TOPPERS 活用アイデア・アプリケーション開発コンテスト

部門	：	活用アイデア部門　　アプリケーション開発部門
作品のタイトル	：	Toppers_JSP と Scicos_lab / (Scilab でも可)による 組込みメカトロニクス制御シミュレーション
作成者	：	塩出　武（ シオデ　タケシ)
対象者	：	実機レス環境でモーター含むメカ制御プログラムの設計 および検証、学習をしてみたい方
使用する開発成果物	：	Toppers_JSP カーネル_Windows シミュレーション環境 VisualC++2010Express 版(Toppers 側シミュレーション用) Scicos_lab(制御用シミュレーション環境)

目的・狙い

・組込みメカ制御を始めてみたいが、機材、予算に制約がある。
・机上の制御理論を試してみたいが、初回動作不良による機材破損を回避するためにも事前に動作検証を行いたい。
・評価機材が上がってくる前に先行で制御プログラムの開発を進めたい。
・組込みによるメカ制御を学習、体験したい。

といった要求に対応するために、オープンソース環境による、メカ制御の設計/検証環境を提案します。(今回はモーター制御を例に挙げます)

１．アイデア/アプリケーションの概要

VisualC++Express による Toppers_JSP シミュレーション環境と、Scicos_lab によるモーターシミュレーション環境を DLL(ダイナミックリンクライブラリ)で接続し、Toppers 側に実装した割込み PID 制御プログラムにより Scicos 側のモーターモデルに対して、速度および位置決め制御をかけます。

