RTC-scilab
クイックスタートガイド

第 1 版　2010 年 12 月 7 日

内容

1. 概要

1.1. RTC-scilab の概要

RTC-scilab は，scilab で開発したコントローラと，RT ミドルウエアによって通信が標準化されたロボットシステムとの接続を容易にするツールボックスです．

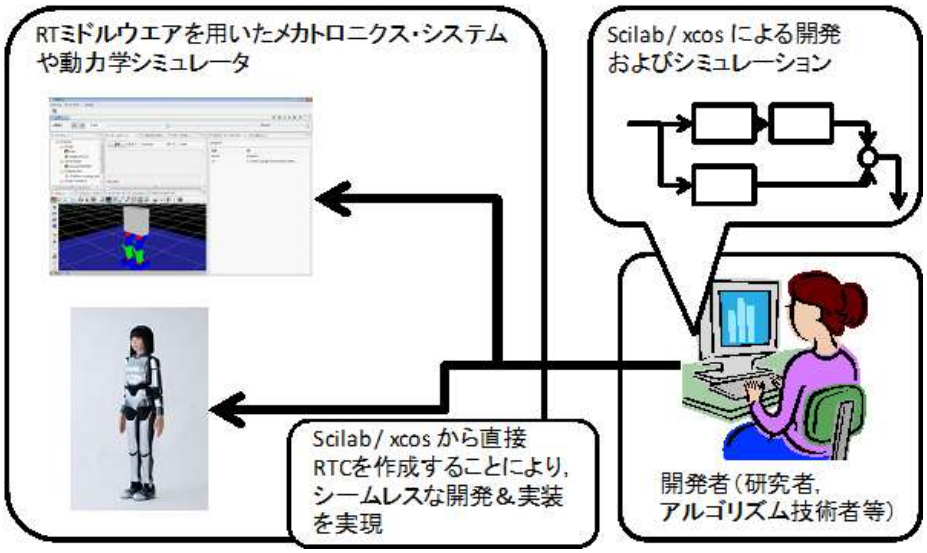


図 1 RTC-scilabの概要

RTC-scilab を用いた場合，開発者はロボット毎に開発が必要な独自のアルゴリズム，たとえば歩行安定制御や機械学習などの最適化アルゴリズムなどを scilab を用いて開発し，scilab 上でシミュレーションしたのちに，すぐさま RT ミドルウエアを用いたシミュレータである OpenHRP3 や，RT ミドルウエアを用いたロボット群およびカメラや音声認識モジュールなどのコンポーネントに適用することができます．この RTC-scilab の効果は研究開発を効率化するのみではありません．これまで機械工学や電子工学の知識の薄かった計算知能などの技術者においては，自身の実装したアルゴリズムをロボットという実体をもったプラットフォームで検証することができます．また，マイコンなどの組み込み技術に疎かった機械技術者においては，自身のコンセプトを実現したロボットを動かす制御プログラムを，scilab/xcos を用いてグラフィカルに実装することができるため，RT ミドルウエアを用いる場合に本来必要であった，CORBA などの分散オブジェクト指向プログラミング技術への理解を必要としません．

これにより，ロボット技術を用いる機会を，より多くの研究者，技術者，企業に提供することが出来ると期待されます．

1.2.　RTC-scilab の準備

1.2.1.　ダウンロード＆展開

ダウンロードのページより最新バージョンのパッケージをダウンロードします．ZIP 圧縮されていますので，解凍してマイドキュメントなどの好きなフォルダに置きます．

1.2.2.　ビルド

展開したフォルダにある，「builder.sce」をダブルクリックして，scilab を起動します．

図 2 RTCscilabを展開したフォルダ

scilab をインストールしたときに，正しくファイルの拡張子と scilab が結び付けられていれば，scilab が自動的に起動し，エディタが開きます．

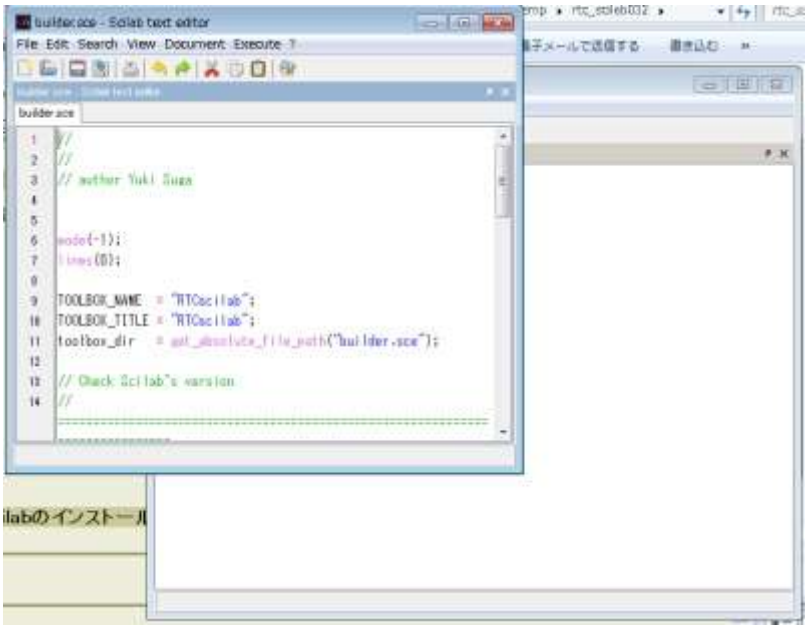


図 3 scilabおよび，scilabのテキストエディタが起動します．

ここで，エディタをアクティブにした状態で，「Ctrl + E」を押し，「builder.sce」のスクリプトを実行します．すると，自動的にビルドされます．もしくは，カレントディレクトリを rtc_scilabXXX のディレクトリに変更したのちに，″exec builder.sce″コマンドでもビルド出来ます．

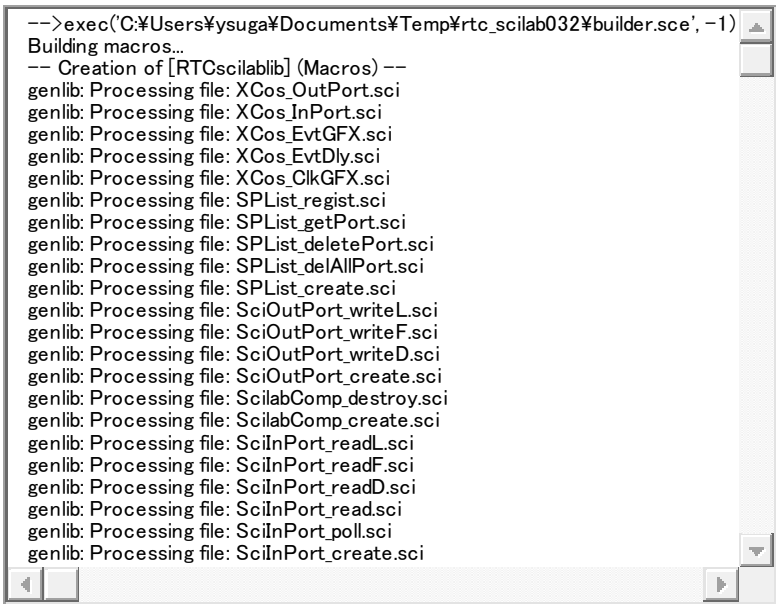


図 4 出力メッセージの例

1.1.1. RTCscilab のインストール

ビルドした RTCscilab のフォルダを，scilab をインストールしたフォルダ内の「contrib」というフォルダに入れます．
普通に scilab5.2.2 をインストールした場合，contrib フォルダの場所は，
C:¥Program Files¥scilab5.2.2¥contrib （32 ビット）
C:¥Program Files (x86)¥scilab5.2.2¥contrib (64 ビット)
になります．
(Linux の場合は筆者の環境 Ubuntu 9.10 では /usr/local/scilab5.2.2/share/scilab/contrib でした．)
アクセス制限のダイアログが出るかもしれませんので，「許可」を選択してください．

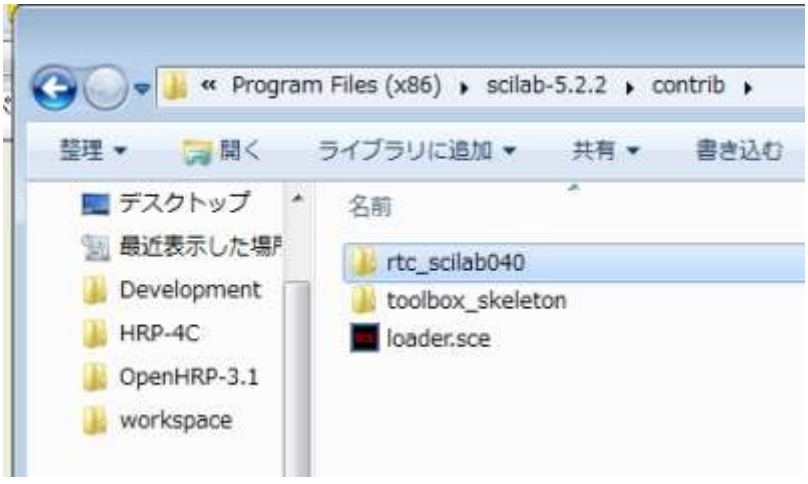


図 5 Windows7(x64)マシンのcontribフォルダ

1.2.3. xcos で使うパレットの読み込み

xcos で使うパレットを読み込んでおきます．コンソールで，
-->xcos
と打ち込み，xcos のパレットを起動します．

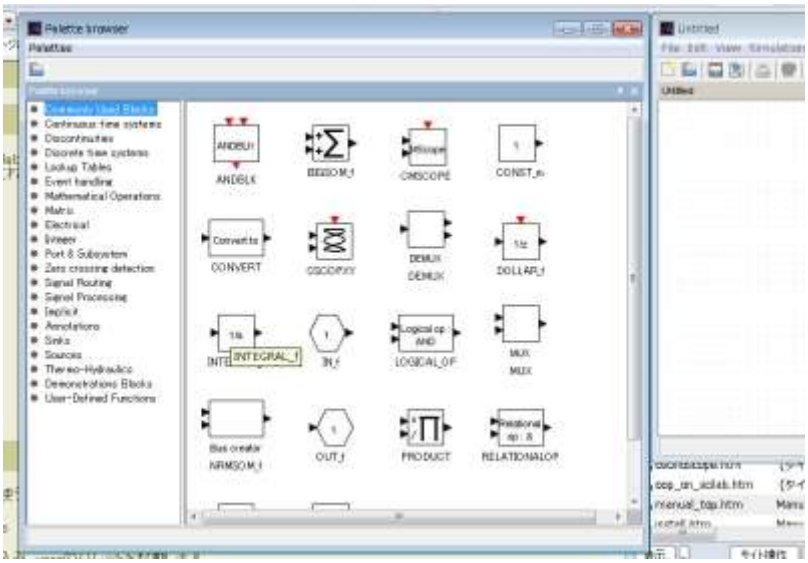


図 6 xcosパレット

メニューの「Palettes > Open」と選択し，RTCscilab をインストールしたフォルダにある「RTC_Blocks.xcos」というファイルを選択します．

