 340m/s とした際、約 15ms かかる計算である。センサー値取得はこれよりも遅い間隔で行う必要があるため、動作周期は余裕を持って 100ms としている。これ以外に、時間値はユーザー入力の基準、及び表示にのみ用いるため、100ms としている。

電源電圧の動作周期は NXT と EV3 で異なる。NXT では変化する電圧値からモーターへの出力値のキャリブレーションを後述のコントローラー部で行っている。これは、動作時間に応じて電源残量が減ることを考慮したものである。即、電圧値の局所的な変化量ではなく、長時間で見た場合の変化量を反映させればよいため、動作周期は 1s としている。EV3 では電源電圧の値を取得できないため、定数値を代わりに用いている。こちらは変化しないため、動作周期は inf（変化しない）としている。

(3) コントローラー

コントローラー部の構成を図 A-4-5 に示す。図中の各ブロックについてその詳細を以降記載する。

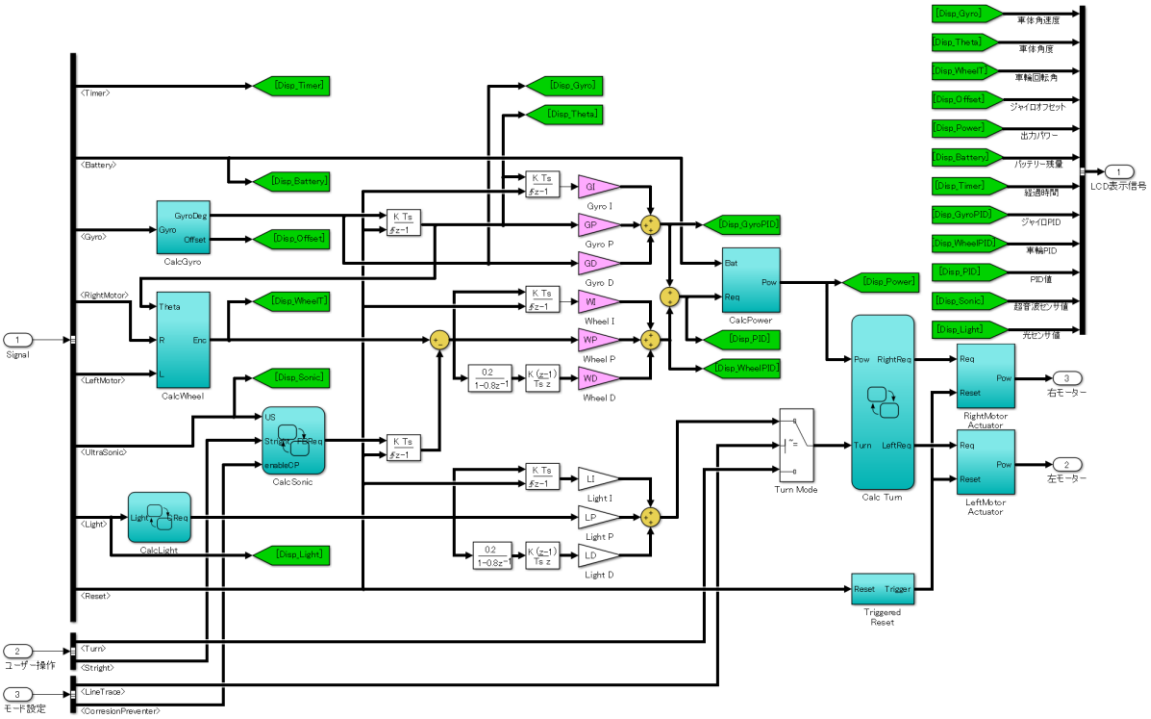

図 A-4-5　NXT/EV3 共通コントローラーモデル

(4) ジャイロ補正

図 A-4-6 に示すモデルではジャイロセンサーから得られた値に対して次の処理を行う。NXT の場合はセンサーから得られる値は0～1024の範囲で変化する。この際、角加速度が発生していないとき、即ちロボットが静止している際に得られる値は(センサーの個体差があるが)614 前後となる。本 CalcGyro 部では、この 614 前後の値をオフセット初期値として扱い、以後センサーの平均値からオフセット値の更新を行う。このオフセット値、及びオフセット値との差分を計算された角加速度として出力する。EV3 の場合はセンサーから得られる値は-480～480 の範囲で変化する。ロボットが静止している際に得られる値は 0 前後となる。したがって、EV3 の場合はオフセット初期値を 0 としている。

