OpenRTM-aist (Python) - 機能 #3407

⑧　ダイレクトデータポート接続機能

2015/12/22 09:45 - n-ando

ステータス:	終了	**開始日:**	2015/12/22
優先度:	通常	**期日:**	2016/03/25
担当者:	miyamoto	**進捗率:**	100%
カテゴリ:		**予定工数:**	30.00時間
対象バージョン:	RELEASE_1_2_0		

説明

同一プロセス内のデータポート接続において、OutPortがInPortの変数に直接書き込む（push型）、またはInPortがOutPortの変数を直接読む（Pull型）形でデータ転送をする方式を実装すること。

関係しているリビジョン

リビジョン 637 - 2016/02/01 13:48 - miyamoto

[compat,bugfix,->RELENG_1_2] bug fix. refs #3407

リビジョン 639 - 2016/02/01 14:15 - miyamoto

[incompat,new_func,new_file,->RELENG_1_2] Direct data put functionality between data ports has been implemented. refs #3407

リビジョン 649 - 2016/02/01 19:53 - miyamoto

[compat,bugfix,->RELENG_1_2] bug fix. refs #3407

リビジョン 650 - 2016/02/01 20:11 - miyamoto

[incompat,new_func,new_file,->RELENG_1_2] Direct data put functionality between data ports has been implemented. refs #3407

リビジョン 669 - 2016/02/27 10:35 - miyamoto

[incompat,bugfix,func,->RELENG_1_2] bug fix. refs #3407

履歴

#1 - 2016/01/14 16:19 - miyamoto

- 期日 を 2016/03/25 にセット

- 担当者 を miyamoto にセット

- 対象バージョン を RELEASE_1_2_0 にセット

- 進捗率 を 0 から 80 に変更

- 予定工数 を 30.00時間 にセット

#2 - 2016/01/14 22:45 - miyamoto

- ファイル test_DirectPort.py を追加

実装

Push型
以下のクラスを実装した。

InPortDirectProvider
インポートプロバイダクラス
InPortProviderクラスを継承してある。
InterfaceTypeは「direct」に設定してある。

InPortDirectConsumer
インポートコンシュマークラス
InPortConsumerクラスを継承してある。

InPortDirectProviderのファクトリを登録するInPortDirectProviderInit関数、InPortDirectConsumerのファクトリを登録するInPortDirectConsumerInit関数をFactoryInit関数で呼び出すことで「direct」という名前のInterfaceTypeを有効にしている。

OutPortBaseクラスに以下の関数を実装した。

getLocalInPort関数:接続したインポートのサーバントを取得する

またInPortのサーバントの設定のためにcreateConnector関数にコードを追加した。

OutPortPushConnectorクラスに以下の関数を実装した。

setInPort関数:ダイレクトで接続する場合にインポートのサーバントを設定する

データの変数への書き込みのためにwrite関数にコードを追加した。
変数宣言のためにコンストラクタにコードを追加した。

InPortクラスに以下の関数を実装した。

write関数:変数_valueにデータを書き込む

変数読み込みのためにread関数にコードを追加した。
データ
変数宣言のためにコンストラクタにコードを追加した。

まずOutPortBaseオブジェクトのcreateConnector関数内でgetLocalInPort関数によりインポートのサーバントを取得し、OutPortPushConnectorオブジェクトのsetInPort関数でインポートのサーバントを設定する。

```
inport = self.getLocalInPort(profile)
(中略)
connector.setInPort(inport)
```

そしてOutPortPushConnectorオブジェクトのwrite関数内でインポートオブジェクトのwrite関数を呼び出せばデータを直接書き込める。

```
self._directInPort.write(data)
```

バッファがないためデータ書き込んだ際にInPortオブジェクトの変数_directInPortをTrueにして、読み込んだ時にFalseにしている。

isNew関数は_directInPortを返し、isEmpty関数は_directInPortの否定を返す。

Pull型
以下のクラスを実装した。

OutPortDirectProvider
アウトポートプロバイダクラス
OutPortProviderクラスを継承してある。
InterfaceTypeは「direct」に設定してある。