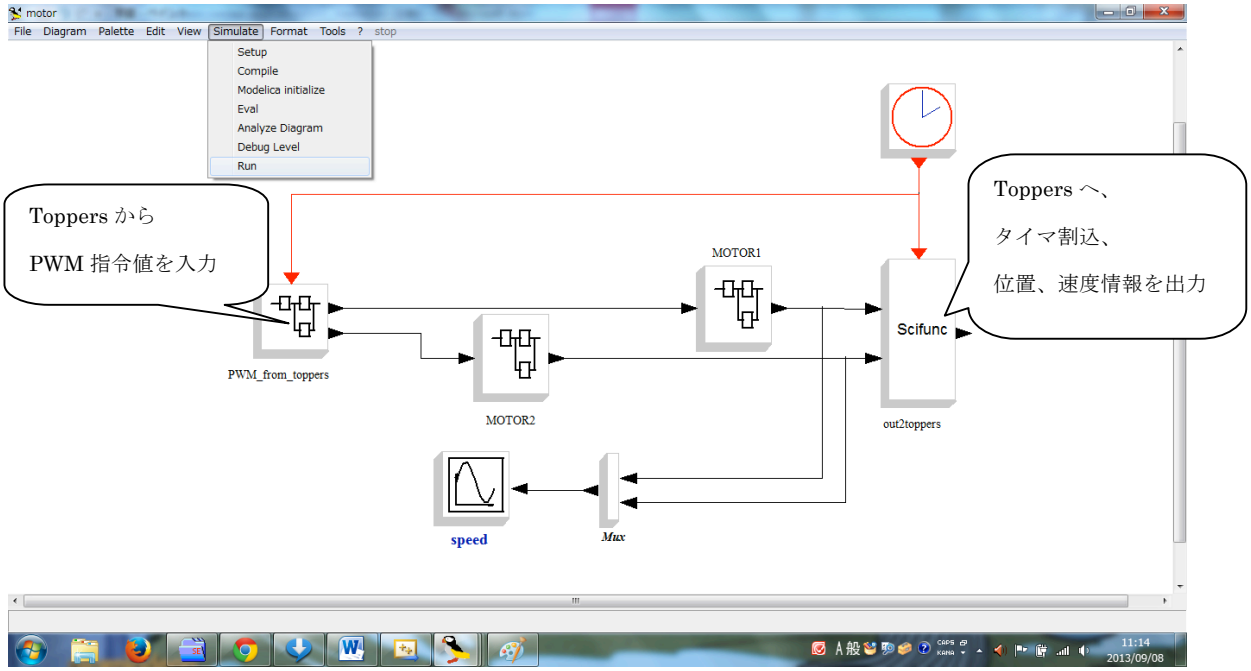


図１： Scicos 側のモータシミュレーションモデル

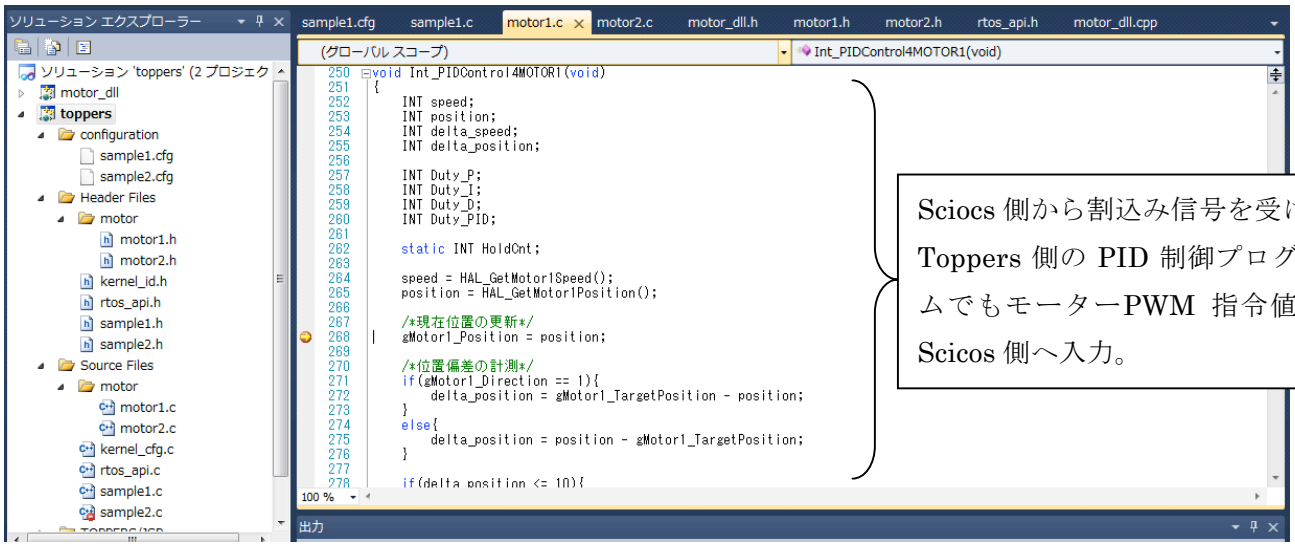


図２： Toppers 側のモータ制御割込みプログラム(Scicos 側より割込みを入れて制御)