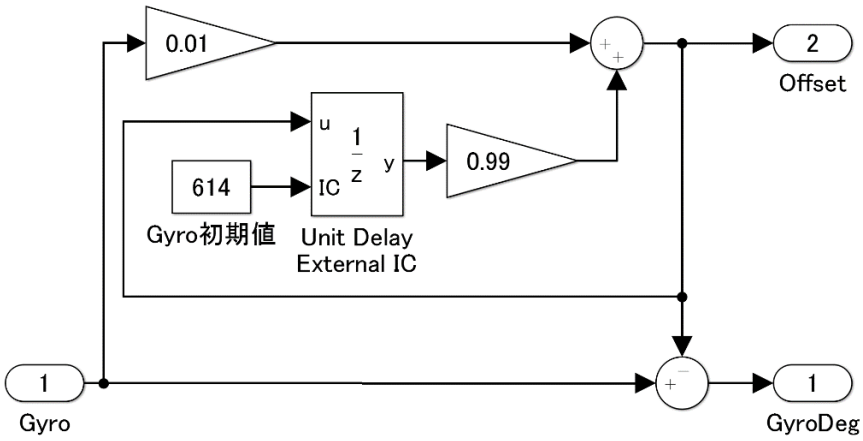

図 A-4-6　NXT/EV3 共通 CalcGyro モデル

(5) 車輪回転角計算

図 A-4-7 に示すモデルでは左右モーターのエンコーダーから得られた値、及び車体角度(これは前述のジャイロ補正部の出力で得られた角加速度を積分したもの)に対して次の処理を行う。車輪角度は、車体の傾きと車輪の回転との合計値であるため、左右のエンコーダー出力結果に対してそれぞれ車体角度を加算する。得られた左右のエンコーダーと車体角度との合計値の平均を出力する。

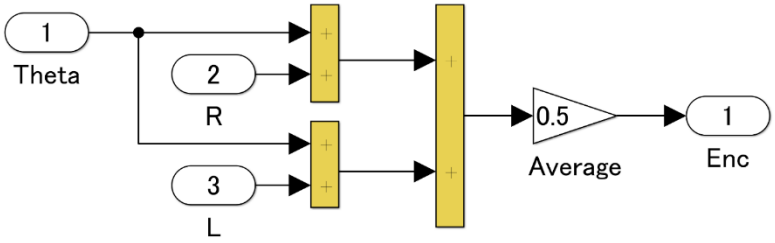

図 A-4-7　NXT/EV3 共通 CalcWheel モデル

(6) PID 計算

図 A-4-8 に示すモデル右側の GyroP/GyroI/GyroD 及び WheelP/WheelI/WheelD 部では前述のジャイロ補正、車輪計算の結果値をそれぞれ PID 計算するものである。各パラメータは ET ロボコンで使用されるパラメータを参考に、限界感度法を用いてより安定するパラメータを取得し設定してある。これにより、車体角度/角加速度が 0 になるように、及び車輪角度が指定値になるようなモーターへの要求が計算される。

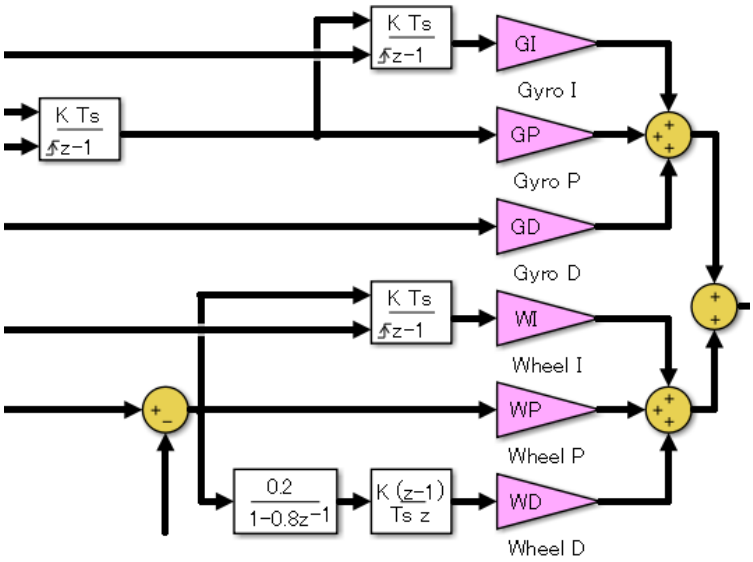

図 A-4-8　NXT/EV3 共通 PID 計算部モデル

(7) パワー計算

図 A-4-9 に示すモデルでは、PID 計算で得られた値に対して次の三つの処理を行う。ここでの得られた値とは、ジャイロ側の PID 計算結果と車輪側の PID 計算結果との合計値である。一つ目は、バッテリーの値をフィルター処理し、ノイズ成分を除いた現在の電圧値を取得する。二つ目は、PID 計算で得られた値と電源電圧値から、モーターに求める出力値を算出する。これにより、電圧値の変化の影響を最小限にした動作を実現できる。最後にロボットの設置面との摩擦係数を加味して、モーターに与える値を-100～100 の範囲で出力する。