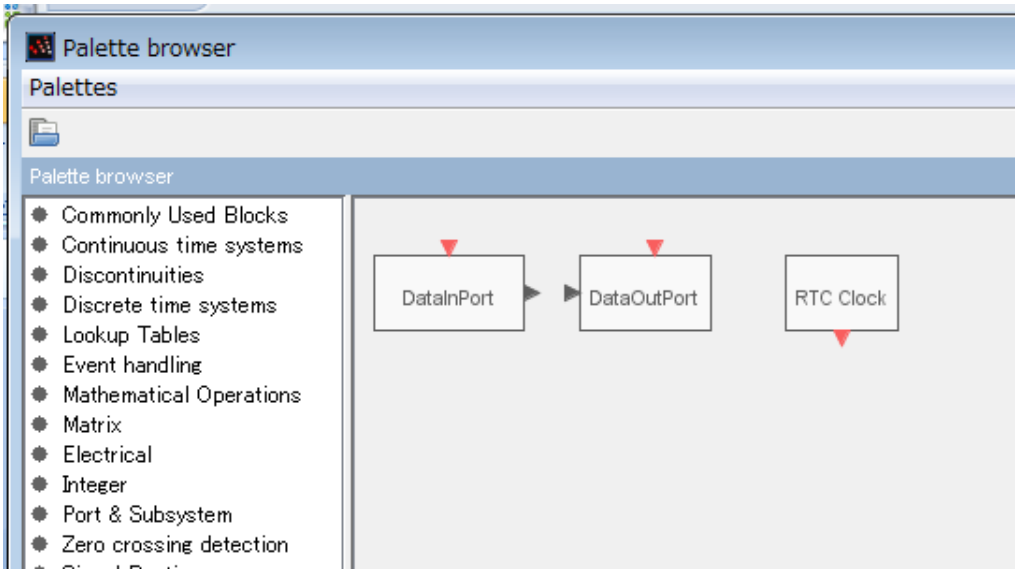


図 7 DataPortパレット

これでデータポートが使えるようになりました．

2. RTCscilab を使う前の準備

2.1. ネームサービスの起動

OpenRTM-aist では，起動しているコンポーネントを管理するためのネームサーバを起動しなくてはなりません．通常のインストール手順であれば，「スタートメニュー」＞「OpenRTM-aist」＞「C++」＞「tools」＞「Start Naming Service」で起動します．ネームサーバーはたびたび起動しますので，デスクトップにアイコンを置くなどすると良いでしょう．

2.2. カレントディレクトリの移動

Windows 版などでスタートメニューから scilab を起動すると，デフォルトのカレントディレクトリ（作業ディレクトリ，作業フォルダ）が scilab がインストールされているフォルダになってしまいます．そこで，カレントディレクトリの移動が必要になります．ファイルメニューからディレクトリの移動を選択し，任意のディレクトリに移動します．

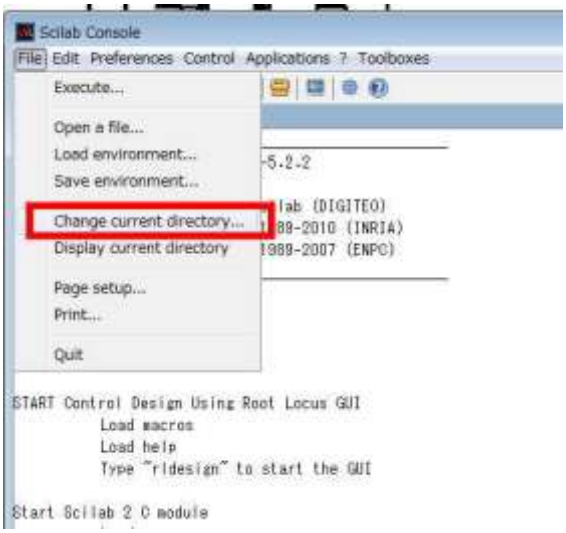


図 8 カレントディレクトリの移動メニュー

2.3. RTCscilab のツールボックスの読み込み

RTCscilab のインストールが成功していると，下図のように Toolboxes のメニューに rtc_scilabXXX が表示されます．これを選択して，まずは RTCscilab を読み込みます．

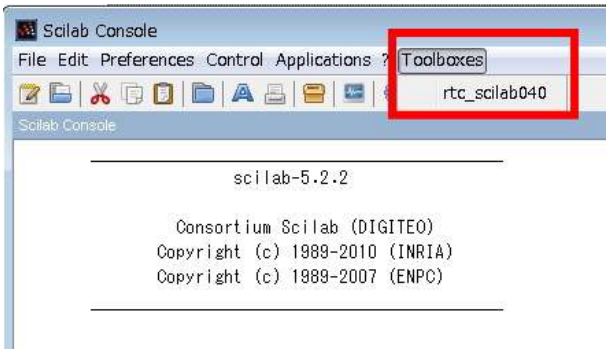


図 9 RTC-scilabのロード確認

2.4. rtc.conf の設定

OpenRTM-aist の利用に慣れた人ならば，rtc.conf の設定が必要なことが想像できると思います．RTC-scilab には，rtc.conf の設定を助ける機能が備わっています．カレントディレクトリに rtc.conf が存在しない場合は，rtcconf_generate コマンドで，デフォルトの rtc.conf を作成できます．

```
--> rtcconf_generate
```

また，rtcconf_modify で rtc.conf の設定を変更できます．

```
--> rtcconf_modify("exec_cxt.periodic.rate", "200")
```

さらに，rtcconf_showall で現在値を確認出来ます．

```
--> rtcconf_showall
```

ひとまず，デフォルトの rtc.conf をカレントディレクトリに用意してから実行してください．

3. 動作確認

RTC-scilab の動作確認をしましょう．コマンドラインでインタラクティブに RTC を生成・制御できます．

3.1. RTC の簡易生成

RTC をいきなり簡単に生成出来ます．

3.1.1. NamingService の起動

Corba の NamingService を起動します．RT ミドルウエアに慣れた人ならば簡単でしょう．

3.1.2. scilab の起動

scilab を起動します．上述したようにカレントディレクトリを適当な場所に変更します．Windows ならばマイドキュメントあたりが問題ないでしょう．

3.1.3. RTC-scilab の読み込み

上述した通り，RTC-scilab を読み込んでください．

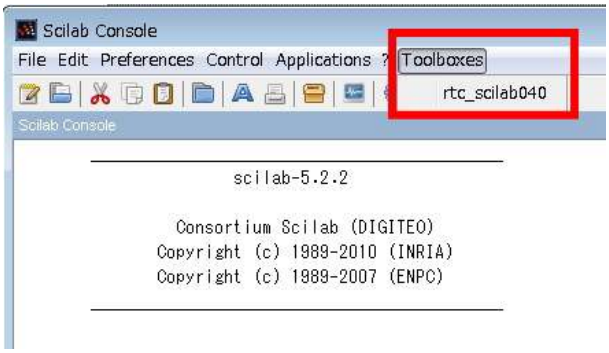


図 10 RTC-scilabのロード

3.1.4. rtc.conf の生成・変更

RTC-scilab での RT コンポーネントの作成には rtc.conf という設定ファイルが必要です．カレントディレクトリに rtc.conf が無い場合は，「rtcconf_generate」コマンドで rtc.conf が生成されます．とりあえずここはデフォルトの rtc.conf で OK です．

--> rtcconf_generate()

3.1.5. initRTM('XXX')

initRTM コマンドを入力します．引数は生成される RTC のデフォルトの名前です．たとえば，initRTM(″hogehoge″)と入力すると，最初に生成される RTC は，「hogehoge0.rtc」になります．この引数は省略できます．省略した場合は labRTC がデフォルトの名前になります．とりあえずここはデフォルトで，

--> initRTM();

と入力してみましょう．

3.1.6. RTC_create

RTC を生成します．

--> rtc = RTC_create()

これで rtc に生成された RTC と結びついた値（これをハンドル値と呼ぶことにします）が代入されます．通常，ハンドルが負の値の場合は生成や制御が失敗になった事を表します．

この時点で RTC が生成されているので，RT System Editor などで確認出来ます．

3.1.7. ポートの追加

RTC に対してポートの追加をするには，「RTC_addInPort」か「RTC_addOutPort」を使います．たとえば，

--> port_in0 = RTC_addInPort(rtc, ″TimedFloat″, ″in0″);

と入力すると，ハンドル値 rtc で指定される RTC に対して，TimedFloat 型の in0 という名前の入力ポートが生成されます．このポート自体のハンドル値も保存しておきます．この RTC-scilab で対応している型は，

- TimedLong
- TimedFloat
- TimedDouble
- TimedFloatSeq
- TimedLongSeq
- TimedDoubleSeq

の 6 つです.

