RTコンポーネント操作マニュアル

動的入出力ポート
および動作テスト用コンポーネント

2022年9月6日版

芝浦工業大学

デザイン工学部デザイン工学科

佐々木 毅

更新履歴

2019 年 4 月 9 日版	第 1 版。「効率的な RT システム開発および運用のための汎用ビューワコンポーネント」(GnuplotViewer) マニュアルから動的入出力ポートに関係する項目を独立。OpenRTM-aist 1.2.0 に合わせ記述を更新。
2022 年 9 月 6 日版	4 章の問合せ先の変更。

目 次

1 開発概要 **1**

2 動的入出力ポートの使用方法 **1**

2.1 動的入出力ポート DynamicInPort, DynamicOutPort の公開メンバ 1

2.1.1 メンバ変数 1

2.1.2 メンバ関数 2

2.2 動的入出力ポートの使用手順 2

2.2.1 ファイルのインクルード 2

2.2.2 変数の宣言 3

2.2.3 コンストラクタによる初期化 3

2.2.4 ポートの追加 3

2.2.5 ポートへのデータ入出力、取得したデータの参照 4

2.2.6 ポートの削除 4

3 テストコンポーネントの使用方法 **5**

3.1 コンポーネントの説明 5

3.1.1 動的入力ポートテストコンポーネント (DynamicInPortTest) 5

3.1.2 コンソール文字列入力コンポーネント (ConsoleInString) 5

3.2 RTSystemEditor によるシステム構築の手順 6

3.3 動的入出力ポート動作テスト用サンプルコンポーネント DynamicInPortTest の使用 6

4 お問い合わせ **8**

1 開発概要

DynamicPort は RT コンポーネントの入出力ポートの動的追加 · 削除機能を実現します。この機能はクラス化されており、開発しているコンポーネントで簡単に利用することができます。アクティブ化時に Configuration からポートの数を読み込み、ポートの生成を行うようなことも、アクティブ状態のままポートの追加や削除を行うことも可能です。

また、この機能の使用方法を示すため、以下のテスト用コンポーネントを作成しました。

- 動的入力ポートテストコンポーネント (DynamicInPortTest)

  動的入力ポートのプログラム例および動作テスト用コンポーネント。

- コンソール文字列入力コンポーネント (ConsoleInString)

  コンソールから入力した文字列を OutPort に出力するコンポーネント。

DynamicPort およびテストコンポーネント群は現バージョンについては Windows にて開発・動作確認を行っていますが、過去のバージョンでは Linux でも動作することを確認しています。開発環境は以下の通りです。

- OS: Windows 10 (64bit 版)
- RT ミドルウエア: OpenRTM-aist C++ v1.2.0
- コンパイラ: Microsoft Visual Studio 2017 Community (VC++ 2017)
- Python: Python 3.7
- CMake: CMake 3.14.1

動的入出力ポートの使用方法については 2 章を、テストコンポーネントの詳細については 3 章を参照してください。

2 動的入出力ポートの使用方法

本章では、動的入出力ポートクラスである `DynamicInPort`, `DynamicOutPort` に関して説明します。

2.1 動的入出力ポート `DynamicInPort`, `DynamicOutPort` の公開メンバ

ここでは、動的入出力ポート `DynamicInPort`, `DynamicOutPort` の公開メンバ変数、メンバ関数に関して概説します。これらの具体的な使用方法 · 手順については 2.2 節を参照してください。

2.1.1 メンバ変数

公開メンバ変数は以下のとおりです。変数名は `DynamicInPort`, `DynamicOutPort` で共通ですが、変数 `m_port` の型はそれぞれ InPortVect<*DataType*>, OutPortVect<*DataType*>となります。

変数名	型	説明
m_data	PortDataVect<*DataType*>	獲得したデータもしくは出力したいデータを格納する変数の動的配列。個々の要素が通常の入出力ポートの `m_`*PortName* に対応する。
m_port	InPortVect<*DataType*> OutPortVect<*DataType*>	入出力ポートの動的配列。個々の要素が通常の入出力ポートの `m_`*PortName*`In` や `m_`*PortName*`Out` に対応する。