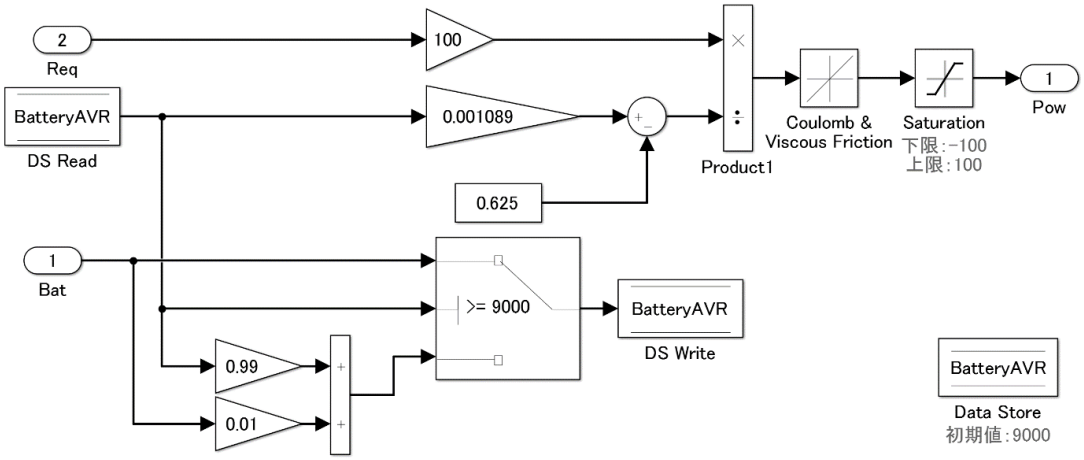

図 A-4-9　NXT/EV3 共通 CalcPower モデル

(8) 旋回処理

図 A-4-10 に示すモデルでは、パワー計算で得られた値及び旋回指示値により次の処理を行う。右モーターに対してはパワー計算で得られた値に旋回指示値を加算したものを右モーターへの要求値として出力する。左モーターに対してはパワー計算で得られた値に旋回指示値を減算した値を左モーターへの要求値として出力する。

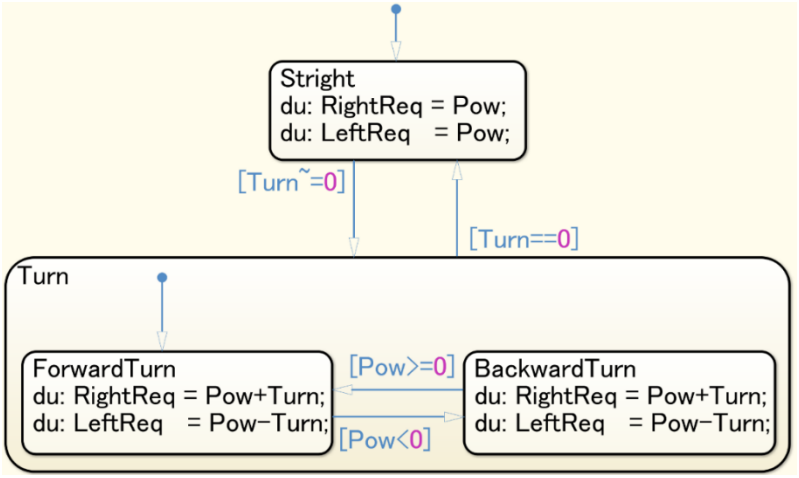

図 A-4-10　NXT/EV3 共通 CalcTurn モデル

(9) アクチュエーター

図 A-4-11 に示すモデル中の RightMotorActuator(右側)及び LeftMotorActuator(左側)部では、旋回処理で得られた右モーターへの要求値及び左モーターへの要求値に対して次の二つの処理を行う。一つ目は、ロボットの自立制御が有効か否かでモーターに対して動力を伝えるか否かの分岐を行う。二つ目は、左右のモーターの特製差に応じて、伝える動力に乗数をかけている。今回用いたロボットでは、EV3 では左右のモーター差がほぼなかったため乗数は 1 としているが、NXT ではモーターの差があったため、右モーターのアクチュエーターでは内部乗数として 1.xx をかけている。最後にこれらの出力値を-100～100 の範囲に制限して出力する。