ここでは今作成した in0 という入力ポートに加えて, out0 という出力ポートを生成しておきましょう.

--> port_out0 = RTC_addOutPort(rtc, "TimedFloat", "out0");

3.1.8. RT System Editor で接続&アクティベート

RT System Editor で作成した RTC の入力ポートと出力ポートを接続して, activate しておいてください. 接続した場合に, dataflow type が push になっている事を確認してください.

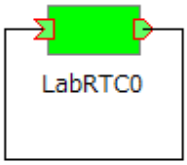


図 11 RTCの状態をこの状態にしておく

3.1.9. ポートへの書き込み

出力ポートにデータを書き込む場合は,「OutPort_write」関数を使います. これは非常にシンプル.

--> OutPort_write(port_out0, 123.456);

3.1.10. ポートの読み込み

入力ポートにデータが来ているか否かは,「InPort_isNew」で確認します.

--> InPort_isNew(port_in0)

このときに, 返り値が%T(真)ならばデータがバッファにあります. %F(偽)ならばナシです.

(最後にセミコロンを付けると, コマンドの返り値がプロンプトに表示されませんので注意. ret などの変数で受けてから, disp(ret)で表示するの OK)

データが来ていたら,「InPort_read」でデータを読み出せます.

--> InPort_read(port_in0)

返り値が読み込んだデータです. 上手くいけば 123.456 というデータが返ってきます.

3.1.11. まとめ

RTC-scilab を使えば，簡単にデータポートを持つRTC を作成できます．今は scilab のプロンプトからインタラクティブに RTC を作成しましたが，sce という拡張子のスクリプトファイルを実行することも出来ます．また，help コマンドを使って，RTC-scilab の機能の解説を見ることが出来ます（英語のみですが・・・）

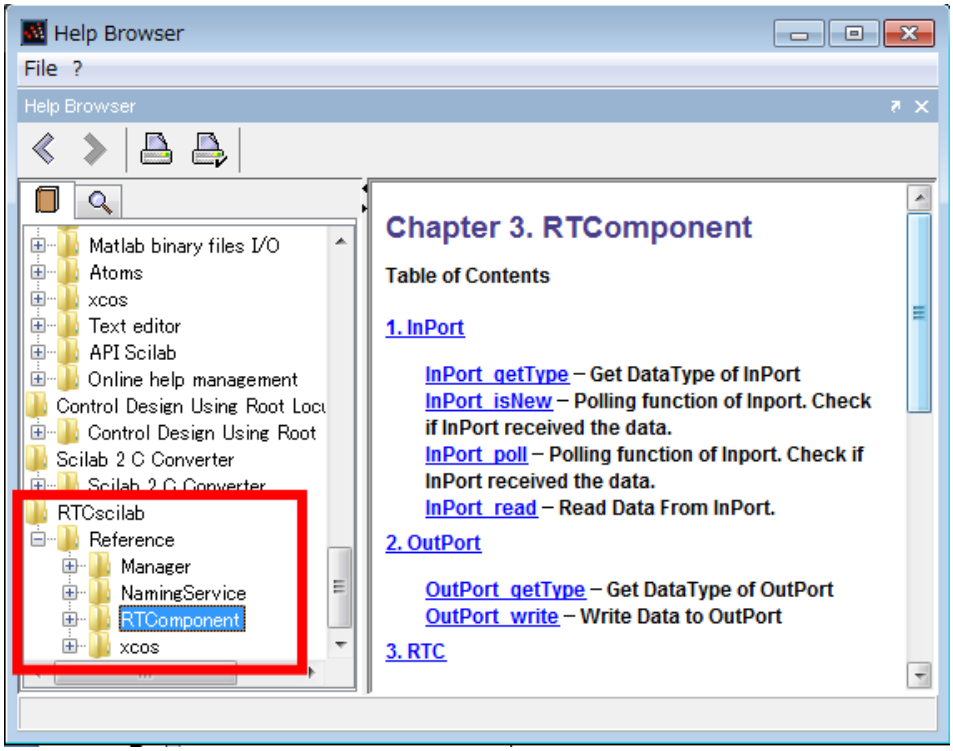


図 12 RTC-scilabのヘルプトピック

4. デモの起動

RTCscilab のツールボックスにはデモが同梱されています．下図のアイコンをクリックして，デモのメニューを起動します．

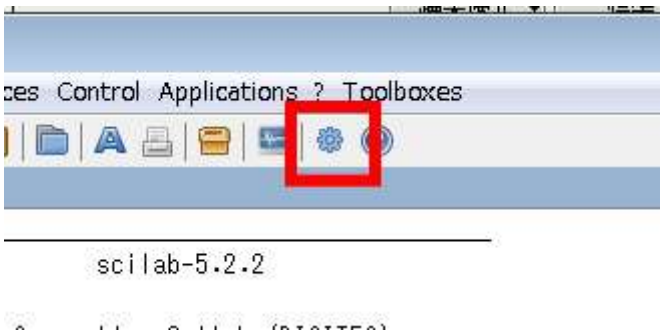


図 13 デモの起動アイコン

デモのメニューには「RTCscilab Toolbox」が表示されていると思うので，クリックしてください．

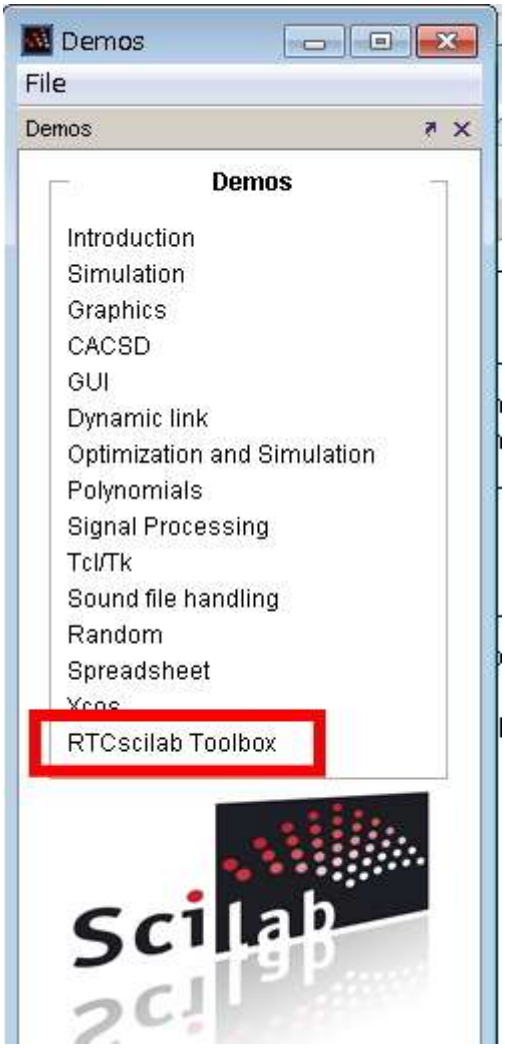


図 14 デモの選択画面

下図の右側のメニューからデモを選びます.