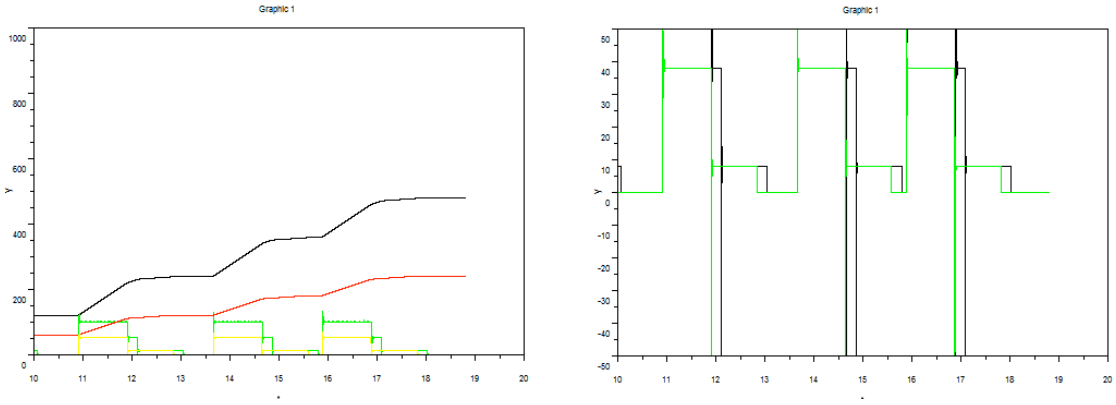


図３： PID 制御状態のモニタ(右は各モータの位置 / 速度 、右は PWM 出力)

