

効率の良い RTコンポーネント 開発のための支援ツール

2017/8/1

産業技術総合研究所

ロボットイノベーション研究センター

ロボットソフトウェアプラットフォーム研究チーム

高橋

- 開発プロセスについて
 - 開発プロセスの概略
 - サマーキャンプでの開発プロセス
- 開発ツールの紹介
 - 本日紹介するツール
 - 標準ログ機能
 - rtshell
 - ExcelRTC
 - RTStorage
 - RTコンポーネントデバッガ
- 最後に

いきなりですが
「開発プロセス」
という言葉をご存知ですか？

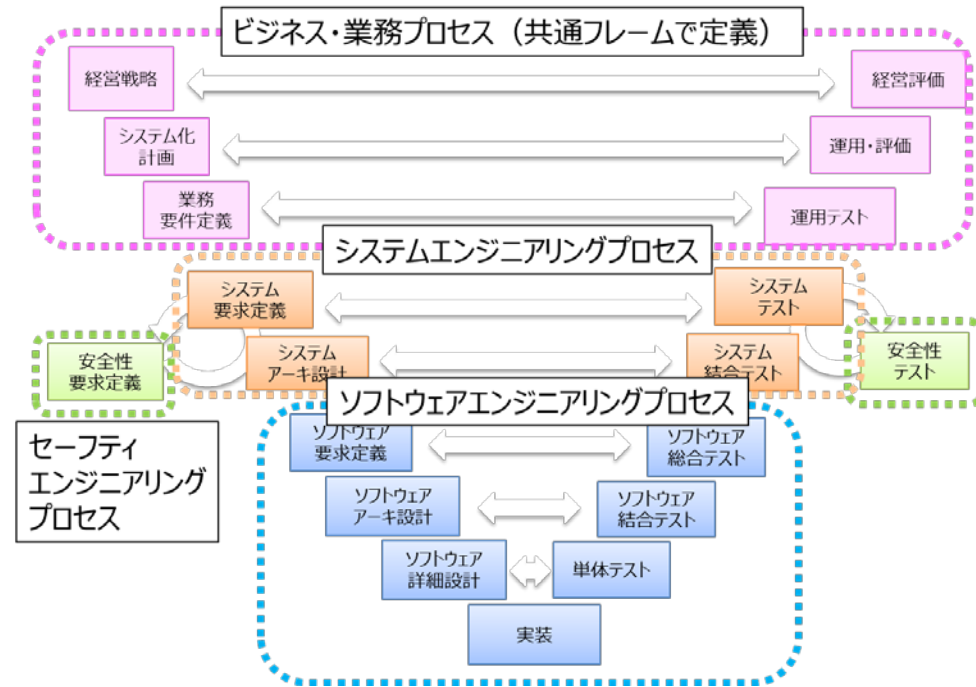
開発工程内の各種タスク・活動のための手法

関係者間の開発方針 共通理解を高めるための ツール

開発はどのくらい
進んでいるの？
遅延 or 前倒し？



次の作業は何を
すれば良いのか・・・



■もし組織のプロセスが確立されていなければ、どうなる？

- (1) 製品(サービス)の品質(Quality)が、安定しない。
- (2) 納期(Delivery)が、不確実となる(納期を必ずしも守れない)。
- (3) 教育訓練の標準化も図れない(作業手順や方法がバラバラ)。
- (4) プロセスの確立や標準化が図れないため、ITによるシステム化が難しい。
- (5) コスト(Cost)は、(1)~(4)の結果として、増大する。

→ 組織がその顧客に提供する製品(サービス)の品質を維持し、または向上させていくためには(顧客満足度の獲得)、組織の活動基盤となる「プロセスの確立」が重要である。

【研究の場合】

- 仮説設定, 研究計画を立てる

- システム構成を考える
 - ハードウェア構成を考える
 - ソフトウェア構成を考える

- ハードウェアを作る・調達する
- ソフトウェアを作る・調達する

- ハードウェアとソフトウェアを組み合わせて,
システムを動かす

- データを取得して仮説を検証する

要求定義

システム設計

ハードウェア設計

ソフトウェア設計

システムテスト

運用

サマーキャンプでの開発に当てはめてみると・・・

どんなシステムを作るか決める
(システム要件の決定)