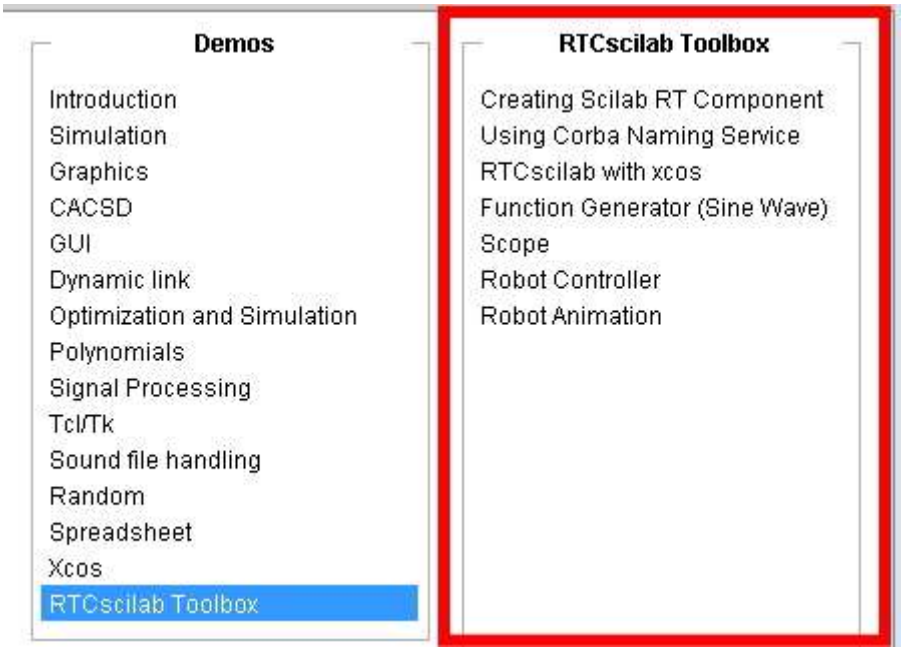


図 15 デモの選択画面その2

デモのソースコードは，rtc_scilabXXX フォルダ内の demos フォルダに格納されている**.sce ファイルです．デ

モのタイトルと，sce ファイルとの関連づけは，RTCscilab.dem.gateway.sce というファイルに記述されています．参考になると思います．

それでは次の章からはデモについて解説しましょう．

4.1. 簡易オシロスコープ

5 分掛からずに RTC を作成できます．

4.1.1. オシロスコープとデータポートをつなげる

まずはもちろん，RT ミドルウエアのネームサーバを起動します．

4.1.1.1. Demo から Scope を起動

前の章で解説したとおりデモを起動して，Scope を選びます．

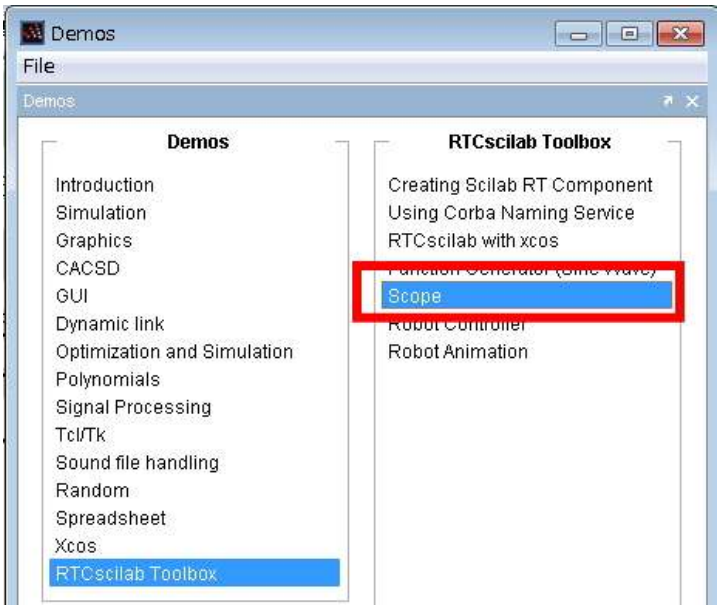


図 16 Scopeデモの起動

すると，下図のようなブロック図が出てきます．この中の「DataInPort」が RTC の InPort です．

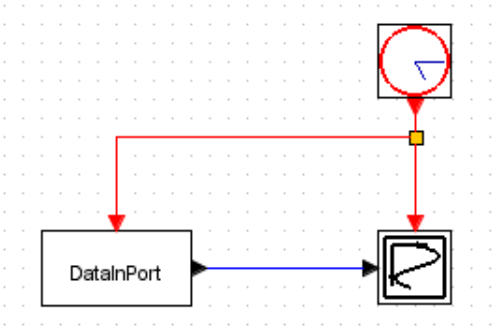


図 17 Scopeデモのブロック線図

では，再生ボタンをクリックして，デモのシミュレーションを起動します．すると，下図のようなオシロスコープのような画面が出ます．

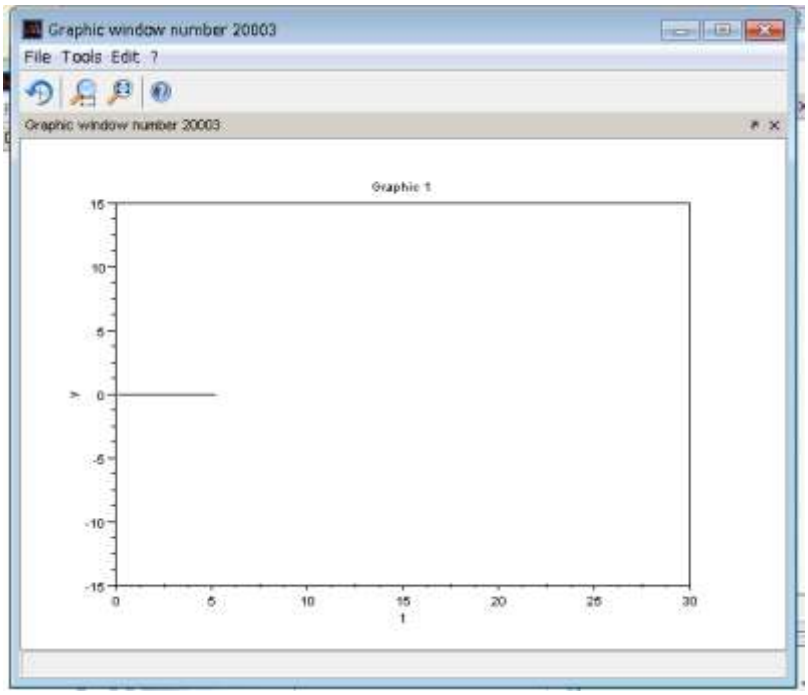


図 18 Scopeの画面

トリガのような機能はありませんが，データを視覚化することが出来ます．この状態で，RT System Editor を観察すると，以下のような RTC が出来ています．

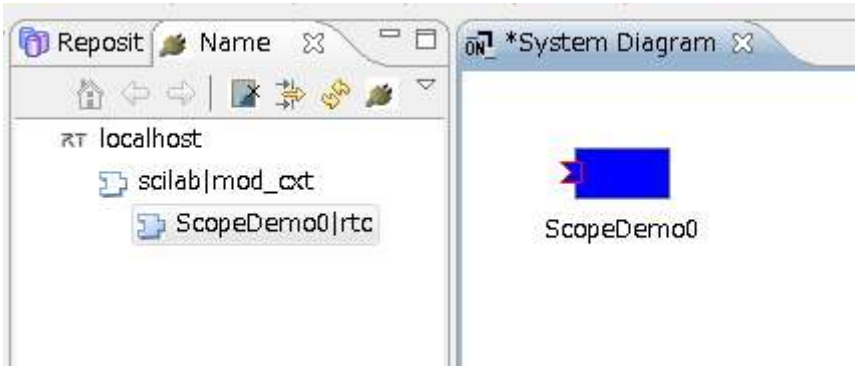


図 19 RT System Editorから見たScopeデモ

デフォルトでは TimedFloat 型のポートですので，ここにデータタイプやサイズなどの設定を変更してあげてくだ

さい(後述). アクティブにすると Scope の画面の情報が更新されるという仕組みです.

4.1.2. 自分で Scope の RTC を作る

DataPort の設定方法などを解説します.

4.1.2.1. scilab の起動・rtc.conf の設定・RTCscilab のロード

2 章の順序で RTC-scilab を使う準備をします.

4.1.2.2. パレットから scope を図に追加

パレットから scope を追加します.

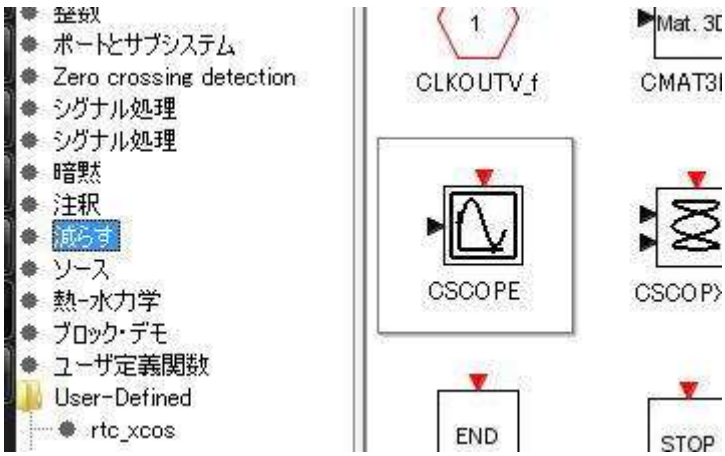


図 20 SinkパレットからCScopeを追加

4.1.2.3. Source パレットからクロックを追加

Source(ソース)パレットから Clock_c を追加します.

4.1.2.4. パレットからデータポートを追加

さらに DataPorts パレットから InPort を追加します.

4.1.2.5. オシロスコープを設定

オシロスコープの縦軸, 横軸の大きさを決めます. ここでは[-1000, 1000]にしてみます. scope ブロックをダブルクリックして設定ダイアログを開き設定をします. ポイントは以下の 3 点

Ymin･･･縦軸最小値 （ここでは- 1000 に）
Ymax･･･縦軸最大値 （ここでは 1000 に）
Refresh period･･･横軸時間幅(30 秒でいいです)

図 21 Scopeの設定画面

4.1.2.6. データポートを設定

次に InPort ブロックの設定を行います．ポイントは

- データサイズ(Output port sizes)
- データポート名(Real parameter vector)
- データ型(Object parameters vector)

ここは試しに TimedLongSeq 型のデータポートでデータを 3 つ，取得できるオシロスコープを作ります．

- Output port sizes => [3, 1] （3×1の行列）
- Real parameter vector => [0] (ポート名が"in0"になる)
- Object parameters vector => list("TimedLongSeq") (TimedLongSeq 型)

4.1.2.7. 最後に接続して起動

さらにイベントポートをソースのパレットから追加して接続すると図のようになります．

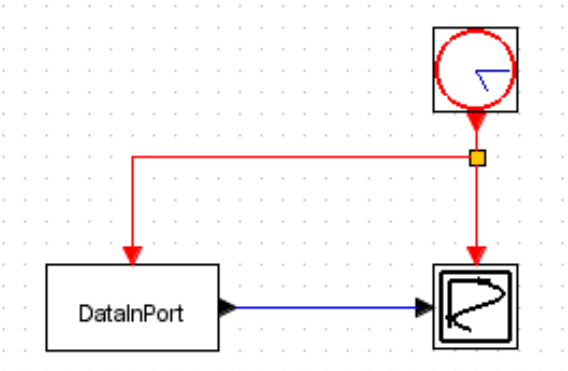


図 22 Scopeの接続

4.1.3. シミュレーションの設定

メニューの「シミュレーション」＞「セットアップ」と選択します．

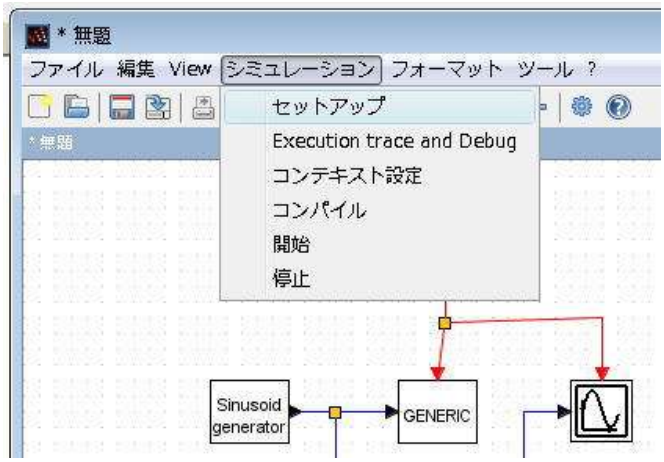


図 23 シミュレーションの設定方法

ここで出てきたダイアログの「実時間スケール」を「1.0」にすると，実時間とシミュレーション時間を同スケールにすることが出来ます．同じってことです（さすがに誤差は出ます）．