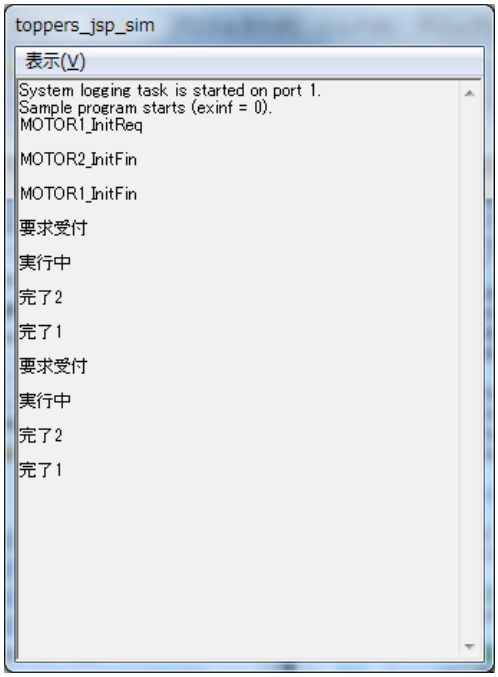


図４： Toppers 側のデバッグモニタ

１．１動作手順

①Scicos_lab を立上げ、ディレクトリを toppers_jsp の実行環境へ移動する。

¥toppers_contest¥jsp-1.4.4-full¥tools¥windows

②DLL ファイルを読み込む

上記¥windows ディレクトリ内の Motorsci.sce ファイルに DLL を読み込む命令を実装しこれを実行。

③Scicos を立上げ、上記¥windows ディレクトリ内の motor.cos ファイルを開く

④VisualC++側の Toppers 環境を実行する。

⑤Scicos の Simulate→Run を実行

＊必ず④→⑤の順序で実行してください Scicos 側から Toppers のコンソールハンドルを探しに行くためです。

⑥Toppers のコンソールを選択して、テストしたい要求キーを実行。

（デフォルトの方式と同様です）

２．アプリケーションの説明

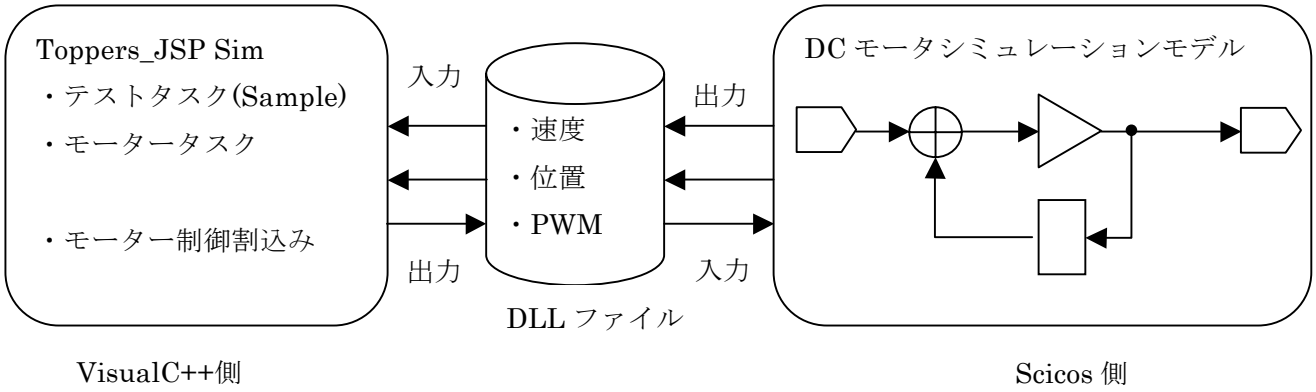


図５： DLL 経由による制御データの共有

DLL ファイルには、モータの位置、速度といった両アプリケーションで共有したい情報を実装します。また、Scicos 側からのサンプリング時間によるタイマ割込みを実行するための機能(Win32 関数)を実装しています。

タイマ割を Toppers 側でなく Scicos 側から実行する理由は、以下の理由によります。

①（Scicos 計算上の）サンプリング間隔に合わせて割込み計算を実行したいため。

②製品、評価の要求によっては us オーダーでタイマ割込みを行う場合が想定されるため。

Windows_API 上でのタイマは最少 1ms サンプリングですが、Scicos 上では動作は遅くなってしまいますが、シミュレーション計算上 1ms 以下でのサンプリングが可能なため、割込みを Scicos 側に持たせることで、疑似的に us オーダーの割込みタイミングを作ることを想定しています。（今回の評価モデルは 1ms サンプリングです）

~~・Toppers 側から(VisualC++上での)ソフトブレークをかけた際に Scicos 側もブレークさせたいため（~~SendMessage 機能で Scicos 側から割込みを飛ばすことで、Toppers 側のブレークに連動して Scicos 側をハングさせる方式を試しましたがうまく行きませんでした）