システム全体で期待通りの動作
をするか確認する
(発表+デモする)

RTM には各プロセスごとに様々なツールが存在し、
開発を助けてくれます
今回はチーム開発のボトルネックになりそうな
テストプロセスで使えるツールについて解説します

シ
(ロボ

ソフ

テストする

テストする

ソフトウェア構成を決める
コンポーネント構成, ポート定義

複数コンポーネントを接続してテストする

ソフトウェアの中身を決める
コンポーネント内部設計

単体コンポーネントをテストする

プログラムを書く

プログラムをテストする

チーム
作業

個人
作業

ここからは時間の都合上、
比較的簡単に使える方法を
5個 紹介します

単体テスト

単体コンポーネントを
テストする

標準ログ機能

rtshell

ExcelRTC

ソフトウェア結合テスト

複数コンポーネントを
接続してテストする

RTStorage

RTコンポーネント
デバッガ

単体コンポーネントをテストする ～ 標準ログ機能 ～ 1/4

名前	OpenRTM-aist 標準ログ機能
作者	産総研
URL	http://officertc.dokkoisho.com/ https://github.com/Nobu19800/ExcelRTC
ライセンス	LGPL
おすすめ度	★★★★★
主な用途	コード実行位置のトレース, エラー処理の検出, 処理値の確認
特長	とにかく手軽, 導入コストが低い



インストール方法

- 特に無し
 - OpenRTM-aist をインストールすれば使える

標準ライブラリに含まれる出力関数 (printf, std::cout) を使う方法もありますがあまりおすすめしません。

- ログの ON/OFF が簡単にできず, 不要なログの消し忘れに繋がる
- 簡単にファイル出力へ変更できない
- 今後 OpenRTM-aist に追加予定の新しいログ機能に追随できない

単体コンポーネントをテストする ～ 標準ログ機能 ～ 2/4

使い方 (ログの埋め込み)

1. 下記サンプルを参考にコード中にログを埋め込む
2. ログレベルに気をつける

C++ の場合

```
RTC::ReturnCode_t xxxRTC::onExecute(RTC::UniqueId ec_id)
{
    //
    // 中略
    //

    RTC_INFO("Hello, Summer Camp!");

    if(terr) {
        RTC_ERR(("Not registered: %d", errno));
    }
}
```

Pythonの場合

```
def __init__(self, manager):
    #
    # 中略
    #

    # set RTC log buffer
    self.log = OpenRTM_aist.Manager.instance().getLogbuf("xxxRTC")

def onExecute(self, ec_id):
    #
    # 中略
    #

    self.log.RTC_INFO("Hello, Summer Camp!")

    if(terr):
        self.log.RTC_ERR("Not registered: " + str(errno))
```

コンポーネントを結合して
動作した際に
必要な情報はこのレベル

ログレベル	想定される用途
FATAL	動作継続不可能な異常
ERROR	動作に影響のある異常
WARN	動作に影響の無い異常
INFO	処理上の重要な情報
NORMAL	処理上の情報
DEBUG	デバッグ用の情報
TRACE	デバッグ用の処理箇所特定情報
VERBOSE	デバッグ用の冗長な情報

個人で開発する際や
詳細なデバッグ時のみ
必要な情報はこのレベル

単体コンポーネントをテストする ～ 標準ログ機能 ～ 3/4

使い方 (ログの出力)

1. まずは標準出力に出力してみる
実行するコンポーネントの `rtc.conf` を書き換える
基本的に実行時のパスにある `rtc.conf` が優先的に読み込まれる

```
logger.enable: YES  
logger.file_name: stdout  
logger.log_level: NORMAL
```

もしくは、起動オプションに付与する

```
xxRTCComp.exe -o "logger.enable:YES" -o "logger.file_name:stdout" -o "logger.log_level:NORMAL"
```

2. 標準出力ではなく、ファイルに出力したい場合は `logger.file_name` を任意のファイル名に変更する

```
logger.file_name: rtc_XXXRTC.log
```

【やってみよう】

OpenRTM-aist に同梱されるサンプルコンポーネント

ConsoleIn, ConsoleOut を

ログレベル=VERBOSE で **標準出力** に

ログを出力する設定で起動し、

RTSystemEditor 上で各ポートを接続、Activate してみる

何か気づいた点がありますか？ ?