図 24 シミュレーションの設定

4.1.4. シミュレーションの開始

これでシミュレーションを起動すると RT コンポーネントが起動されます．アクティブにすれば同時に起動したオシロスコープの画面にデータが反映されます．

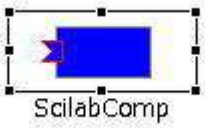


図 25 Scopeデモ

4.2. 簡易ファンクションジェネレータ

4.2.1. デモを見る

デモのメニューを起動し，下図のように Function Generator (Sine Wave)を起動します．

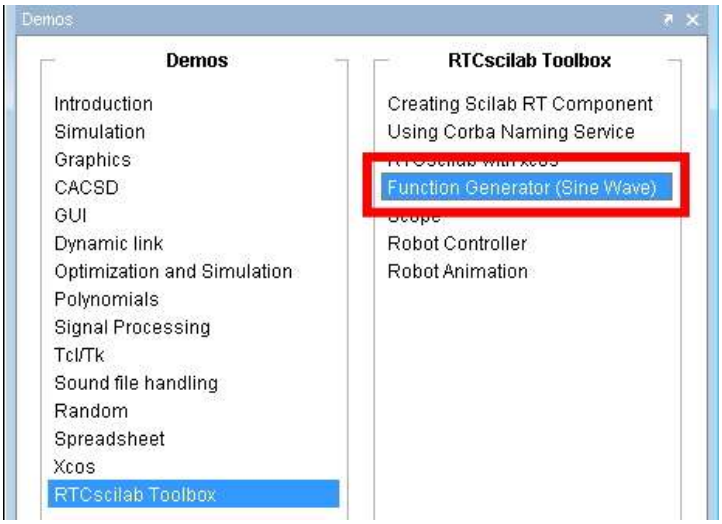


図 26 ファンクションジェネレータのデモ

すると，下図のようなブロック線図が表示されます．

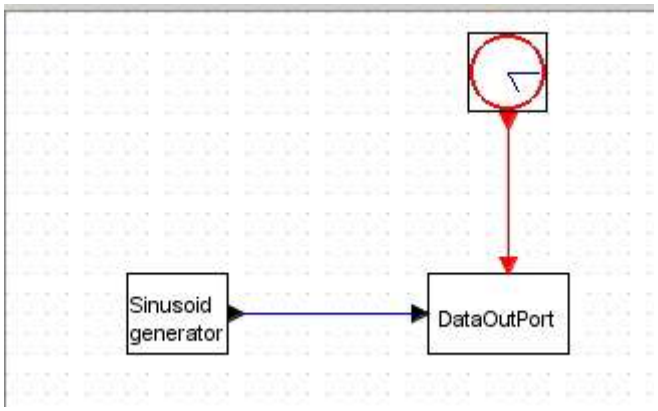


図 27 SIN波生成器

再生ボタンをクリックしてシミュレーションを開始すると，下図のような RTC が現れます．

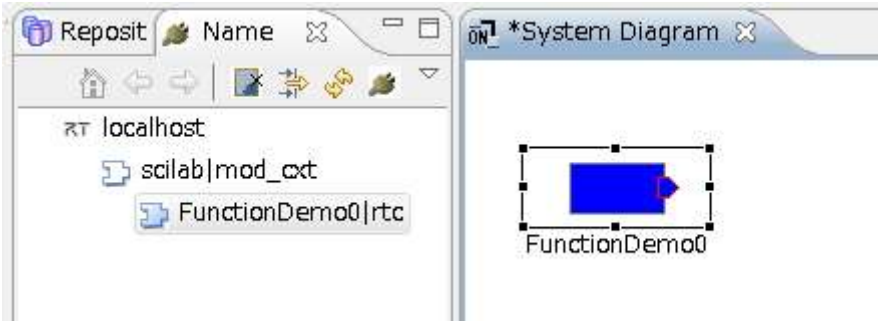


図 28 ファンクションジェネレータRTC

4.2.2.　自分で作ろう，SIN 波を出す RTC

4.2.2.1.　RTC-scilab の準備

まずは，2 章の要領で RTC-scilab の準備をします．

4.2.2.2.　DataPort の配置

まずは DataPort の Out 側を配置します．「ソース」のパレットから，CLOCK_c と SINUSOID GENERATOR を配置します．ここで接続すると下図のようになります．

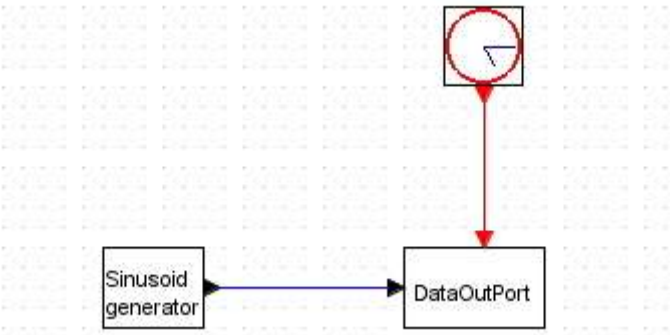


図 29 SIN波生成器

さらに出力する波形を確認するためのオシロスコープ（SCOPE．シンク（SINK）のパレットの中にあります）を配置してつなげます．SCOPE にもイベントが必要なので，SCOPE 側のイベントポート（赤いポート）から，GENERIC につながっているイベントのコネクションにドラッグすると接続が分岐します．