2.1.2 メンバ関数

コンストラクタ、デストラクタを除く公開メンバ関数は以下のとおりです。関数名、機能は DynamicInPort, DynamicOutPort で共通です。

関数名	引数	戻り値の型	説明
addPort	なし	int	入出力ポートを 1 つ追加する。追加されたポートの名前には連番が振られる（これまでに削除されたポートがある場合でも、累積で連番が振られていく）。追加に成功すれば 0 を、エラーの時はそれ以外の値を返す。実際にこのポートを利用するには RTM::RTObject_impl::addInPort() / addOutPort() によるポートの登録が必要。
resetPort	unsigned int	int	引数で指定されたポート番号（InPortVect / OutPortVect の要素番号）の入出力ポートをリセットする。リセット後、ポート名の連番やポート番号は現在のまま変化しない。RTM::RTObject_impl::removeInPort() / removeOutPort() により登録解除を行ったポートを RTM::RTObject_impl::addInPort() / addOutPort() で再登録する場合は予めこの関数でリセットする必要がある。リセットに成功すれば 0 を、エラーの時はそれ以外の値を返す。
deletePort	unsiged int	int	引数で指定されたポート番号（InPortVect / OutPortVect の要素番号）の入出力ポートを削除する。引数が指定されない場合は最後のポートを削除する。削除に成功すれば 0 を、エラーの時はそれ以外の値を返す。ポートを削除するとそれより番号の大きいポートのポート番号は 1 つずつ詰められる（ポート名の連番はそのまま）。ポートを削除する前には RTM::RTObject_impl::removeInPort() / removeOutPort() によるポートの登録解除が必要。
getSize	なし	unsigned int	追加されているポートの数を返す。ポート番号（InPortVect / OutPortVect の要素番号）は 0 番から順に付けられるので、利用可能なポートは 0 番から getSize()-1 番までである。
getName	int	const char*	引数で指定されたポート番号（InPortVect / OutPortVect の要素番号）のポート名を返す。ポート名はポートの登録の際に利用する。

2.2 動的入出力ポートの使用手順

動的入出力ポートを使用する手順を以下に示します。2.2.1 節, 2.2.2 節はコンポーネントの.h ファイルに追加する処理、2.2.3 節から 2.2.6 節は.cpp ファイルに追加する処理です。

2.2.1 ファイルのインクルード

コンポーネントの.h ファイルの

```
using namespace RTC;
```

の直前で dynamic_port.hpp ファイルをインクルードします。

ファイルのインクルード

```
#include "dynamic_port.hpp"
```

なお、インクルードパスは適宜変更するか開発環境の設定に追加してください。

2.2.2 変数の宣言

• 動的 InPort を宣言する場合:

コンポーネントの.h ファイルの<rtc-template block="inport_declare">内にて宣言を行います。

動的 InPort の宣言

```
DynamicInPort<DataType> VariableName;
```

〈例〉m_ShortDataIn という変数名の TimedShortSeq 型のデータを受け取る動的 InPort の宣言

```
DynamicInPort<TimedShortSeq> m_ShortDataIn;
```

ここで、*DataType* はやりとりするデータの型、*VariableName* は変数名です。変数名は自由につけてかまいませんが、OpenRTM-aist の通常の入出力ポートの名前の付け方にならい、m_*PortName*In としておくとわかりやすいと思います。

• 動的 OutPort を宣言する場合:

コンポーネントの.h ファイルの<rtc-template block="outport_declare">内にて宣言を行います。DynamicInPort が DynamicOutPort になることを除いては、動的 InPort の場合と同様です。

動的 OutPort の宣言

```
DynamicOutPort<DataType> VariableName;
```

〈例〉m_ShortDataOut という変数名の TimedShortSeq 型のデータを出力する動的 OutPort の宣言

```
DynamicOutPort<TimedShortSeq> m_ShortDataOut;
```

ここでは InPort と OutPort のそれぞれについて記述しましたが、基本的に InPort と OutPort で手順は変わりませんので、以下では InPort に対してのみ説明します。OutPort に対しては In の部分を Out に変更してください。

2.2.3 コンストラクタによる初期化

コンポーネントの.cpp ファイルの<rtc-template block="initializer">内にてコンストラクタに引数を与えて初期化を行います。

コンストラクタによる初期化

```
VariableName(PortName)
```

〈例〉ShortDataIn というポート名で変数 m_ShortDataIn を初期化

```
m_ShortDataIn("ShortDataIn")
```