【ヒント】

```
"%RTM_ROOT%"Components\C++\Examples\%RTM_VC_VERSION%\ConsoleInComp.exe  
-o "logger.enable:YES" -o "logger.file_name:stdout" -o "logger.log_level:VERBOSE"
```

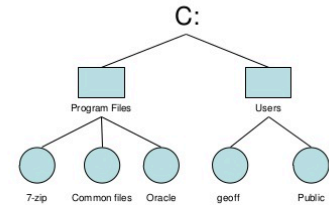
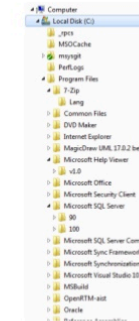
```
"%RTM_ROOT%"Components\C++\Examples\%RTM_VC_VERSION%\ ConsoleOutComp.exe"  
-o "logger.enable:YES" -o "logger.file_name:stdout" -o "logger.log_level:VERBOSE"
```

単体コンポーネントをテストする ～ rtshell ～ 1/2

名前	rtshell
作者	産総研 ジェフさん
URL	https://github.com/gbiggs/rtshell
ライセンス	LGPL
おすすめ度	★★★★☆
主な用途	コンポーネントの状態操作, ポート出力値の確認 ポートへのデータ挿入
特長	Unix ライクなコマンドラインツール OpenRTM – aist 標準機能として取り込まれ, メンテナンスが継続している スクリプト化してしまえば, テストの再現が容易

AIST

rtshellのバーチャルファイルシステム



www.aist.go.jp / ADVANCED INDUSTRIAL SCIENCE AND TECHNOLOGY (AIST)

インストール方法

- 【Windows】 OpenRTM-aist をインストールすれば使える (バージョン 1.1.2 以降)
- 【Linux】 pip でインストール
 - sudo pip install rtshell

【やってみよう】

OpenRTM-aist に同梱されるサンプルコンポーネント

ConsoleIn の **out** ポートを **rtprint** で標準出力するバッチファイルと,

ConsoleOut の **in** ポートを **rtinject** でデータ (123) を送信するバッチファイルを作成してみる

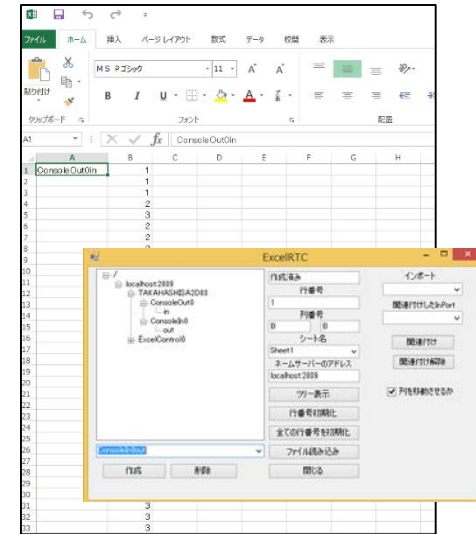
【ヒント】

`rtprint /localhost/TAKAHASHISA2D83.host_cxt/ConsoleIn0.rtc:out`
を `rtprint_consolein.bat` として保存し、コマンドプロンプトで実行

`rtinject /localhost/TAKAHASHISA2D83.host_cxt/ConsoleOut0.rtc:in -c "RTC.TimedLong({time}, 1)"`
を `rtinject_consoleout.bat` として保存し、コマンドプロンプトで実行

単体コンポーネントをテストする ~ExcelRTC~ 1/2

名前	ExcelRTC
作者	産総研 宮本さん
URL	http://officertc.dokkoisho.com/ https://github.com/Nobu19800/ExcelRTC
ライセンス	???
おすすめ度	★★★☆☆
主な用途	ポート出力値の確認, ポートへのデータ挿入
特長	CSVに直接データが反映できるのでデータ解析がしやすい 宮本さんに要望を言えばすぐに修正してくれる可能性がある