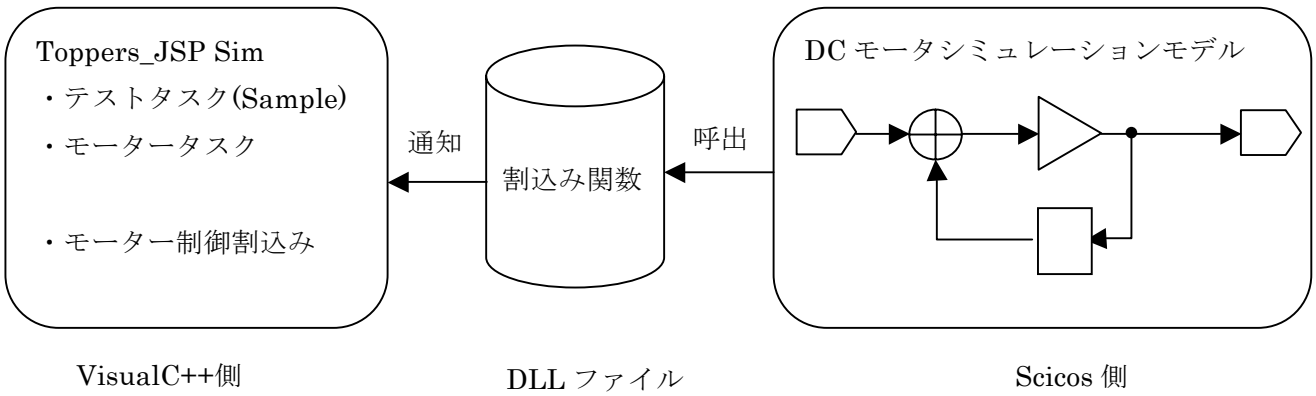


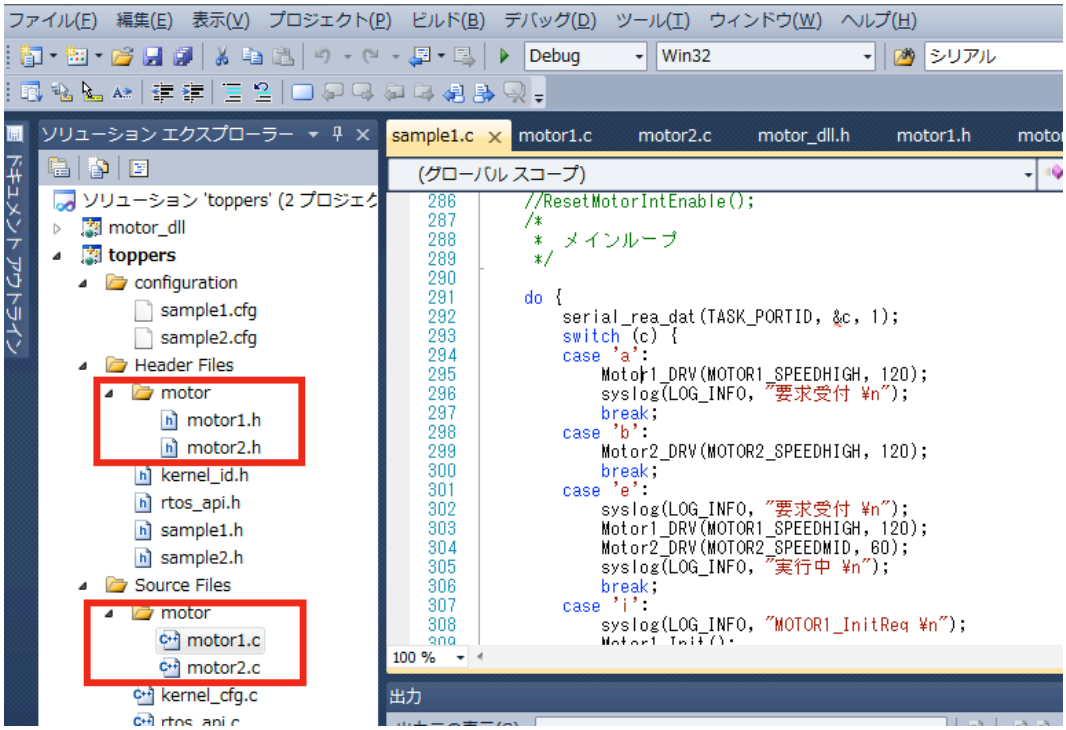
図６：Scicos 側から DLL 経由によるタイマ割込み通知

２．１　Toppers 側実装内容

・モータ制御用に”motor1.c/h” , motor2.c/h　ファイルを追加。

・上記 2 ファイルに、モータの制御タスクと割込み関数を追加。

・コンフィギュレーションファイルに motor1、2 の OS リソースを追加



２．２モーター制御タスク(motor1 / 2)の構成

・他タスクへの公開関数
　要求速度、駆動長さの入力
・タスク機能トップ関数
　OS によるメッセージ受付
・タスク機能内部関数
　駆動前設定
・割込み PID 制御関数
　位置、速度(HAL 関数化)を
　検出して PWM を計算出力
・HAL 層
　共有データを HAL 関数化

motor1.c ファイル(2 側も同様)

・モーターの速度定義
・PID 制御パラメータ定義
・モータータスク要求の定義
・タスクへのメッセージ内容の定義

motor1.h ファイル(2 側も同様)