ここで、*VariableName* は 2.2.2 節で宣言した変数名、*PortName* は RTSystemEditor 上で表示されるポートの名前です。ポートを追加すると、ポートの名前が順に *PortName*0、*PortName*1、*PortName*2、... となります。ポート名も自由に付けることができますが、前述のとおり変数名を m_*PortName*In としておき、*PortName* を設定する（もしくは In の部分まで *PortName* に含める）とわかりやすくなると思います。

2.2.4 ポートの追加

ポートを追加したい時には以下のようにポートの追加と登録を行います。ポートの追加には動的入出力ポートのメンバ関数である *VariableName*.addPort() を使います。ポートを複数個追加したい場合は以下の処理を追加したいポートの数だけ繰り返してください。

動的 InPort の追加

VariableName.addPort();
addInPort(*VariableName*.getName(*VariableName*.getSize()-1),
VariableName.m_port[*VariableName*.getSize()-1]);

〈例〉変数 m_ShortDataIn に InPort を 1 つ追加し、それを登録

```
m_ShortDataIn.addPort();
addInPort(m_ShortDataIn.getName(m_ShortDataIn.getSize()-1),
          m_ShortDataIn.m_port[m_ShortDataIn.getSize()-1]);
```

2.2.5 ポートへのデータ入出力、取得したデータの参照

OpenRTM-aist では、ポートからのデータの入力、ポートへのデータの出力等は m_*PortName*In や m_*PortName*Out という変数のメンバ関数を用いて行い、獲得したデータや出力したいデータは m_*PortName* という変数に格納されます*。2.1.2 節で述べたように、動的入出力ポートの場合にはこれらがそれぞれ *VariableName*.m_port[*PortNumber*], *VariableName*.m_data[*PortNumber*] となります。具体的な使用例を以下に示します。

通常の入出力ポートと動的入出力ポートの対応関係

	通常の入出力ポート	動的入出力ポート
入出力ポート変数	m_*PortName*In / m_*PortName*Out	*VariableName*.m_port[*PortNumber*]
データ格納変数	m_*PortName*	*VariableName*.m_data[*PortNumber*]

〈例〉

変数 m_ShortDataIn の i 番目のポートからデータの読み込み

```
m_ShortDataIn.m_port[i].read()
```

変数 m_ShortDataIn の i 番目のポートに新しいデータが送られてきたかどうかチェック

```
m_ShortDataIn.m_port[i].isNew()
```

変数 m_ShortDataIn の i 番目のポートから得た可変長配列データの長さを獲得

```
m_ShortDataIn.m_data[i].data.length()
```

変数 m_ShortDataIn の i 番目のポートから得た可変長配列データの j 番目のデータを参照

```
m_ShortDataIn.m_data[i].data[j]
```

2.2.6 ポートの削除

ポートを削除したい時にはまずポートの登録を解除し、次に削除を行います。ポートの削除には動的入出力ポートのメンバ関数である *VariableName*.deletePort(*PortNumber*) を使います。*PortNumber* は削除したいポートの番号です。ポート番号はその動的入出力ポート変数が現在保持するポートに対して順に 0,1,2,... と付けられています。つまり、ポートを削除するとそれより番号の大きいポートのポート番号は 1 つずつ詰められます。例えば 3 番と 5 番のポートを削除したい場合、先に 3 番のポートを削除すると、5 番のポートは 1 つ分番号が詰められて 4 番になるため注意が必要です。もちろん、先に 5 番のポートを削除して次に 3 番のポートを削除すれば問題はありません。

*RTCBuilder を用いて Var Name の設定を省略した場合の変数名。RTCBuilder を用いて Var Name を指定した場合にはそれがデータを格納する変数の名前となる。

動的 InPort の削除

```
removeInPort(m_ShortDataIn.m_port[PortNumber]);
m_ShortDataIn.deletePort(PortNumber);
```

〈例〉変数 m_ShortDataIn のすべてのポートを登録解除し、削除

```
while(m_ShortDataIn.getSize() > 0){
   removeInPort(m_ShortDataIn.m_port[m_ShortDataIn.getSize()-1]);
   m_ShortDataIn.deletePort(m_ShortDataIn.getSize()-1);
}
```

3 テストコンポーネントの使用方法

本章では、動的入出力ポートのポートの追加 · 削除機能の動作テストサンプルとして用意している DynamicInPortTest コンポーネントの使用方法を説明します。

3.1 コンポーネントの説明

3.1.1 動的入力ポートテストコンポーネント (DynamicInPortTest)

