

## コア - バグ #3697

### ECにアタッチされたRTCが即座に追加されないためACTIVATE時に適切に遷移しない問題

2016/10/31 16:06 - n-ando

<b>ステータス:</b>	担当	<b>開始日:</b>	2016/10/31
<b>優先度:</b>	通常	<b>期日:</b>	
<b>担当者:</b>	n-ando	<b>進捗率:</b>	90%
<b>カテゴリ:</b>		<b>予定工数:</b>	0.00時間
<b>対象バージョン:</b>			
<b>説明</b>			
金広です。			
ありがとうございます。 動くようになってきました。			
ただまだ問題があります。 rtc1,rtc2,rtc3と3つのコンポーネントを作って、rtc1の実行コンテキストを使ってrtc2,rtc3も直列実行するように設定すると、 <a href="https://github.com/fkanehiro/hrpsys-base/blob/master/python/rtm.py#L454">https://github.com/fkanehiro/hrpsys-base/blob/master/python/rtm.py#L454</a>			
rtc1,rtc2,rtc3という順序でactivateすれば問題ないのですが、rtc3,rtc2,rtc1という順序でactivateすると、rtc3とrtc2はACTIVE_STATEに遷移しません。 この辺りも何か作法に変更あったのでしょうか。			
ロボットソフトウェアプラットフォーム研究チームの宮本と申します。			
実行コンテキストで複数のコンポーネントを実行する際にactivateする順序によって一部のコンポーネントがACTIVE_STATEに遷移しない問題ですが、原因を特定できたのでご報告いたします。			
まず実行コンテキストにアタッチしたコンポーネントはExecutionContextWorkerクラスのm_compsという配列に追加されるのですが、ExecutionContextインターフェースのadd_componentで実行コンテキストにコンポーネントを追加しようとしても即座にはm_compsにコンポーネントは追加されません。 この時点では別の配列に格納しておいて、updateComponentListという関数が呼び出された時点でm_compsに追加されます。			
コンポーネントの処理実行中にコンポーネントの追加をしないようにするために、このようになっています。			
> rtc1,rtc2,rtc3という順序でactivateすれば問題ないのですが、rtc3,rtc2,rtc1と > いう順序でactivateすると、rtc3とrtc2はACTIVE_STATEに遷移しません。			
updateComponentList関数はinvokeWorkerDoという関数で呼び出されていて、invokeWorkerDo関数はPeriodicExecutionContextのsvc関数で呼び出されています。 ただ全てのコンポーネントがINACTIVE_STATEの場合、以下の箇所処理が止められてしまいinvokeWorkerDo関数が実行されずupdateComponentList関数も実行されません。			
<pre>int PeriodicExecutionContext::svc(void) {     (省略)      Guard guard(m_workerthread.mutex_);     while (!m_workerthread.running_)     {         m_workerthread.cond_.wait();         //ここで止められているため下のinvokeWorkerDo関数は実行されず、updateComponentList関数も実行されない         //既にアタッチしたコンポーネントをアクティブにするとsignal関数が呼び出されて下の処理が実行される     }      coil::TimeValue t0(coil::gettimeofday());     ExecutionContextBase::invokeWorkerDo();     ExecutionContextBase::invokeWorkerPostDo();     coil::TimeValue t1(coil::gettimeofday());</pre>			

既にアタッチしてあるコンポーネントをactivateした場合、invokeWorkerDo関数が呼び出されてupdateComponentList関数も呼び出されるのでコンポーネントが追加されます。

rtc1をrtc2、rtc3より先にactivateした場合にはm\_compsにrtc2、rtc3が追加されているのでACTIVE\_STATEに遷移しますが、rtc2、rtc3を先にactivateした場合はこの時点でm\_compsにrtc2、rtc3が追加されておらずコンポーネントを見つけられずに失敗します。

全てのコンポーネントがINACTIVE\_STATEの時にコンポーネントを追加しても問題はないので、今後修正する予定です。

よろしくお願いします。

## 関係しているリビジョン

リビジョン 2795 - 2016/11/01 02:13 - n-ando

[incompat,bugfix,EC] EC activation bug for multiple RTCs when they activated from participants RTCs. refs #3697

リビジョン 2795 - 2016/11/01 02:13 - n-ando

[incompat,bugfix,EC] EC activation bug for multiple RTCs when they activated from participants RTCs. refs #3697

リビジョン 2796 - 2016/11/10 10:39 - kawauchi

[incompat,bugfix,EC] Return code was added. refs #3697

リビジョン 2796 - 2016/11/10 10:39 - kawauchi

[incompat,bugfix,EC] Return code was added. refs #3697

リビジョン 3097 - 2017/12/10 08:48 - n-ando

merged changes from trunk/OpenRTM-aist r2795-2805 during 2016-11

[incompat,bugfix,EC,->RELENG\_1\_2] EC activation bug for multiple RTCs when they activated from participants RTCs. refs #3697

[incompat,bugfix,EC,->RELENG\_1\_2] Return code was added. refs #3697

[incompat,bugfix,EC,->RELENG\_1\_2]Modified for ExtTrigExecutionContext. #3697

[compat,bugfix,->RELENG\_1\_2]refs #3703

[compat,bugfix,->RELENG\_1\_2] Initialization of connector\_id was corrected. refs #3708

[compat,->RELENG\_1\_2] ExtTrigger sample has been added in windows source package. refs #3709

[compat,1.2,->RELENG\_1\_2] Version number updated.