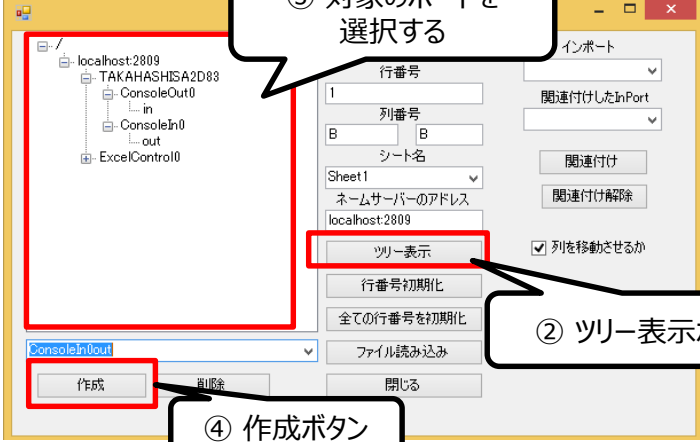
インストール方法

- レポジトリからソースコードをダウンロード
 - git clone <https://github.com/Nobu19800/ExcelRTC.git>
- Release フォルダに含まれる ExcelControlComp.exe を利用する

単体コンポーネントをテストする ～ExcelRTC～ 2/2

使い方

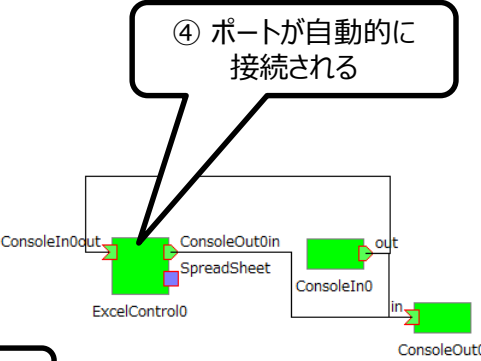
1. ExcelControlComp.exe を実行
2. 表示されたウィンドウのツリー表示ボタンを押下
3. 表示されたツリー情報から対象のポートを選択
4. 作成ボタン押下でポートが自動的に接続し、データが Excel シートに出力される



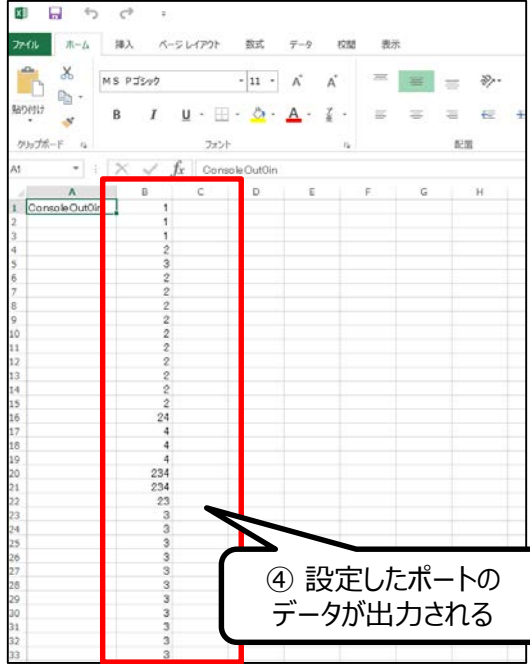
③ 対象のポートを選択する

② ツリー表示ボタン

④ 作成ボタン



④ ポートが自動的に接続される

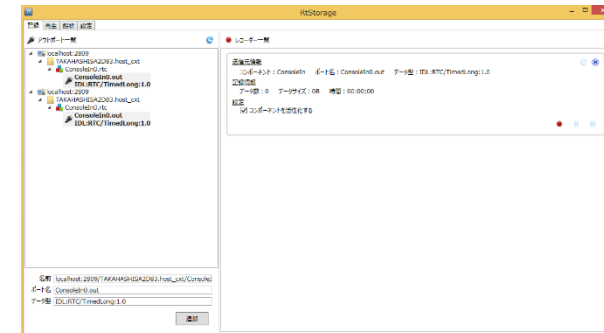


④ 設定したポートのデータが出力される

ConsoleOut0	B	C	D	E	F	G	H
1	1						
2	1						
3	2						
4	3						
5	2						
6	2						
7	2						
8	2						
9	2						
10	2						
11	2						
12	2						
13	2						
14	2						
15	2						
16	24						
17	4						
18	4						
19	4						
20	234						
21	234						
22	23						
23	3						
24	3						
25	3						
26	3						
27	3						
28	3						
29	3						
30	3						
31	3						
32	3						
33	3						