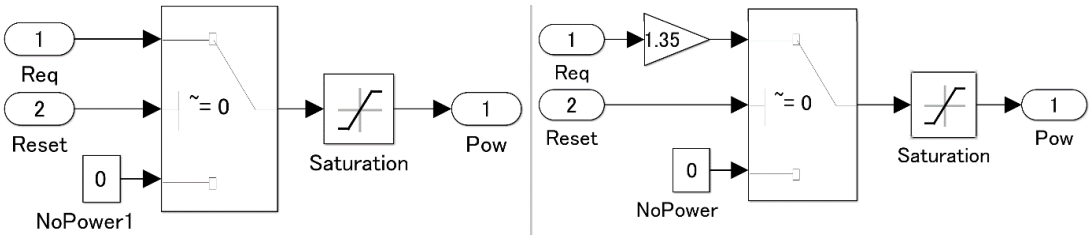

図 A-4-11　NXT/EV3 共通 Left(左)/Right(右)MotorActuator モデル

(10) 物体との距離に応じた処理

図 A-4-12 に示すモデルでは、超音波センサーの出力値及び入力として与えられた車輪の角速度要求に対して次の二つの処理を行う。まず、障害物検知が有効か否かで、入力値をそのまま出力するか障害物回避処理を加算した結果を出力するかの分岐を行う。次に、障害物検知が有効である場合、超音波センサーの出力値に応じて①入力として与えられた車輪の角速度要求をそのまま出力する、②入力として与えられた角速度要求がゼロより大きい場合はゼロを出力する。そうでなければ入力として与えられた角速度要求をそのまま出力する。③入力として与えられた角速度要求がゼロより大きい場合は規定値(-60)を出力する。そうでなければ入力して与えられた角速度要求から規定値(-60)を減算した値を出力する。の動作を行う。①は超音波センサーの出力値が 23 以上の場合、即ち 23cm 未満に物体を検知していない場合に実施される。②の場合は超音波センサーの出力値が 23 以下 10cm 異常の場合、即ち 23cm 未満に物体を検知しているが、10cm よりは離れている場合に実施される。③の場合は超音波センサーの出力値が 10cm 未満の場、即ち 10cm 未満に物体を検知している場合に実

施される。①～③により、障害物検知が有効な際に、前方に障害物を検知していない、若しくは十分に遠い場合は通常の走行を行い、物体が近づいてきた際は前進をやめ、さらに近づいてきた際は障害物にぶつからないよう後退する動作を行う。

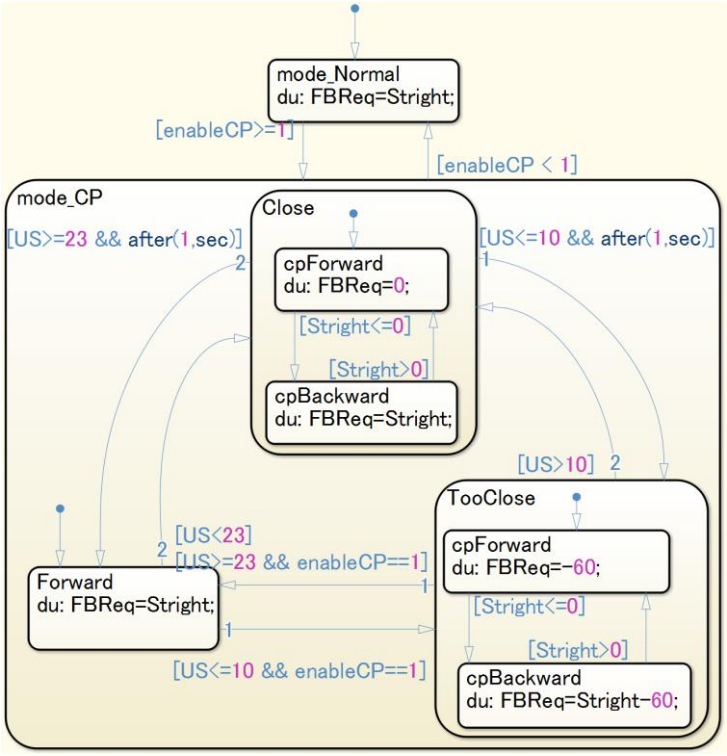

図 A-4-12　NXT/EV3 共通 CalcSonic モデル

(11) ライントレース

図 A-4-13 に示すモデルでは、光センサーの出力値に対して次の処理を行う。NXT の場合、光センサーの出力値は 0～100 で変化する。光センサーの出力値が規定値(NXT の場合は 45)以上の際は 1 を、そうでない場合は-1 を出力する。この出力値は後の旋回の判定に用いる。この規定値は事前に黒線とそうでない場所のセンサー出力値を取得しておき、十分に振り分けが可能なしきい値とする。これにより、ライントレース時に黒線を検知した場合とそうでない場合で旋回方向を反転し、黒線の縁を走行できるようになる。