[incompat,->RELENG\_1\_2] CPU affinity setting has been added. refs #3711

[incompat,->RELENG\_1\_2] CPU affinity setting for ExecutionContext (only PeriodicEC) has been added. refs #3711

[incompat,->RELENG\_1\_2] CPU affinity setting for ExecutionContext (only PeriodicEC) has been added. refs #3711

リビジョン 3097 - 2017/12/10 08:48 - n-ando

merged changes from trunk/OpenRTM-aist r2795-2805 during 2016-11

[incompat,bugfix,EC,->RELENG\_1\_2] EC activation bug for multiple RTCs when they activated from participants RTCs. refs #3697

[incompat,bugfix,EC,->RELENG\_1\_2] Return code was added. refs #3697

[incompat,bugfix,EC,->RELENG\_1\_2]Modified for ExtTrigExecutionContext. #3697

[compat,bugfix,->RELENG\_1\_2]refs #3703

[compat,bugfix,->RELENG\_1\_2] Initialization of connector\_id was corrected. refs #3708

[compat,->RELENG\_1\_2] ExtTrigger sample has been added in windows source package. refs #3709

[compat,1.2,->RELENG\_1\_2] Version number updated.

[incompat,->RELENG\_1\_2] CPU affinity setting has been added. refs #3711

[incompat,->RELENG\_1\_2] CPU affinity setting for ExecutionContext (only PeriodicEC) has been added. refs #3711

[incompat,->RELENG\_1\_2] CPU affinity setting for ExecutionContext (only PeriodicEC) has been added. refs #3711

## 履歴

#1 - 2016/11/01 02:05 - n-ando

- 進捗率を0から50に変更

## 構造

ExecutionContextWorker-<>ExecutionContextBase<|---PeriodicExecutionContext

## 原因

- PeriodicExecutionContext では参加しているすべてのRTCがINACTIVEの時スレッドを止める。
- ExecutionContextWorker::addComponent/removeComponent  
ではスレッドが動いているときには、コンポーネントの参照を保持するリストを更新せず、1周期回り切ったときにリストの更新を行う。
- リストの更新を行う関数 ExecutionContextWorker::updateComponentList()  
は1)isRunning()==falseの時(ECのstateがstoppingの時)、またはスレッドが1周期回り切ったときのみ
- スレッドが止まっているため、後から追加されたコンポーネントはリストに追加されず、RTCにアタッチもされないためACTIVEにならない。

## 解決策

- ExecutionContextBase::onAdded(Removed)Component() 等のテンプレート関数を利用
- これらの関数にaddComponent/removeComponent時に実行したい内容を追加することが可能
- PeriodicExecutionContext のスレッドが止まっているときには m\_worker.updateComponentList() を呼ぶようにする
- updateComponentList() はprotected なので、そのままではPeriodicECでは呼べない
- mutexをモジュール化することはできないので、あきらめてpublicに変更

#2 - 2016/11/01 02:09 - n-ando

- 進捗率を 50 から 90 に変更

## テスト

examples/Composite のSensorCompを利用して、初期化関数を以下のように変更

- ControllerとMotorはrtc.confで.soをロードしておく。
- 以下のように、SensorのECを取り出し、Controller, Motor にアタッチ
- Motor, Controller, Sensor の順で activate した。

```
void MyModuleInit(RTC::Manager* manager)
{
+   std::cout << "1" << std::endl;
+   SensorInit(manager);
-   RTC::RtcBase* comp;
+   std::cout << "2" << std::endl;
+   RTC::RtcBase* scomp;
+   std::cout << "3" << std::endl;
+   RTC::RtcBase* ccomp;
+   std::cout << "4" << std::endl;
+   RTC::RtcBase* mcomp;
+   std::cout << "5" << std::endl;

+   // Create a component
-   comp = manager->createComponent("Sensor");
+   scomp = manager->createComponent("Sensor");
+   std::cout << "6" << std::endl;
+   RTC::RtObject_ptr srtc = scomp->getObjRef();
+   std::cout << "7" << std::endl;

+   ccomp = manager->createComponent("Controller");
+   std::cout << "8" << std::endl;
+   RTC::RtObject_ptr crtc = ccomp->getObjRef();
+   std::cout << "9" << std::endl;

+   mcomp = manager->createComponent("Motor");
+   std::cout << "10" << std::endl;
+   RTC::RtObject_ptr mrtc = mcomp->getObjRef();
+   std::cout << "11" << std::endl;
+
+   RTC::ExecutionContextList_var ecs = srtc->get_owned_contexts();
+   std::cout << "12" << std::endl;
+   ecs[0]->add_component(crtc);
+   std::cout << "13" << std::endl;
+   ecs[0]->add_component(mrtc);
+   std::cout << "14" << std::endl;
+
+   ecs[0]->activate_component(mrtc);
+   std::cout << "15" << std::endl;
+   ecs[0]->activate_component(crtc);
+   std::cout << "16" << std::endl;
+   ecs[0]->activate_component(srtc);
+   std::cout << "17" << std::endl;
+}
```

## 結果

変更前

```
n-ando@ubuntu1404:~/work/OpenRTM-aist/ec_bug/trunk/OpenRTM-aist/examples/Composite$ ./SensorComp
1
2
3
```

```
4
5
6
7
8
9
10
11
12
13
14
15
16
Sensor Activated
17
```

変更後

```
^Cn-ando@ubuntu1404:~/work/OpenRTM-aist/ec_bug/trunk/OpenRTM-aist/examples/Compote$ ./SensorComp
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Motor Activated
15
Controller Activated
16
Sensor Activated
17
^Cn-ando@ubuntu1404:~/work/OpenRTM-aist/ec_bug/trunk/OpenRTM-aist/examples/Compote$
```

Motor, Controller, Sensor の順でも activate できている。