複数コンポーネントを接続してテストする ~RTStorage~ 1/2

名前	RTStorage
作者	zoetrope さん
URL	https://github.com/zoetrope/RtStorage
ライセンス	Microsoft Permissive License (Ms-PL)
おすすめ度	★★★★☆☆
主な用途	ポート出力値の確認, ポートへのデータ挿入
特長	シンプルな UI, データ検索機能 独自定義IDLデータを取り扱える



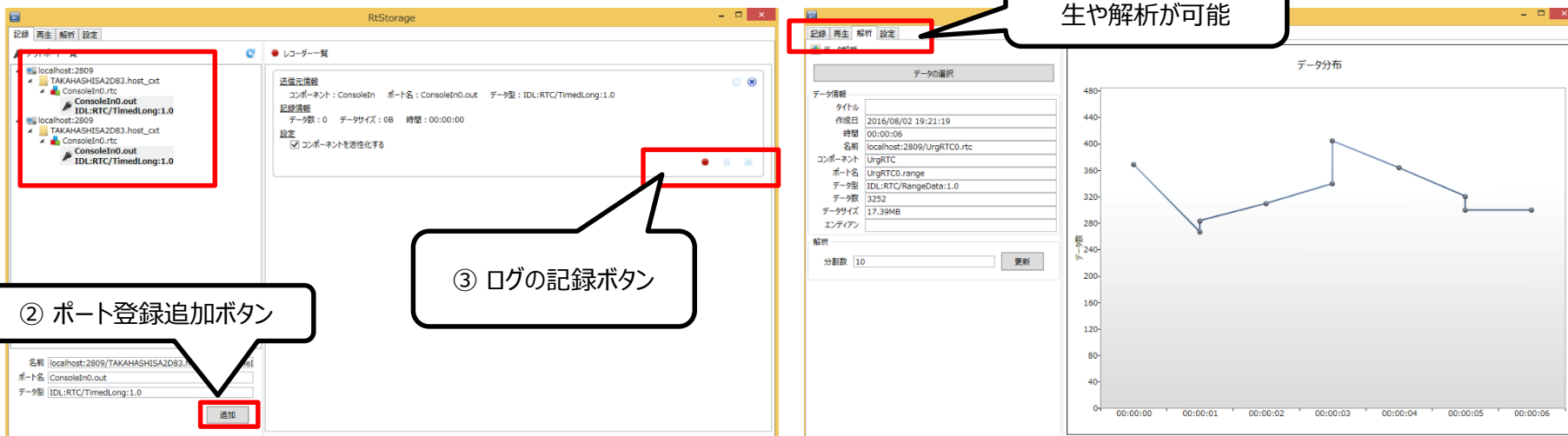
インストール方法

- レポジトリからソースコードをダウンロード
 - git clone <https://github.com/zoetrope/RtStorage.git>
- Clone したディレクトリに含まれる RtStorage.sln を VisualStudio で開き, ソリューションのビルドを実行
- ビルドに成功すると RtStorage.exe が生成される

複数コンポーネントを接続してテストする ~RTStorage~ 2/2

使い方

1. RtStorage.exe を実行
2. 実行中の RTC が表示されるので記録したいポートのペアを選択し、追加ボタンを押下
3. レコード一覧に表示されたボタンで、ログの記録、一時停止、停止操作ができる
4. 上部のタブを切り替えることで、ログの再生や解析が可能