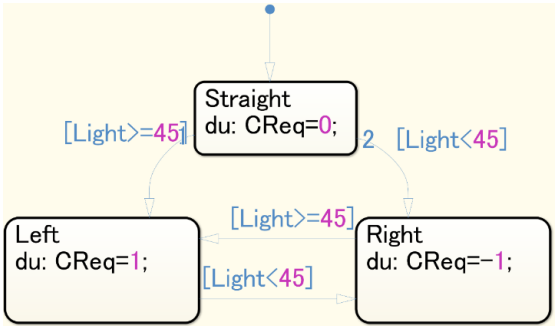

図 A-4-13　NXT/EV3 共通 CalcLight モデル

また、モデルの LightP/LightI/LightD 部は、光センサー計算結果を PID 計算するものである。短い間隔で黒線を検知できている場合、即ち停止時若しくは走行中と想定される場合は旋回指示を弱く

する。一方で、黒線の検知間隔が長くなっている場合、即ちカーブ中若しくは黒線を見失った状態と想定される場合は旋回指示を強くする。これにより、スムーズな走行を試みている。

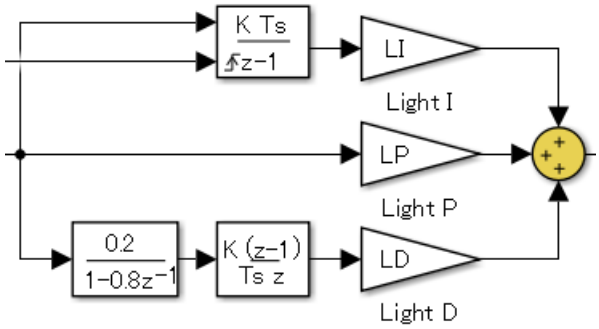

図 A-4-14　NXT/EV3 共通光センサーPID モデル

(12) モード変換

図 A-4-15 に示すモデルの Turn Mode 及び TriggerdReset では、次の処理を行う。まず Turn Mode 部では、ライントレースが有効か否かにより、前述の光センサーPID 計算結果を旋回処理に伝えるか、ユーザーからの旋回指示を旋回処理に伝えるかを切り替える。Triggered Reset 部では、ユーザーからのボタン押下指示があるたびに、自立制御の有効/無効を切り替える。

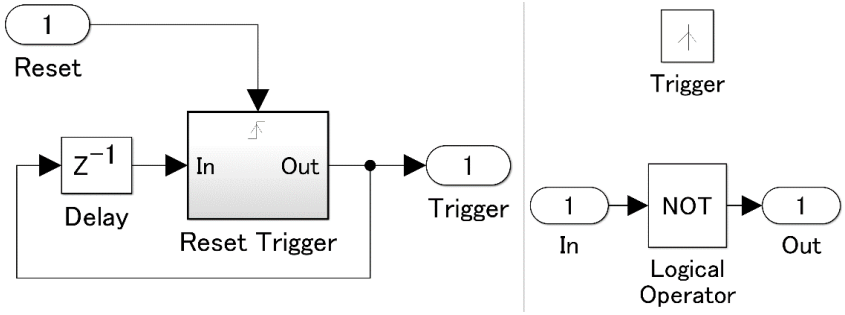

図 A-4-15 Triggered Reset(左)と Reset Trigger(右)モデル

(13) モーター

図 A-4-1 及び図 A-4-2 に示すモデルの右下に位置する左モーター、及び右モーターは、コントローラー部で計算された要求値をロボットのモーターに伝えるものである。値に応じて、回転方向および回転パワーが決定される。

(14) PC との通信

図 A-4-16 に示すモデルでは、コントローラー部及び診断コントローラー部で計算された LCD 表示信号の一部を PC へ転送する。ここでは、診断コントローラー部の PID 値(ジャイロ PID 値と車輪 PID 値との合計値)及び、コントローラー部の PID 値、ジャイロ PID 値、車輪 PID 値、車輪回転角及び車体角度を転送する。