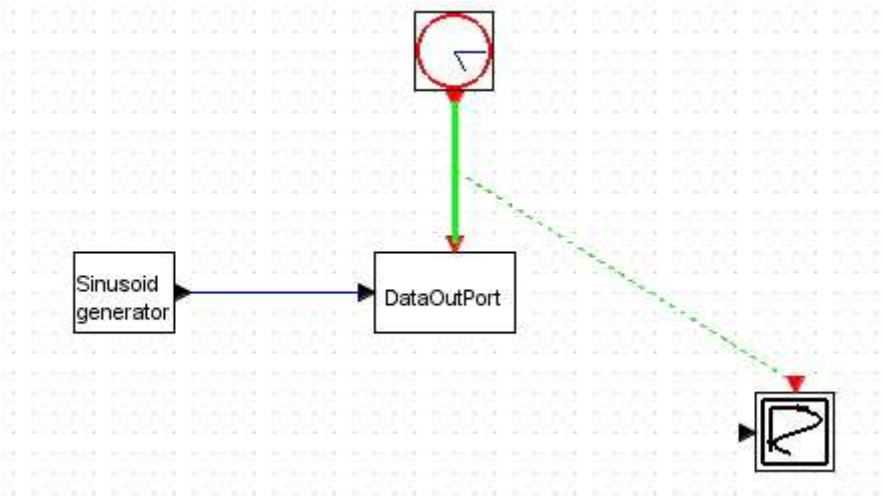


図 30　Scopeを追加してみます

同様に出力も分岐させます．

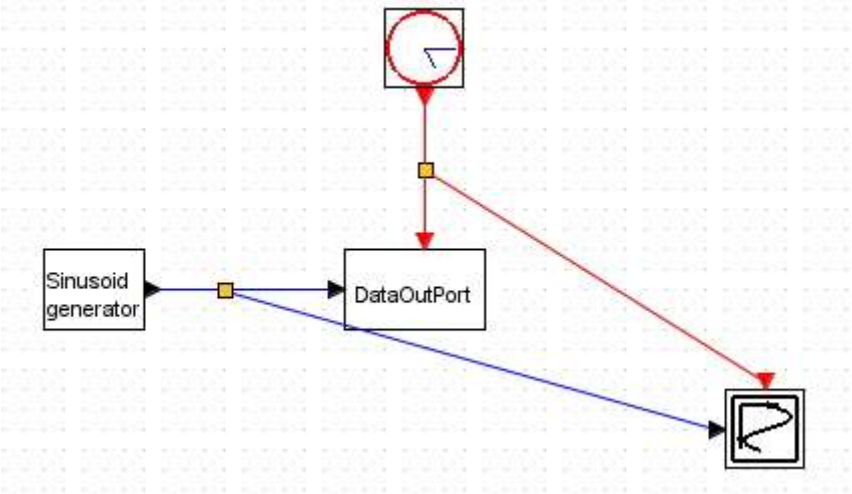


図 31 Scopeの準備

最後に，接続を表している線上でダブルクリックすると，ピボットと作成できますので，これをドラッグして配置を整えます．

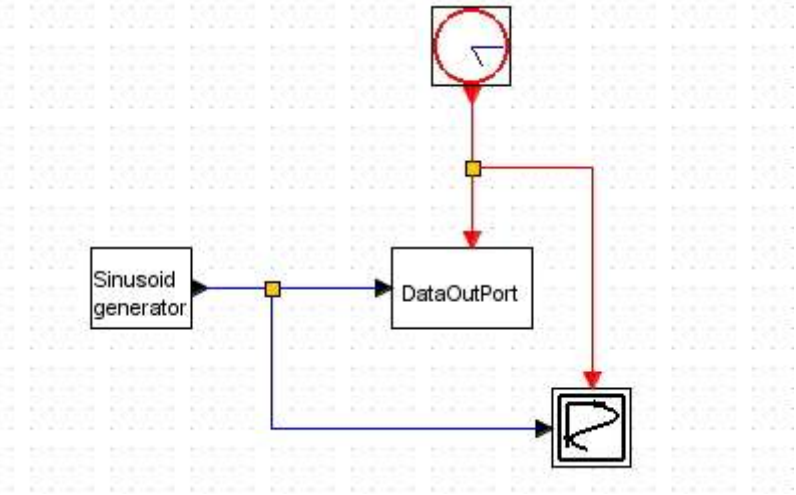


図 32 ピボットを線上ダブルクリックで追加して折れ目を付けると見やすい

4.2.3. シミュレーションの設定

メニューの「シミュレーション」＞「セットアップ」と選択します．

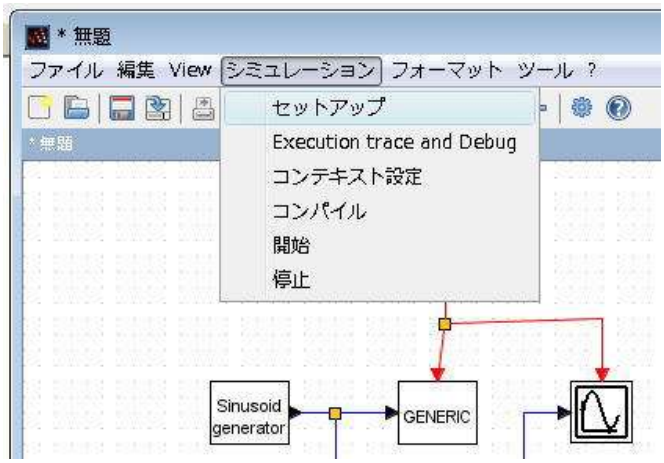


図 33 シミュレーションの設定方法

ここで出てきたダイアログの「実時間スケール」を「1.0」にすると，実時間とシミュレーション時間を同スケールにすることが出来ます．同じってことです（さすがに誤差は出ます）．

図 34 シミュレーションの設定

4.2.4. シミュレーションの開始

シミュレーションを開始すれば，SIN 波がでます．自動的に作成された RTC に OutPort があるのがわかると思います．

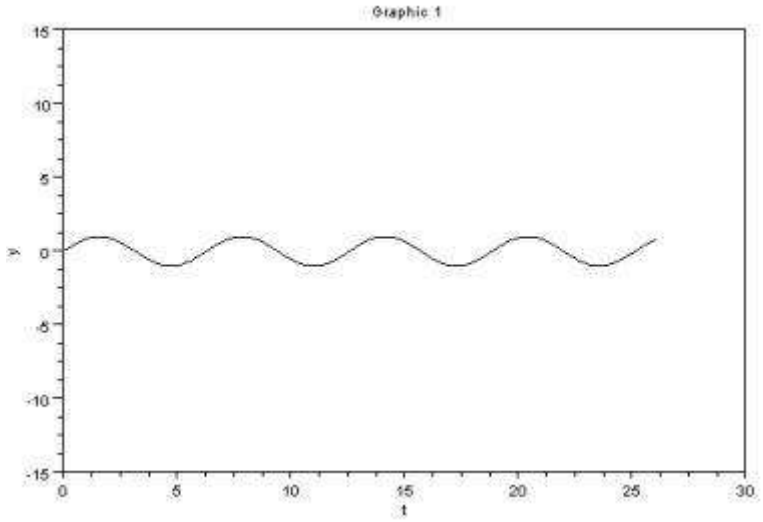


図 35 SIN波が出てる

4.2.5. その他にも…いろいろ

たとえば「Sawtooth generator」でのこぎり波を出したり,

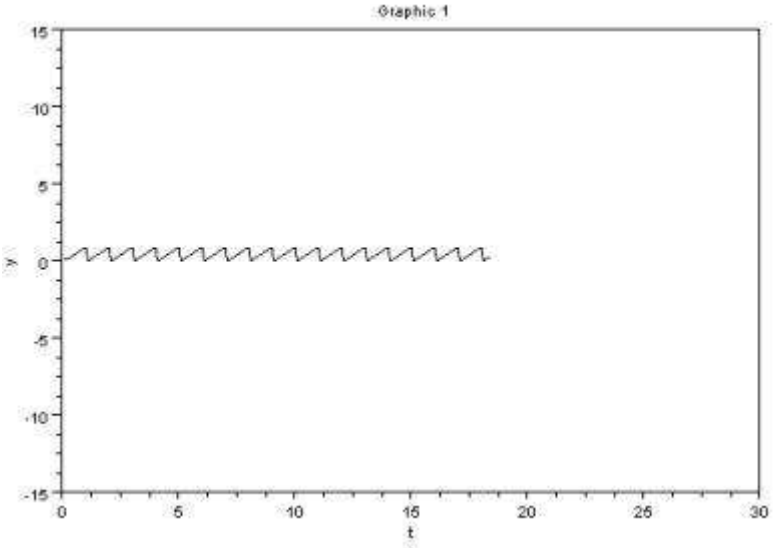


図 36 のこぎり波

また下図のように周波数や位相の異なる SIN 波を加えることによって…

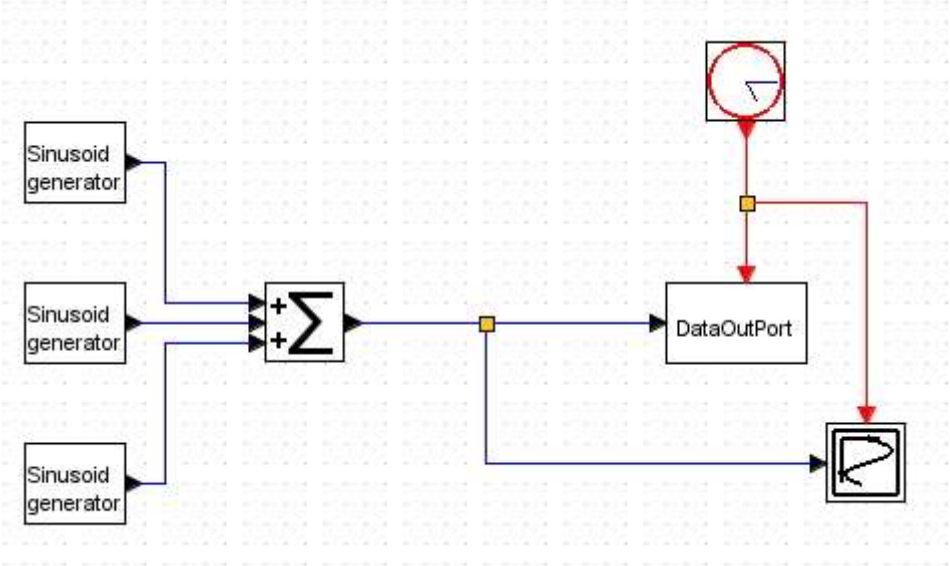


図 37 SIN波の合算

下のような波形を作ることが出来ます.

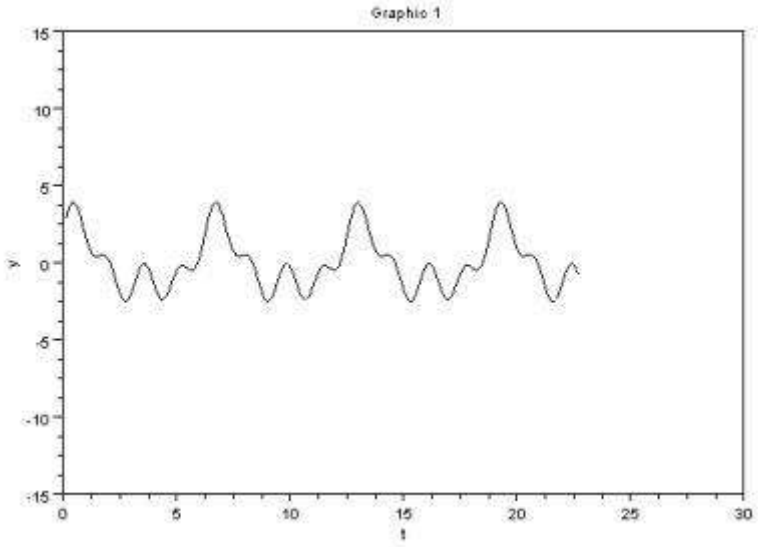


図 38 複雑な波形

また, ファイルに保存した波形を出力することもできます. RANDOM ノイズや, .au 拡張子の音声ファイルを再生したりできます. scilab の細かい内容については, scilab の解説書やウェブサイトを参考にするとよいでしょう.