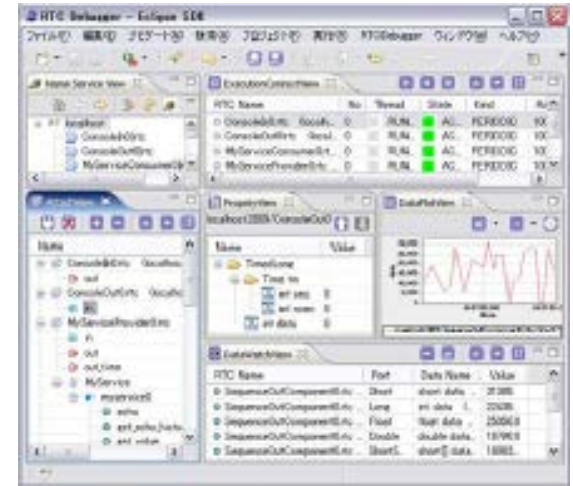


The screenshot displays the RtStorage application interface, which is divided into several sections:

- Port Selection (Left):** A tree view shows the system's network configuration. A red box highlights the 'ConsoleIn0.rtc' component under the 'localhost:2809' host. Below this, a '追加' (Add) button is highlighted with a red box. A callout bubble labeled '② ポート登録追加ボタン' (Port registration add button) points to this button.
- Recording Controls (Middle):** A 'レコーダー一覧' (Recorder List) section shows the selected component and its settings. A red box highlights a set of control buttons (record, stop, play) at the bottom right of this section. A callout bubble labeled '③ ログの記録ボタン' (Log recording button) points to these buttons.
- Data Selection and Analysis (Right):** A 'データの選択' (Data Selection) panel shows details for the selected data, including title, creation date, time, name, component, port name, data count, data size, and encoding. Below this, a '解析' (Analysis) section has a '分割数' (Number of divisions) set to 10 and an '更新' (Update) button. A callout bubble labeled '④ タブの切替でログ再生や解析が可能' (Tab switching allows log playback and analysis) points to the '記録' (Record), '再生' (Play), and '解析' (Analysis) tabs at the top of this panel.
- Data Distribution Graph (Far Right):** A line graph titled 'データ分布' (Data Distribution) shows the data size over time. The x-axis represents time from 00:00:00 to 00:00:06, and the y-axis represents data size in bytes, ranging from 0 to 480. The graph shows a fluctuating data size, peaking around 00:00:03.

複数コンポーネントを接続してテストする ～RTコンポーネントデバッガ～ 1/2

名前	RTコンポーネントデバッガ (RTC Debugger)
作者	セック http://www.sec.co.jp/
URL	http://www.sec.co.jp/robot/download_tool.html
ライセンス	非営利利用のみ
おすすめ度	★★★★☆
主な用途	ポート出力値の確認, ポートへのデータ挿入, 画像データのプレビュー
特長	Eclipse 上で動作するので SystemEditor 等と合 わせて使い易い



インストール方法

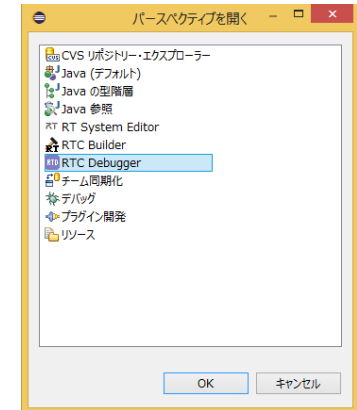
- 以下URLからアーカイブをダウンロード
 - http://www.sec.co.jp/robot/download_tool.html
- 解凍したフォルダ内でスクリプトを実行
 - 【Windows】 コマンドプロンプトで下記を実行
install_1.1.0.20120522.bat
 - 【Linux】 シェルで下記を実行
chmod a+x install_1.1.0.20120522.sh && ./install_1.1.0.20120522.sh
- 生成されたフォルダ一式 (org.openrtm.debugger_1.x.x.yyyymmdd) をOpenRTMのインストールされたフォルダの plugins 以下にコピー
 - 【Windows】 (例) C:¥Program Files¥OpenRTM-aist¥1.1.2¥utils¥OpenRTP¥plugins
 - 【Linux】 (例) /usr/share/openrtm-1.1/eclipse/plugins/