DynamicInPortTest は、動的入力ポートのプログラム例の提示と動作テスト用のコンポーネントです。アクティブ状態で入力ポートの InPortManip に文字列が入力されると、それに応じて動的入力ポートの追加と削除を行います。入力ポートの InPortManip に入力された文字列が数字のみの場合、その数字の番号の入力ポートがあればそのポートを削除します。入力ポートの InPortManip に入力された文字列に数字以外の文字が含まれていた場合、その文字数だけポートを追加します。動的入力ポート StringData に入力があった場合、そのポートの名前、番号と文字列をコンソールに出力します。非アクティブ化するとすべての動的入力ポートを削除します。

- InPort

名称	型	説明
InPortManip	TimedString	動的入力ポートの追加・削除コマンド。
StringData	DynamicInPort<TimedString>	コンソールに出力する文字列データ。

- OutPort

  なし

- Configuration 変数

  なし

- サービスポート

  なし

3.1.2 コンソール文字列入力コンポーネント (ConsoleInString)

ConsoleInString は、OpenRTM-aist のホームページ (http://www.is.aist.go.jp/rt/OpenRTM-aist/) にサンプルとして取り上げられている ConsoleIn コンポーネントの出力の型を TimedString としたものです。アクティブ化すると入力を促すメッセージがコンソールに表示されます。コンソールから入力を行うと、その文字列が OutPort に出力されます。

- InPort

  なし

- OutPort

名称	型	説明
OutString	TimedString	コンソールから入力された文字列。

- Configuration 変数

  なし

- サービスポート

  なし

3.2 RTSystemEditor によるシステム構築の手順

RTSystemEditor を用いたシステム構築は通常、以下の手順で行われます。

RTSystemEditor を用いたシステム構築の手順

1. ネームサーバの起動
2. RTSystemEditor の起動
3. ネームサーバへの接続
4. コンポーネントの起動
5. システムの構築と実行

これらの手順はどのようなシステムでも共通ですので、操作方法は OpenRTM-aist のホームページ (`http://www.openrtm.org/`) を参照してください。以下の節では手順 5 に関して説明します。

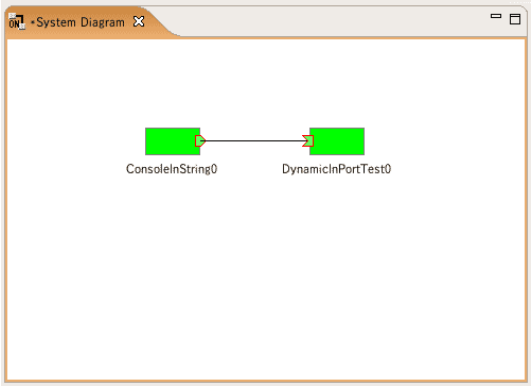
3.3 動的入出力ポート動作テスト用サンプルコンポーネント DynamicInPortTest の使用

この動作テストに使用するコンポーネントの種類とその数は

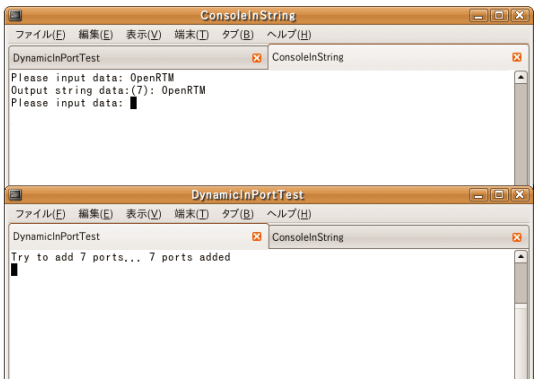
- 動的入力ポートテストコンポーネント (DynamicInPortTest) · · · · · · 1 つ
- コンソール文字列入力コンポーネント (ConsoleInString) · · · · · · 1 つ

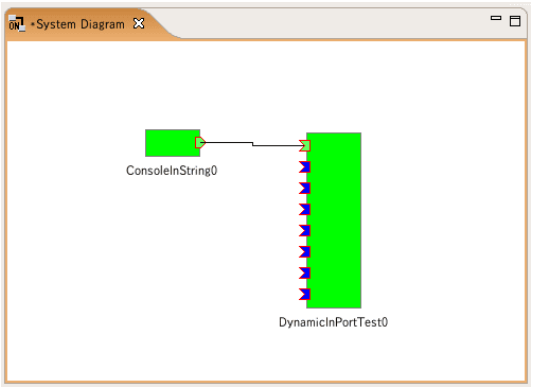
の計 2 つです。

まず、ConsoleInString の OutString ポートと DynamicInPortTest の InPortManip ポートを接続し、アクティブ化します。