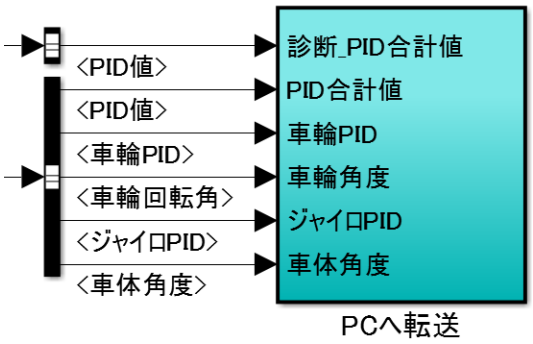

図 A-4-16　NXT/EV3 共通 PC へ転送モデル

(15) LCD 表示

図 A-4-17 に示すモデルでは、コントローラー部で計算された LCD 表示信号の一部をロボットに表示させる。NXT と EV3 ではハードウェアの違いにより表示できる項目数が異なる。共通して表示するのは車体角速度、車体角度、車輪回転角、ジャイロオフセット、出力パワー(CalcPower の出力値)、バッテリー残量および経過時間である。EV3 ではこれに加えて、ジャイロ PID 値と車輪 PID 値を出力する。

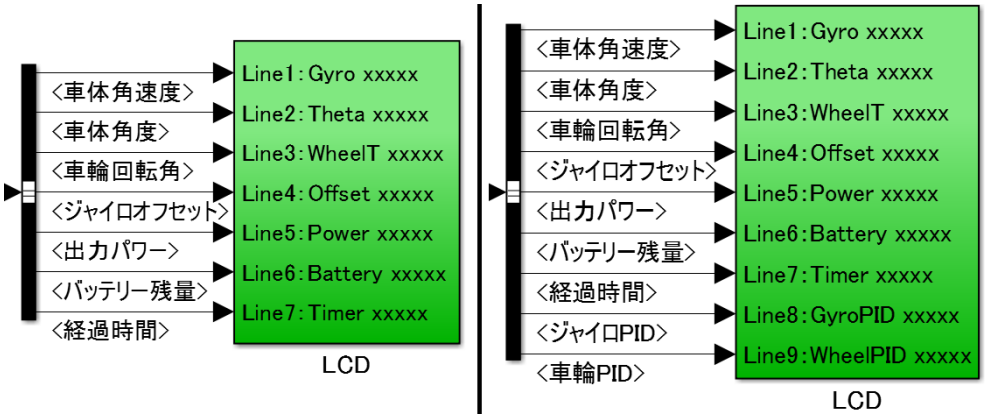

図 A-4-17　NXT(左)/EV3(右) LCD 表示モデル

A-4.2　PC 側 Simulink モデル

Simulink を用いて作成した PC 側処理のモデルの全体構成を図 A-4-18 に示す。モデルは、ロボットからの出力を受ける部分、出力結果から診断を行う診断モジュール部、及びその結果を表示する Scope から構成される。さらに、PC 側の処理速度をロボット側の処理速度にあわせるため、外部ライブラリの Real-Time Pacer を用いている。