複数コンポーネントを接続してテストする ～ RTコンポーネントデバッガ～ 2/2

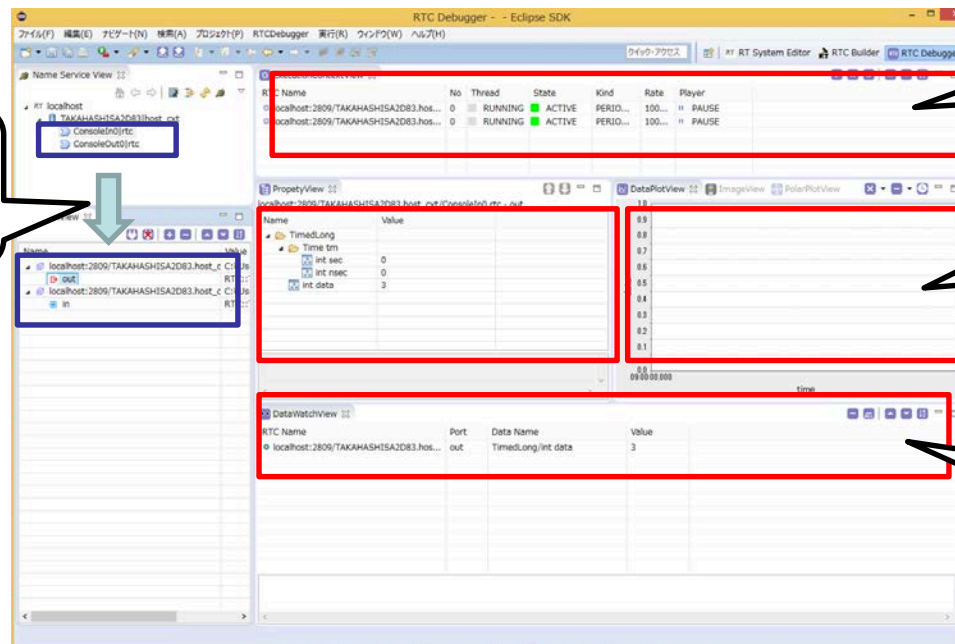
使い方

1. OpenRTP を起動
2. メニュータブの“ウィンドウ”→“パースペクティブを開く”→“その他”
→“RTC Debugger” を選択
3. 画面左上の NameServerView から RTC を 左下の Attach View に
ドラッグ&ドロップする

※同梱されているマニュアルを読むとさらに色々な機能が載っています



③ NameServerView
から RTC を Attach
View にドラッグ&ドロップ



Name	No	Thread	State	Kind	Rate	Player
localhost:2809/TAKAHASHISA2D83.hos...	0	RUNNING	ACTIVE	PERIO...	100...	PAUSE
localhost:2809/TAKAHASHISA2D83.hos...	0	RUNNING	ACTIVE	PERIO...	100...	PAUSE

Name	Value
TimeedLong	
Time tm	
mt sec	0
mt msec	0
int data	3

RTC Name	Port	Data Name	Value
localhost:2809/TAKAHASHISA2D83.hos...	out	TimeedLong/int data	3

RTCの状態などを表示

画像プレビューや
データをプロットしたグ
ラフを出力

ポート間のデータの記録や
データの再生

RTC おすすめ開発スタイル

- コードができあがる前から、ちょこちょこテストする
 - コンパイル→実行→トレースログ確認など
- 実装 → テスト → 実装 → … のサイクルを早めるためにまずは全ての RTC を PC 上で Python で動作させてみる
 - 性能ボトルネックは C++ で書き直す
 - RTC はテンプレートから各言語のコードが自動生成できるので移植が容易
- テストを自動化, 省力化する
 - スクリプトなどを利用してテストへの心理的な負担を減らす
- コードを修正したら, (自信があっても) まず単体テストにかける
 - デグレード (修正による新規バグ埋め込み) を検出できる

OpenRTM-aist 1.2.0 でログ機能が変わる！

- 標準ログ機能（SystemLogger）の改善
- logger がプラグイン対応し、標準以外のモジュールも使用可能に！
- サンプルとしてOSS のログ転送フレームワーク **fluent-bit** を使った logger が同梱
 - ネットワーク越しにログを集約することも可能に！