5. OpenHRP3 との連携

動力学シミュレータの OpenHRP3 と連携して，ロボットの動作をシミュレートすることが出来ます．

5.1. OpenHRP3 のインストール

OpenHRP3 は複数のモジュールからなる動力学シミュレータです．下記 URL からダウンロードします．
http://www.openrtp.jp/openhrp3/jp/

下図のリンクからダウンロードへ移動します．

図 39

ダウンロードサイトの一番下まで移動します．

図 40

下図のリンクから，OpenHRP3 のインストーラをダウンロードしてインストールします．原稿執筆時点では，OpenHRP3.1 Release が最新版です．すると，"C:¥Program Files¥OpenHRPSDK¥GrxUI"のフォルダに，GrxUI.exe がインストールされていますので，この実行ファイルを実行して OpenHRP3 を起動します．GrxUI.exe へのショートカットをデスクトップ等に配置しておくとよいでしょう．

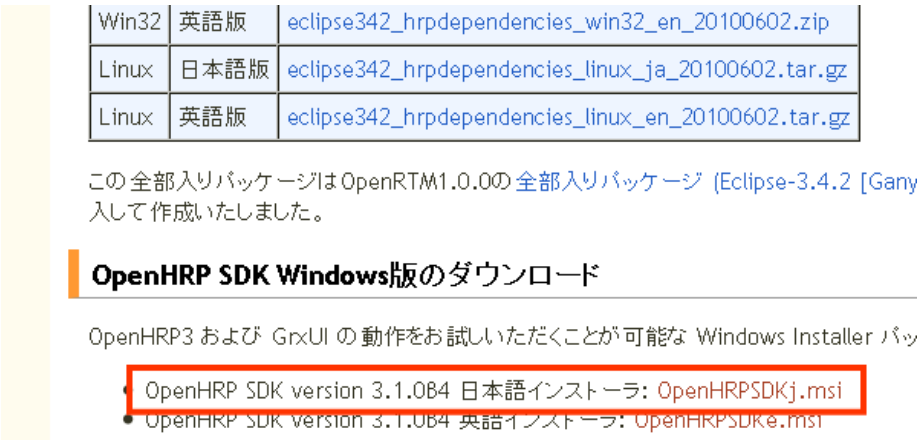


図 41

5.2. デモ

では，デモを実行してみましょう．まず，通常どおりネームサーバーを起動します．

次に，GrxUI.exe を実行して，OpenHRP3 を起動します．

次に，GrxUI のメニューから「GrxUI」＞「プロジェクトの読み込み」を選択します．ここで，rtc_scilab をインストールしたフォルダ内の demos の中の openhrp というフォルダ（通常は，C:¥Program Files¥scilab-5.2.2¥contrib¥rtc_scilab060¥demos¥openhrp¥）の openhrp_demo.xml を開きます．すると，下図のようなモデルが3Dビューに表示されます．