図 7 ：Toppers 側モーター制御用追加内容

２．３モータ駆動制御の内容

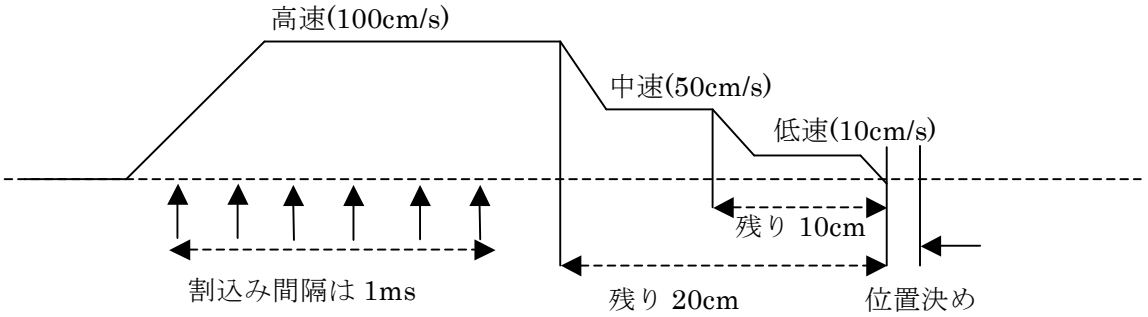


図８：割込みテスト用モーター駆動内容

シミュレーション動作の確認用に、図 8 の内容のモーター制御を実装しています。

・モーター動作機能

速度：高速、中速、低速の 3 速度を持ち、目標位置までの残り距離に応じて、段階的に減速を行う。

位置決め：目標位置到達後、10ms 間位置決め制御。(1ms サンプリングで 10 回)

・モーター数

2 モーター（制御方式はいずれも同じ）

2 モータにそれぞれにタスクを割り当てて、個別動作、2 モーター順番動作、並列動作を行います。

２．４評価プログラム内容（sample1.c）

デフォルトの main_task 内の case 文を変更して以下のモーター駆動機能を追加しています。

・初期化（両モーター）

・モーター１単体駆動

・モーター２単体駆動

・モーター１、２、並列駆動

＊連続で駆動要求を入れた場合は前回駆動を待ちます。

各タスクで自身のモーターの駆動完了を待ってから、メッセージプールを解放するため、その間次の駆動要求はメッセージプール取得待ちとなるためです。

３．DLL 実装内容

３．１共有データ内容

モーター1、2 向けにそれぞれ以下の制御情報を共有しています。

・モータ位置(cm)

・モーター速度(cm/s)

・PWM 周期(1000cnt)

・PWM 指令値(0~1000cnt)

・割込み許可信号（フラグ）

＊データ共有は VisualC++に用意された専用命令を用いています。

```
#pragma data_seg("ラベル名")　～　#pragma data_seg() /* シミュレーション用 */
```