OutPortDirectConsumer
アウトポートコンシュマークラス
OutPortConsumerクラスを継承してある。

Push型と同じく、Init関数を呼び出すことで「direct」のInterfaceTypeが有効になる。

OutPortPullConnectorクラスに以下の関数を追加した。

setInPort関数:ダイレクトに接続する際にInPortオブジェクトにOutPortPullConnectorオブジェクトを設定する
read関数:データをダイレクトに読み込む

変数書き込みのためにwrite関数にコードを追加した。
変数宣言のためにコンストラクタにもコードを追加した。

InPortクラスに以下の関数を追加した。

addOutPortConnector:対象のOutPortPullConnectorオブジェクトをリストに追加する
removeOutPortConnector:対象のOutPortPullConnectorオブジェクトをリストから削除する

データ読み込みのためにread関数にコードを追加した。
変数宣言のためにコンストラクタにもコードを追加した。

まず、Push型と同じくOutPortBaseオブジェクトのcreateConnector関数内でInPortのサーバントをOutPortPullConnectorオブジェクトのsetInPort関数に渡す。
OutPortPullConnectorオブジェクトのsetInPort関数ではInPortオブジェクトのaddOutPortConnector関数を呼び出しリスト_outPortConnectorListにOutPortPullConnectorオブジェクトを追加している。
OutPortPullConnectorオブジェクトのwrite関数で変数にデータを書き込み、InPortオブジェクトのread関数でリストのサイズが1以上の時にリスト内のOutPortPullConnectorオブジェクトから変数を読み込む。
切断された時はOutPortPullConnectorオブジェクトのdisconnect関数内でInPortオブジェクトのremoveOutPortConnector関数を呼び出しリストから削除する。

テスト

添付したテスト用コードでテストを行った。
テスト用コードは以下の動作を行う。

setUp関数

マネージャ初期化
self.manager = OpenRTM_aist.Manager.init(sys.argv)
self.manager.activateManager()

データポート生成
self._d_in = RTC.TimedLong(RTC.Time(0,0),0)
self._inIn = OpenRTM_aist.InPort("in", self._d_in)
prop = OpenRTM_aist.Properties()
self._inIn.init(prop)
self.inport_obj = self._inIn.getPortRef()
→インポートとアウトポートを生成する

test_Push関数

isNew関数でデータが書き込まれたか判定できるかの確認
self._d_out.data = 100
self._outOut.write()
ret = self._inIn.isNew()
self.assertTrue(ret)

read関数でデータがダイレクトに読み込めるかの確認

data = self._inIn.read()
self.assertEqual(data.data, 100)
self.assertTrue(data is self._d_out)

test_Pull関数

データをダイレクトに読み込めるかの確認
data = self._inIn.read()
self.assertEqual(data.data, 100)
self.assertTrue(data is self._d_out)

補足

プロバイダ、コンシュマーのファクトリを登録した際にRTシステムエディタのコネクタの設定でInterfaceTypeが増殖する不具合が発生した。
調べてみたところOutPortBaseクラスとInPortBaseクラスのinitProviders関数とinitProviders関数に問題があったので修正した。

self.appendProperty("dataport.interface_type",
OpenRTM_aist.flatten(consumer_types))
→
for consumer_type in consumer_types:
self.appendProperty("dataport.interface_type",consumer_type)

self.appendProperty("dataport.interface_type",
OpenRTM_aist.flatten(provider_types))
→
for provider_type in provider_types:
self.appendProperty("dataport.interface_type",provider_type)

#3 - 2016/02/25 05:49 - miyamoto

- 進捗率 を 80 から 90 に変更

Pull型の実装を一部変更した。

OutPortPullConnectorのread関数でコールバックを呼び出すように変更した。

#4 - 2016/03/17 11:01 - miyamoto

- 進捗率 を 90 から 100 に変更

#5 - 2017/08/30 14:19 - n-ando

- ステータス を 新規 から 終了 に変更

ファイル

test_DirectPort.py	2.05 KB	2016/01/14	miyamoto