Compliant Control Architecture - Bug #2220
'wrong order' of port configurations cause component crashes

04/07/2015 04:22 PM - C. Emmerich

Status:	Resolved	**Start date:**	04/07/2015
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

When configuring a component's ports with a wrong order (which is first the input ports and then the output ports) the component crashes with a segmentation fault when the input ports are already feeded during that configuration process, e.g.

```
Program received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0x7fffe6ffd700 (LWP 13601)]
rsb::InformerBase::uncheckedPublish (this=0x0, data=..., type=...)
   at /home/christian/citk/jenkins/jobs/rsb-cpp-master-toolkit-flexirob-nightly/workspace/src/rsb/Informer.cpp:92
92         EventPtr event = createEvent();
(gdb) bt
#0  rsb::InformerBase::uncheckedPublish (this=0x0, data=..., type=...)
   at /home/christian/citk/jenkins/jobs/rsb-cpp-master-toolkit-flexirob-nightly/workspace/src/rsb/Informer.cpp:92
#1  0x00007ffff6fc95e4 in uncheckedPublish<rci::Pose> (type=..., data=..., this=<optimized out>)
   at /vol/flexirob/releases/nightly/include/rsb0.12/rsb/Informer.h:159
#2  cca::OutputPort<rci::Pose>::publish (this=<optimized out>, data=...) at
/vol/flexirob/releases/nightly/include/cca0.5/cca/port/OutputPort.h:104
#3  0x00007ffff6fdd38a in flexirob::RobotCartesianPoses::onProcess (this=0x65c6c0)
   at /home/christian/citk/jenkins/jobs/cca-lwr-trunk-toolkit-flexirob-nightly/workspace/src/cca/lwr/RobotCartesianPoses.cpp:75
#4  0x00007ffff771c532 in cca::Node::process (this=0x65c6c0)
   at /home/christian/citk/jenkins/jobs/cca-trunk-toolkit-flexirob-nightly/workspace/src/cca/Node.cpp:367
#5  0x00007ffff77204b6 in cca::Node::inputCallback (this=0x65c6c0, event=..., name=...)
   at /home/christian/citk/jenkins/jobs/cca-trunk-toolkit-flexirob-nightly/workspace/src/cca/Node.cpp:390
#6  0x00007ffff7721fd1 in operator() (a2=..., a1=..., p=<optimized out>, this=<optimized out>) at
/usr/include/boost/bind/mem_fn_template.hpp:280
#7  operator()<boost::_mfi::mf2<void, cca::Node, boost::shared_ptr<rsb::Event>, const std::basic_string<char>&>,
boost::_bi::list1<boost::shared_ptr<rsb::Event>&> > (a=<synthetic pointer>, f=..., this=<optimized out>) at
/usr/include/boost/bind/bind.hpp:392
#8  operator()<boost::shared_ptr<rsb::Event> > (a1=..., this=<optimized out>) at /usr/include/boost/bind/bind_template.hpp:32
#9  boost::detail::function::void_function_obj_invoker1<boost::_bi::bind_t<void, boost::_mfi::mf2<void, cca::Node,
boost::shared_ptr<rsb::Event>, std::string const&>, boost::_bi::list3<boost::_bi::value<cca::Node*>, boost::arg<1>,
boost::_bi::value<std::string> > >, void, boost::shared_ptr<rsb::Event> >::invoke (function_obj_ptr=..., a0=...) at
/usr/include/boost/function/function_template.hpp:153
#10 0x00007ffff5ec8078 in operator() (a0=..., this=<optimized out>) at /usr/include/boost/function/function_template.hpp:767
#11 rsb::EventFunctionHandler::handle (this=<optimized out>, event=...)
   at /home/christian/citk/jenkins/jobs/rsb-cpp-master-toolkit-flexirob-nightly/workspace/src/rsb/Handler.cpp:71
#12 0x00007ffff773e5ba in cca::InputPortBase::InputHandler::handle (this=0x65dba0, event=...)
   at /home/christian/citk/jenkins/jobs/cca-trunk-toolkit-flexirob-nightly/workspace/src/cca/port/InputPort.h:138
#13 0x00007ffff5efa974 in rsb::eventprocessing::ParallelEventReceivingStrategy::deliver (this=0x699360, handler=..., e=...)
   at
/home/christian/citk/jenkins/jobs/rsb-cpp-master-toolkit-flexirob-nightly/workspace/src/rsb/eventprocessing/ParallelEventReceivingStrategy.
ingStrategy.cpp:159