３．２割込み要求の制御

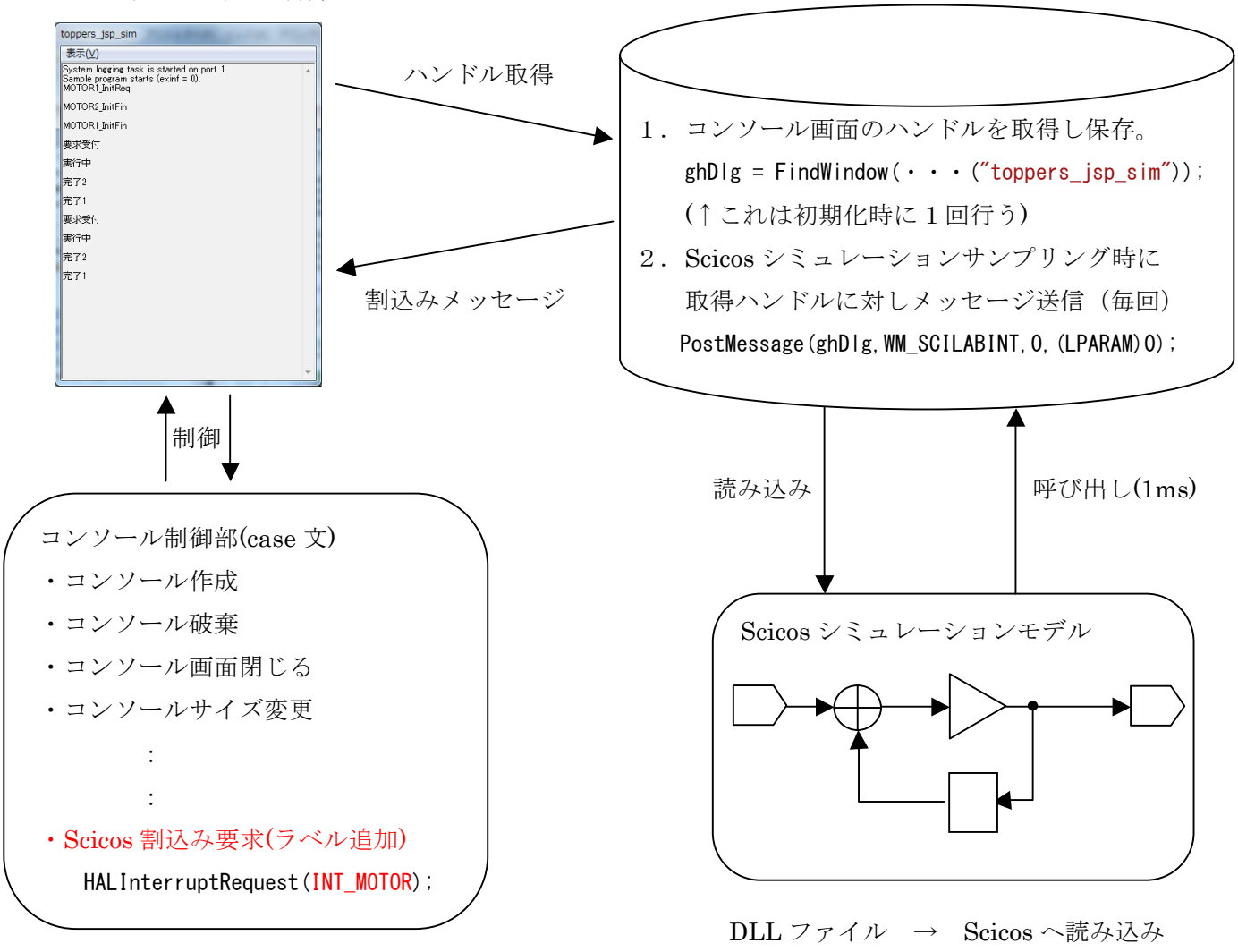


図９：DLL による割込み機能の説明

４．変更・追加ファイル

<変更ファイル>

Sample1.cfg：モーター用タスク等 OS リソース追加

Sample1.c：評価用プログラム追加

Sample1.h：評価用プログラム追加

Hw_serial.c：コンソール制御部への Scicos 割込み機能追加

Constants.h：Scicos 割込みラベル定義追加

Vwindows.h：VisualC++2010Express 環境でのシミュレーション立上げ用変更

Hal_resource.rc：VisualC++2010Express 環境でのシミュレーション立上げ用変更

<評価用追加ファイル>

Motor1.c：モーター1 制御タスク、割込み、HAL 層

Motor1.h：モーター1 制御用パラメータ、変数定義

Motor2.c：モーター2 制御タスク、割込み、HAL 層

Motor2.h：モーター2 制御用パラメータ、変数定義

Rtos_api.c：OS サービスコールのラッパ関数定義(未使用)

Rtos_api.h：OS サービスコールのラッパ関数定義

<DLL ファイル関連>

Motor_dll.cpp：共有変数の定義、関数の実体

Motor_dll.h：共有変数、関数の定義

Morot_dll.def：共有関数の定義

＊使用したファイル類は別途添付致します。

<使用ソフト>

・Microsoft_VisualC++2010Express

http://www.microsoft.com/visualstudio/jpn/downloads#d-2010-express

・Scicos_lab4.41

http://www.scicoslab.org/

５．制限事項（動作不良含む）

①初回駆動は出来るが、何度か駆動していると、動作完了が返ってこなくなり、以降の駆動要求を受け付けられなくなることがある。

②シミュレーション確認がメインということもあり、制御の精度はそれなりです。