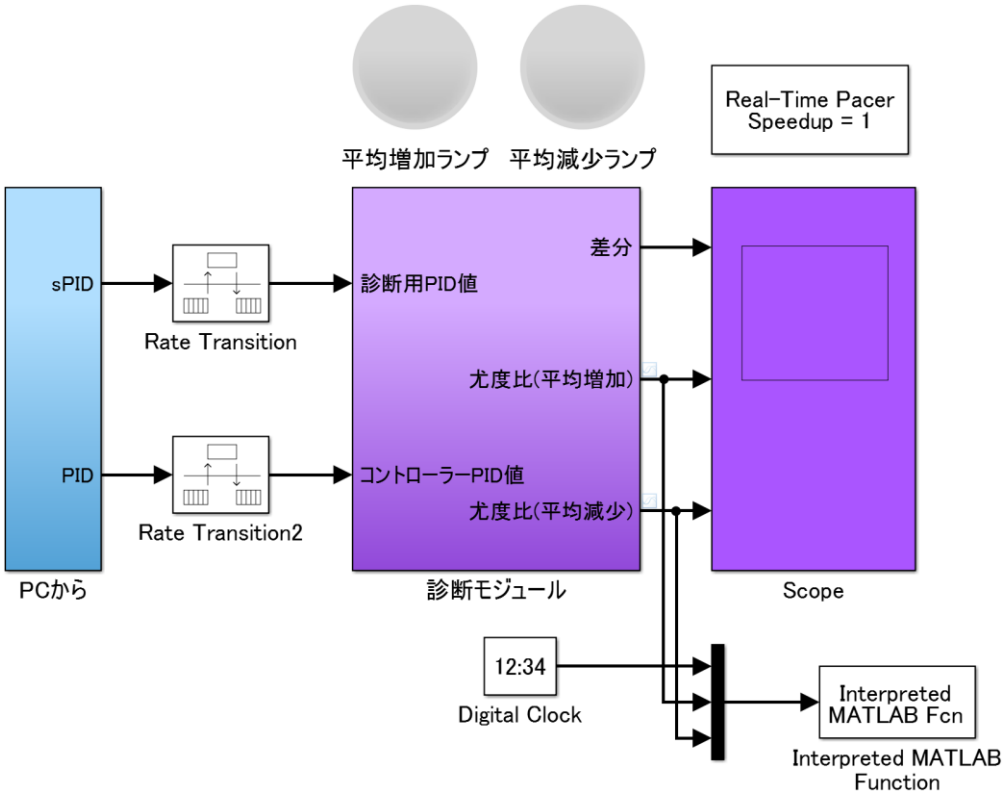

図 A-4-18　PC 側 Simulink モデル

(1) ロボットからの出力処理

図 A-4-19 に示すモデルでは、次の二つの処理を行う。ひとつは、ロボットからの出力を受ける。今回はロボットのモデルと PC 側とのデータ転送に MATLAB のワークスペースを用いる。ロボット側では逐次ワークスペースに値を書き込むのに対し、PC 側では逐次ワークスペースの値を読む処理を行っている。値を読み取った上で、PC 上に各データ表示を実現している。読み取ったデータのうち診断用コントローラーの PID 値とコントローラーの PID 値を出力し、以降の診断の計算に用いる。

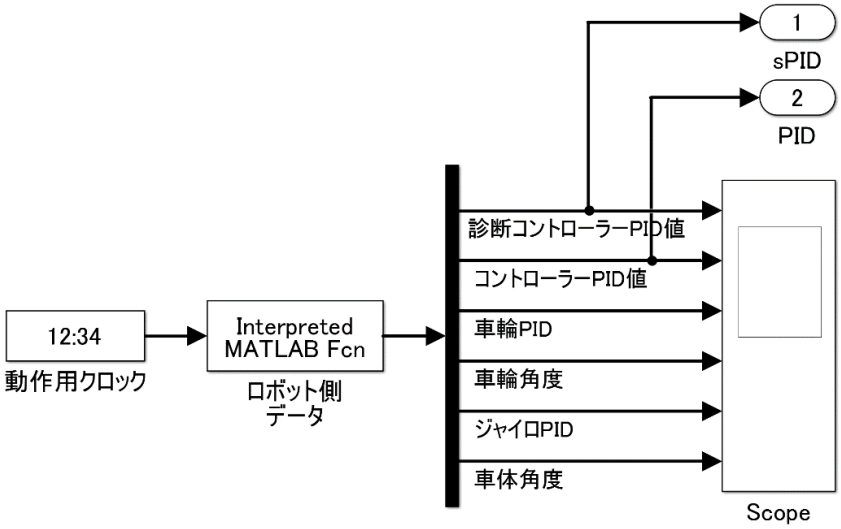

図 A-4-19　ロボットからの出力処理モデル

(2) 診断モジュール

図 A-4-20 に示すモデルでは、SPRT による判定を行う。SPRT については報告書本文 6.3 節を参照いただきたい。ロボットからの出力処理により得られた二つの値の差分から、平均増加、平均減少の尤度比をそれぞれ求める。誤検出率と非検出率については、事前にコントローラーにのみ不良挿入したモデルでのデータ取得を行い、そこから適切な値を割り出し適応させる。その結果は図 A-4-20 中の変数 A,B で用いられている。判断結果は正常な場合 1、判断保留の場合 0.1、異常な場合 0 とする。