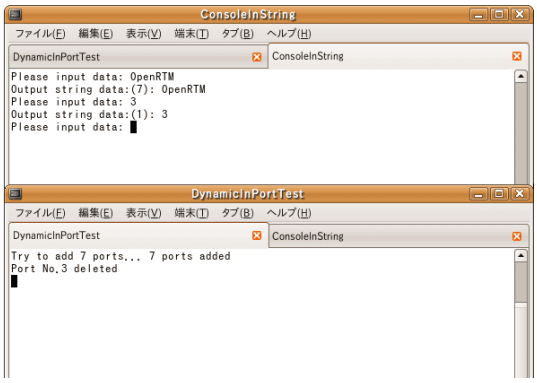


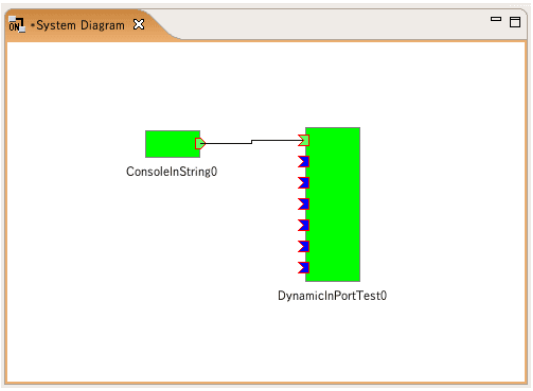
この状態で、数字以外の文字を含む文字列を ConsoleInString のコンソールから入力すると、その数だけ Timed-String 型の入力ポートが追加されます。



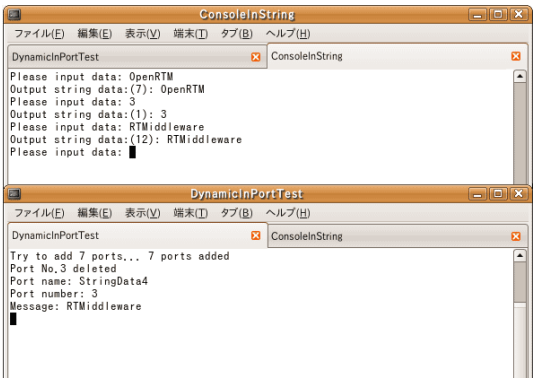


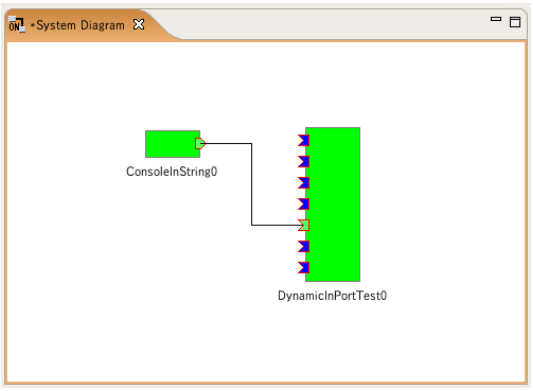
また、数字のみの文字列を ConsoleInString のコンソールから入力すると、そのポート番号の入力ポートが削除されます。ポート番号は動的入力ポートの生成されたものから順に 0,1,2,· · · と付けられ、削除された番号は詰められます（2.2.6 節を参照）。対応する番号のポートが存在しない場合には何も行われません。





さらに、生成された動的入力ポートに対し入力を行うとそのポートの名前（StringData#, ここで#は削除されたポートを含め累積何番目に生成されたのかを表す数）、ポート番号と入力された文字列がコンソールに出力されます。





4　お問い合わせ

本提案につきましては、まだ改善の余地があるものと考えております。ご要望、バグ報告、マニュアルの記述の不備等に関しましては、芝浦工業大学デザイン工学部デザイン工学科の佐々木までご連絡ください。

問合せ先

〒 135-8548

東京都江東区豊洲 3-7-5

芝浦工業大学 本部棟 6 階 06K10 佐々木研究室

Tel:03-5859-8834

sasaki-t at ieee. org