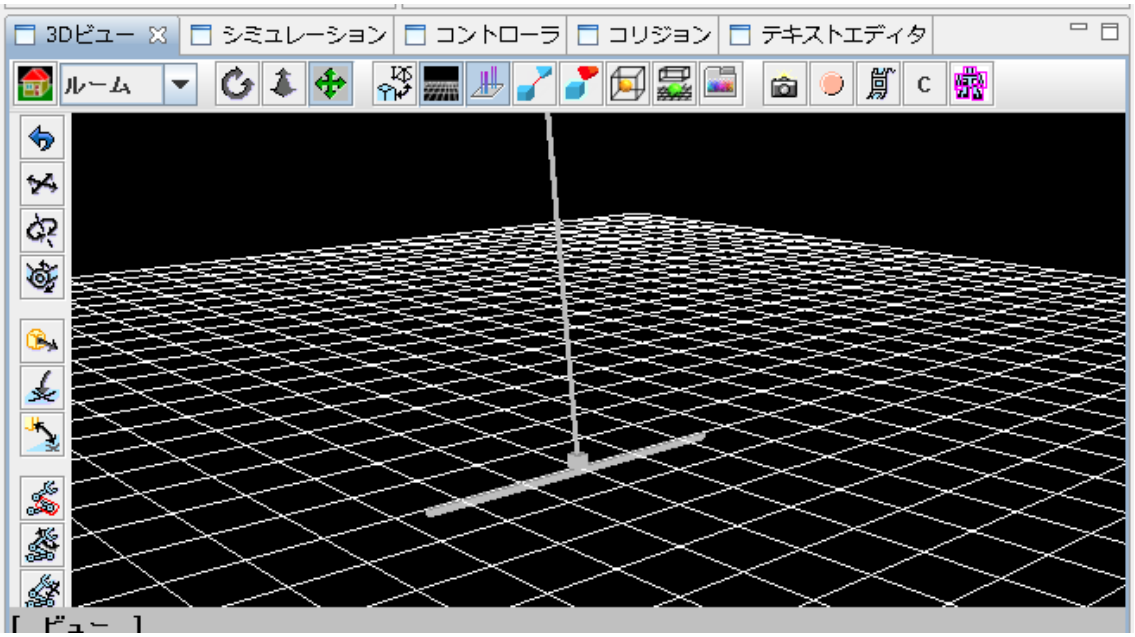


図 42

この状態で，図の実行ボタンをクリックしてシミュレーションを実行してみましょう．

図 43

すると，下図のように倒立振子は倒れてしまいます．これを，振子下部にある台車を異動させることによって立たせるのが，倒立振子の問題です．

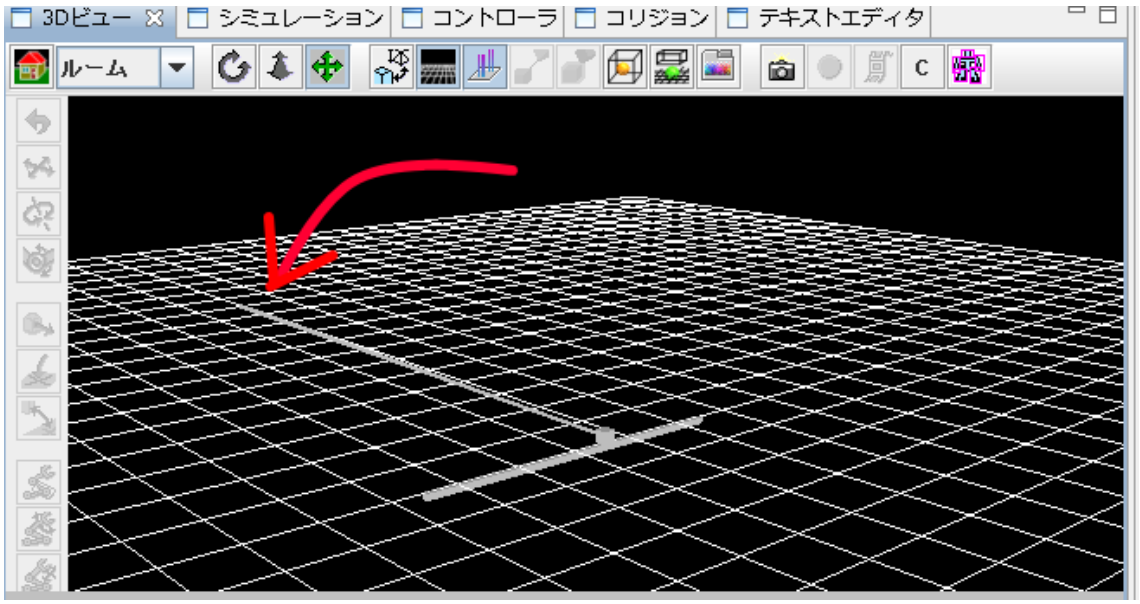


図 44

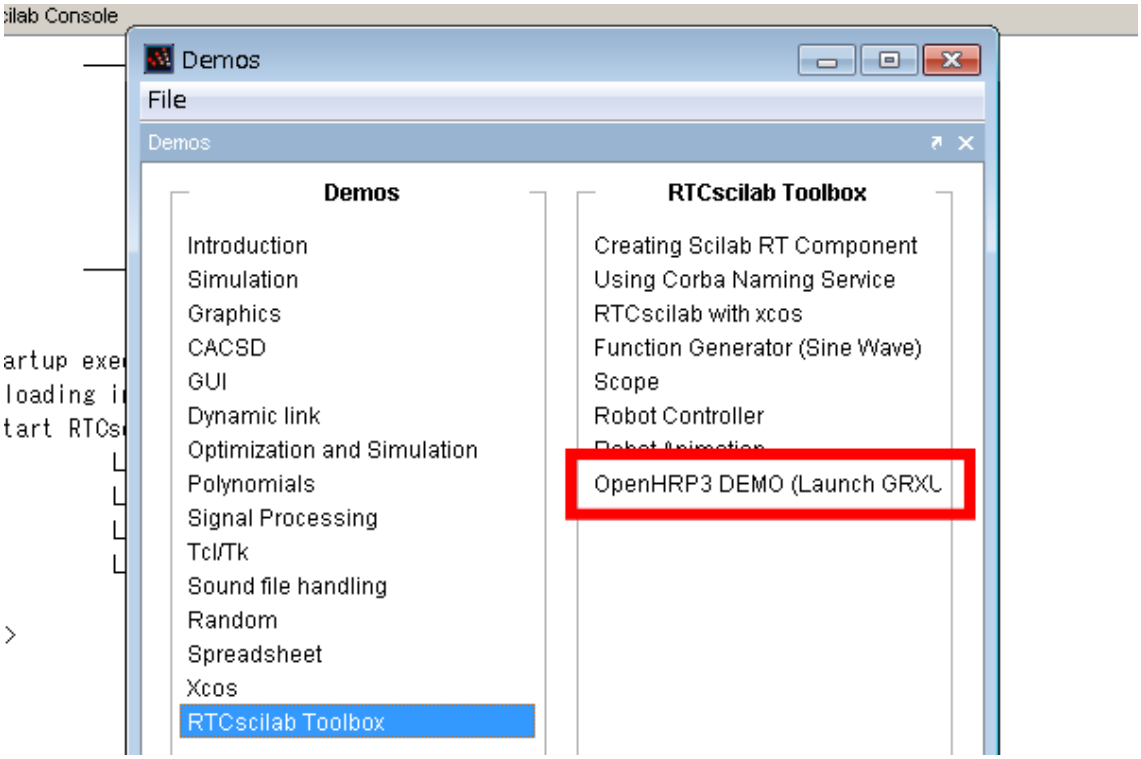


図 45

すると，下図のようなブロック線図が現れます．

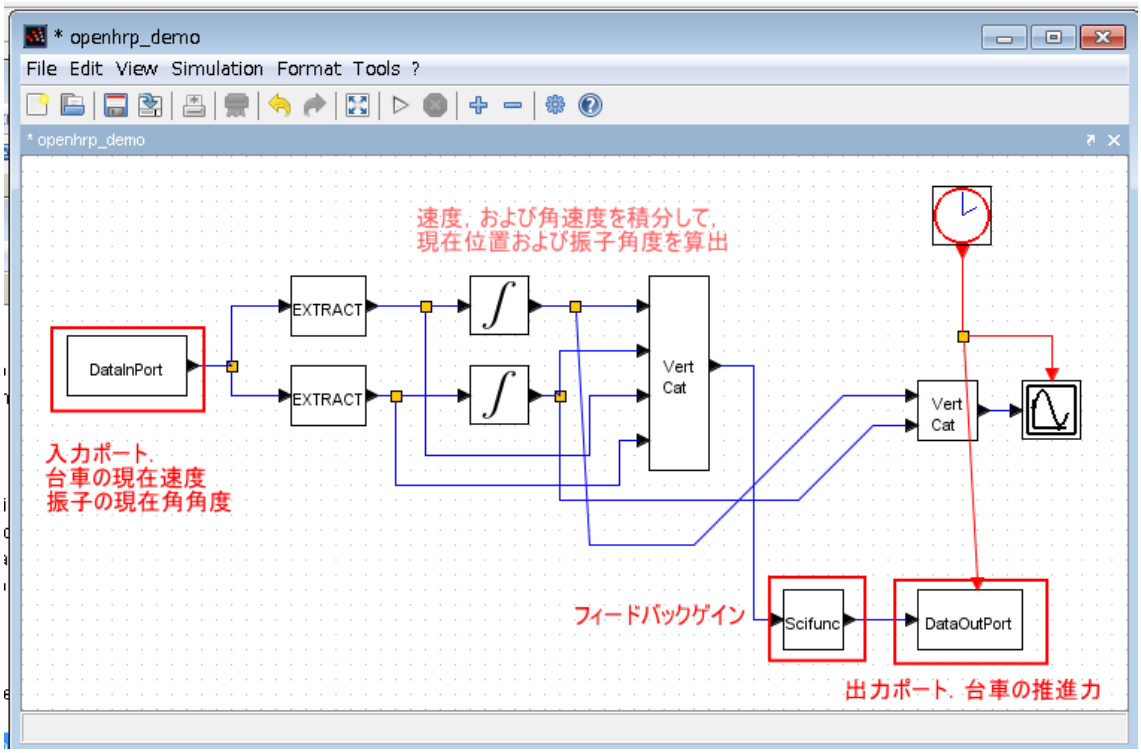


図 46

右上のクロックブロックは OpenHRP3 と同期するための特別なブロックです．これを実行すると，入力ポート 1 つ，出力ポート 1 つをもつ RT コンポーネントが作成されます．

再度，OpenHRP3 を実行すると，自動的にコントローラと接続して，今度は倒立振子がたちます．

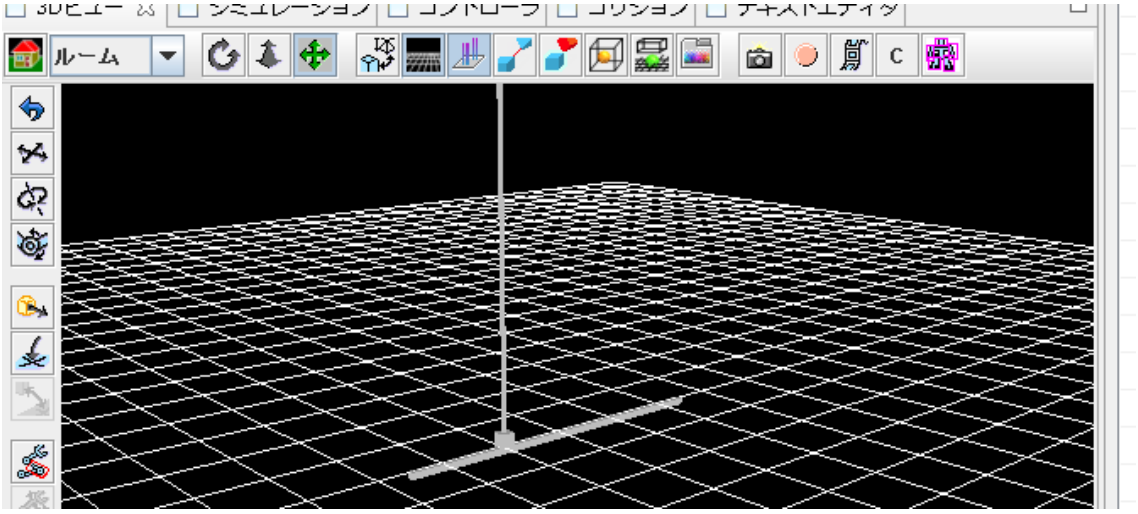


図 47

RTC-scilab で作成したコントローラ「InvPenController」は，下図のようにシミュレータ内のロボットと接続さてお
ります．

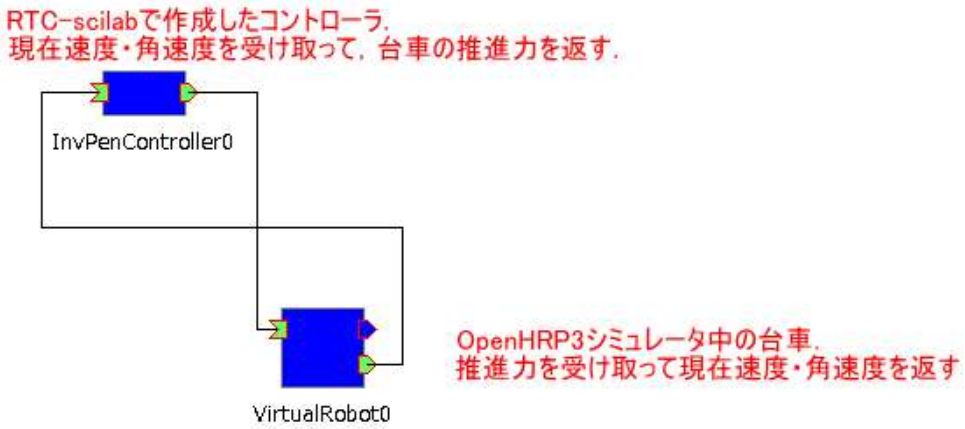
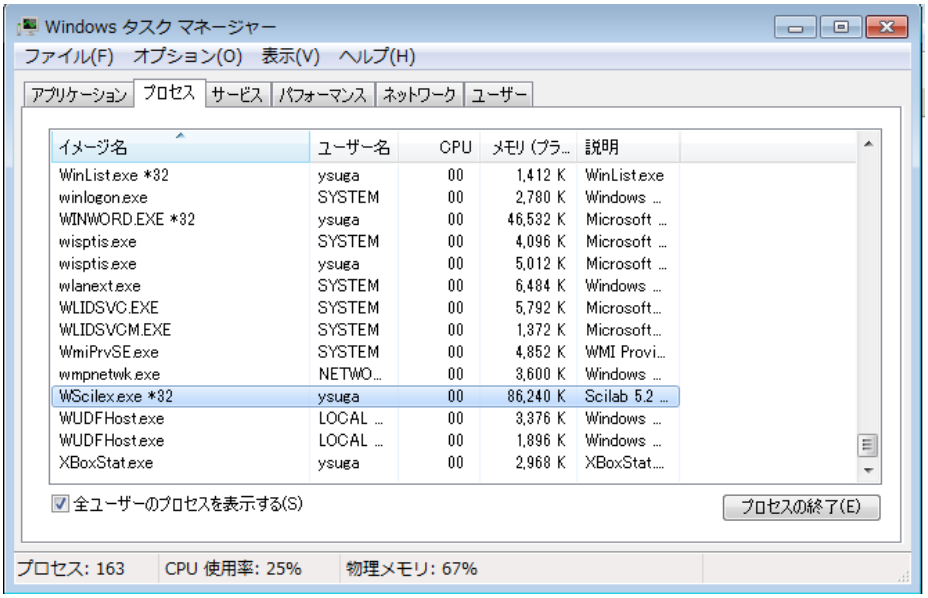


図 48

6. トラブルシューティング

6.1. scilab が応答しない

終了時に exitRTM を行っても上手く終了出来ない場合が Windows では起こります．これについては原因が
良くわかっていません．ひとまずはタスクマネージャーで終了させてください．タスクマネージャーは
Ctrl+Alt+Delete キーで起動するか，ツールバーを右クリックして起動できます．
scilab のプロセス名は「WScilex.exe」です．これを終了させます．


Windows タスク マネージャー
ファイル(F) オプション(O) 表示(V) ヘルプ(H)
アプリケーション プロセス サービス パフォーマンス ネットワーク ユーザー
イメージ名 ユーザー名 CPU メモリ (プラ... 説明
WinList.exe *32 ysuga 00 1,412 K WinList.exe
winlogon.exe SYSTEM 00 2,780 K Windows ...
WINWORD.EXE *32 ysuga 00 46,532 K Microsoft ...
wisptis.exe SYSTEM 00 4,096 K Microsoft ...
wisptis.exe ysuga 00 5,012 K Microsoft ...
wlanext.exe SYSTEM 00 6,484 K Windows ...
WLIDSVC.EXE SYSTEM 00 5,792 K Microsoft...
WLIDSVCM.EXE SYSTEM 00 1,372 K Microsoft...
WmiPrvSE.exe SYSTEM 00 4,852 K WMI Provi...
wmpnetwk.exe NETWO... 00 3,600 K Windows ...
WScilex.exe *32 ysuga 00 86,240 K Scilab 5.2 ...
WUDFHost.exe LOCAL ... 00 3,376 K Windows ...
WUDFHost.exe LOCAL ... 00 1,896 K Windows ...
XBoxStat.exe ysuga 00 2,968 K XBoxStat....
全ユーザーのプロセスを表示する(S)
プロセスの終了(E)
プロセス: 163 CPU 使用率: 25% 物理メモリ: 67%