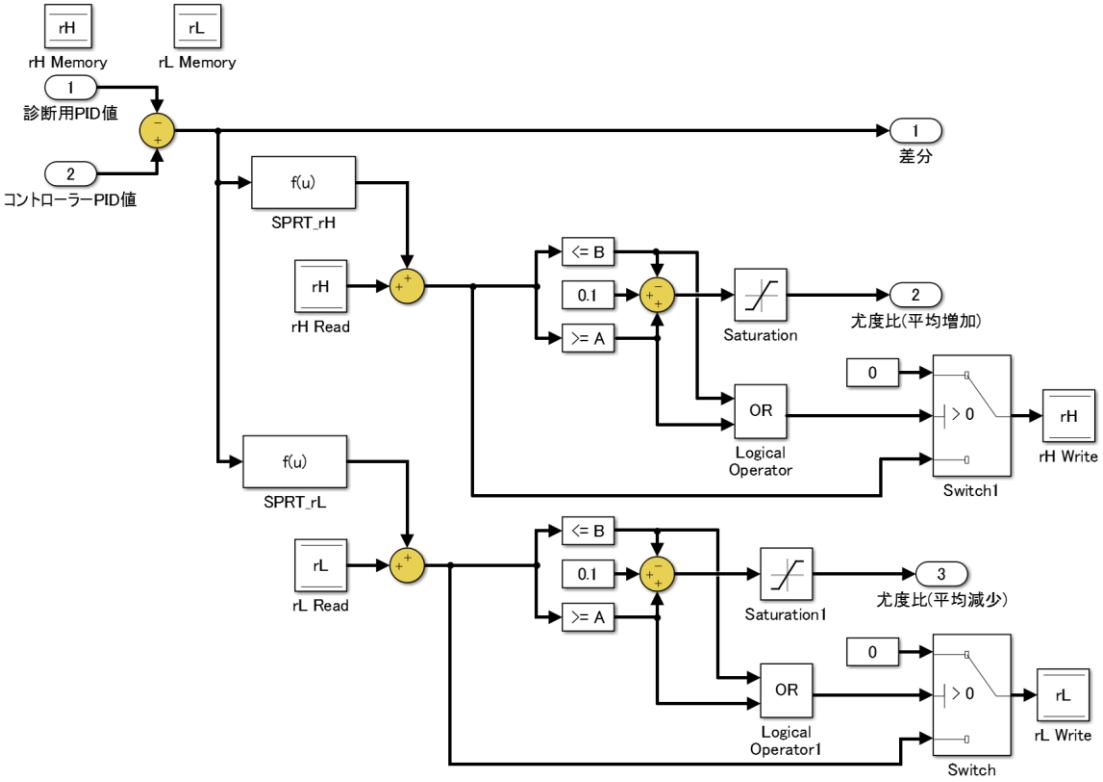

図 A-4-20　SPRT 診断モデル

A-5　UI パネル

図 A-5-1 に示す UI パネルではロボットへの操作指示、PID 制御係数のリアルタイム変更、及び動作モード変更が行える。

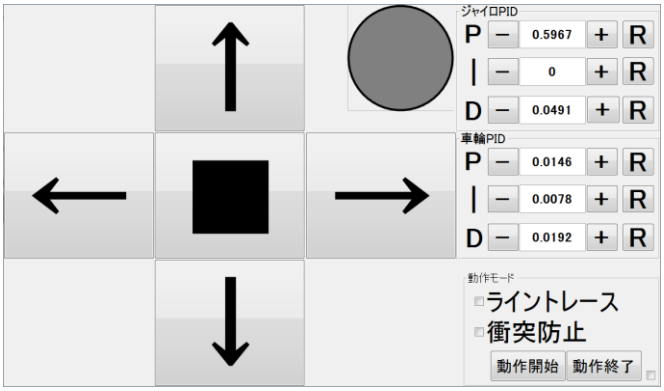

図 A-5-1　PC 側操作 UI パネル

(1) ロボットへの操作指示

UI パネル左側の上下左右の矢印、中央の■ボタンを押すことで、ロボットにそれぞれ前後の移動命令、左右の旋回命令、その場に停止(旋回もやめる)をそれぞれ行うことができる。

(2) PID 制御係数のリアルタイム変更

UI パネル右上にある車体 PID、車輪 PID のパラメータを変更することで、リアルタイムにロボットが用いる PID パラメータを変更可能である。初期値は本実験で安定して動作した値を設定している。左右のボタンで初期値の 10%の増減が行えるほか、数値入力によっても直接値を変更可能である。R ボタンは値を初期値に戻すものである。

(3) 動作モード変更

UI パネル右下にある項目のチェックボックスのチェックを切り替えることにより、それぞれライントレース、衝突防止を有効/無効に切り替えることが可能である。初期値はライントレース無効、衝突防止無効となっている。

(4) SPRT 判定結果表示

UI パネル中央上にある円状のランプでは、SPRT による診断を行っている際、判定結果を次のように表示する。

- 緑：正常
- 赤：異常
- グレー：判定保留(診断を行っていない場合も本表示となる)