#14 0x00007ffff5efcc06 in operator() (a2=..., a1=..., p=<optimized out>, this=<optimized out>) at
/usr/include/boost/bind/mem_fn_template.hpp:280
```

```
#15 operator()<boost::_mfi::mf2<void, rsb::eventprocessing::ParallelEventReceivingStrategy, boost::shared_ptr<rsb::Handler>,
boost::shared_ptr<rsb::Event> >, boost::_bi::list2<boost::shared_ptr<rsb::Handler>&, const boost::shared_ptr<rsb::Event>&> >
(a=<synthetic pointer>, f=..., this=<optimized out>)
    at /usr/include/boost/bind/bind.hpp:392
#16 operator()<boost::shared_ptr<rsb::Handler>, boost::shared_ptr<rsb::Event> > (a2=..., a1=..., this=<optimized out>)
    at /usr/include/boost/bind/bind_template.hpp:89
#17 boost::detail::function::void_function_obj_invoker2<boost::_bi::bind_t<void, boost::_mfi::mf2<void,
rsb::eventprocessing::ParallelEventReceivingStrategy, boost::shared_ptr<rsb::Handler>, boost::shared_ptr<rsb::Event> >,
boost::_bi::list3<boost::_bi::value<rsb::eventprocessing::ParallelEventReceivingStrategy*>, boost::arg<1>, boost::arg<2> > >,
void, boost::shared_ptr<rsb::Handler>&, boost::shared_ptr<rsb::Event> const&>::invoke (
    function_obj_ptr=..., a0=..., a1=...) at /usr/include/boost/function/function_template.hpp:153
#18 0x00007ffff5efd95a in operator() (a1=..., a0=..., this=<optimized out>) at
/usr/include/boost/function/function_template.hpp:767
#19 rsc::threading::OrderedQueueDispatcherPool<boost::shared_ptr<rsb::Event>,
rsb::Handler>::DeliverFunctionAdapter::deliver (this=<optimized out>,
    receiver=..., message=...) at /vol/flexirob/releases/nightly/include/rsc0.12/rsc/threading/OrderedQueueDispatcherPool.h:183
#20 0x00007ffff5f01a4c in rsc::threading::OrderedQueueDispatcherPool<boost::shared_ptr<rsb::Event>, rsb::Handler>::worker
(this=0x699378,
    workerNum=<optimized out>) at
/vol/flexirob/releases/nightly/include/rsc0.12/rsc/threading/OrderedQueueDispatcherPool.h:337
#21 0x00007ffff5efd8a0 in operator() (this=<optimized out>) at /usr/include/boost/function/function_template.hpp:767
---Type <return> to continue, or q <return> to quit---
#22 boost::detail::thread_data<boost::function<void ()> >::run() (this=<optimized out>) at
/usr/include/boost/thread/detail/thread.hpp:117
#23 0x00007ffff4b50a4a in ?? () from /usr/lib/x86_64-linux-gnu/libboost_thread.so.1.54.0
#24 0x00007ffff647c182 in start_thread (arg=0x7fffe6ffd700) at pthread_create.c:312
#25 0x00007ffff678c47d in clone () at ../sysdeps/unix/sysv/linux/x86_64/clone.S:111
```

The cause for this seems to be that the component port already recieves and processes data as well as finally wants to publish data while the output ports might not yet be completely constructed/configured.

Associated revisions

Revision 784 - 04/08/2015 11:22 AM - anordman

Fixed order of port configuration

- Now splitted into base port conf and port-specific conf
- Port is considered cofigured not before both configurations are finished

fixes #2220

Revision 786 - 04/08/2015 11:33 AM - anordman

Fixed component print

- Missing include to Tick (due to fixed beat input port)

refs #2220

History

#1 - 04/07/2015 05:08 PM - Anonymous

Due to source:trunk/cca/src/cca/port/OutputPort.h#L100, line OutputPort.h:104 (see backtrace) should never be reached when the output port is unconfigured. Looking at source:trunk/cca/src/cca/port/OutputPort.h#L146 and source:trunk/cca/src/cca/port/OutputPort.h#L157 though, the order of commands is wrong, so that the port is considered configured already before the informer is actually constructed. This is a bug!

#2 - 04/08/2015 11:22 AM - Anonymous

- Status changed from New to Resolved

- % Done changed from 0 to 100

Applied in changeset cca|r784.