RTM_URG Reference Manual

Generated by Doxygen 1.5.4

Mon Dec 3 09:59:42 2007

Contents

1 RTM_URG Main Page

RTM-URG

1.1 RTM-URG とは

RTM-URG とは、URG-04LX, Top-URG からのデータを、RTM 上で利用するためのコンポーネントです。SCIP2.0 に準拠しており、URG, Top-URG に限らず SCIP2.0 規格のセンサからの測距データを CSV 形式で取得することができます。

1.2 入手

rtm_urg-0.0.1.tar.gz を SourceForge.jp からダウンロードできる。

http://sourceforge.jp/projects/beego/files/?release_id=28369#28369

1.3 コンパイル

コンパイルには、OpenRTM-aist (http://www.is.aist.go.jp/rt/OpenRTM-aist/), および SDL (http://www.libsdl.org) が必要となる。 取得パッケージを展開後、configure, make によりコンポーネントを生成できる。

```
% tar zxvf rtm_urg-0.0.1.tar.gz
% cd rtm_urg-0.0.1
% ./configure
% make
```

UrgDeviceComp が UrgDevice/ のディレクトリ以下に、UrgHandlerCsvComp が UrgHandlerCsv/ ディレクトリ以下に生成される。

- UrgDevice / UrgDeviceComp ... URG が接続されている側でデータ取得を行うコンポーネント
  - データ送信用のポートを１つ持つ
  - コンポーネント初期化時に URG への接続を行い、アクティブ時にデータを送信する
- UrgHandlerCsv / UrgHandlerCsvComp ... URG データの受信、利用を行うコンポーネント
  - データ受信用のポートを１つ持つ
  - UrgDeviceComp の動作を確認するためのコンポーネント

1.4 使用例

ConnectorComp::cpp (p. ??) を用い、UrgDeviceComp と UrgHandlerCsvComp を接続して、受信データを UrgHandlerCsvComp 側のターミナルで確認方法を示す。

ConnectorComp::cpp (p. ??) のコンパイル

```
% g++ `rtm-config --cflags` ConnectorComp.cpp `rtm-config --libs`
```

スクリプト **run::sh** (p. ??) を実行すると、UrgHandlerCsv 側のターミナルから、URG からの CSV データが出力されているのがわかる。CSV のデータ形式は、

```
(以降の CSV データ総数),(スキャンライン 0 番目の測距データ),(スキャンライン 1 番目の測距データ), ...
```

となる。 スクリプトの実行方法は、以下の通り。

```
% /bin/sh run.sh
```

詳細は、OpenRTM-aist のページ (http://www.is.aist.go.jp/rt/OpenRTM-aist/) を参照のこと。

1.5 追加実装する予定の機能、今後の予定

- 追加実装する予定の機能
  - SCIP2.0 コマンドを受信し、その応答を返すコンポーネントの作成
  - URG のシリアル ID を指定して接続を行う仕組みを備える
- 今後の予定 現在は、CORBA, および RTM コンポーネントへの理解不足のため、１本の入出力を使うコンポーネントしか作成できなかった。

  CORBA および RTM コンポーネントの理解を行ったのち、センサデータ受信側のコンポーネントから、URG センサを制御するコンポーネントに対して、受信指令を出せるように変更したい。このように変更することで、URG センサの性能を十分に生かしたデータ測距を行うことができる。

  また、出力データのフォーマットは、取得データ情報を含めるように改善を行う予定である。

以上。

2 RTM_URG Page Index

2.1 RTM_URG Related Pages

Here is a list of all related documentation pages:

3 RTM_URG Page Documentation

```
// -*- C++ -*-
#include <iostream>
#include <vector>
#include <string>
#include <rtm/CorbaNaming.h>
#include <rtm/RTObject.h>
#include <rtm/NVUtil.h>
#include <rtm/CORBA_SeqUtil.h>
#include <rtm/CorbaConsumer.h>
#include <assert.h>


using namespace RTC;

void usage()
{
  std::cout << std::endl;
  std::cout << "usage: ConnectorComp [options]" << std::endl;
  std::cout << std::endl;
  std::cout << "  --flush         ";
  std::cout << ": Set subscription type Flush" << std::endl;
  std::cout << "  --new           ";
  std::cout << ": Set subscription type New" << std::endl;
  std::cout << "  --periodic [Hz] ";
  std::cout << ": Set subscription type Periodic" << std::endl;
  std::cout << std::endl;
}

int main (int argc, char** argv)
{
  int _argc(0);
  char** _argv(0);

  std::string subs_type;
  std::string period;
  for (int i = 1; i < argc; ++i)
    {
      std::string arg(argv[i]);
      if (arg == "--flush")         subs_type = "Flush";
      else if (arg == "--new")      subs_type = "New";
      else if (arg == "--periodic")
        {
          subs_type = "Periodic";
          if (++i < argc) period = argv[i];
          else            period = "1.0";
        }
      else if (arg == "--help")
        {
          usage();
          exit(1);
        }
      else
        {
          subs_type = "Flush";
        }
    }

  std::cout << "Subscription Type: " << subs_type << std::endl;
  if (period != "")
    std::cout << "Period: " << period << " [Hz]" << std::endl;


  CORBA::ORB_var orb = CORBA::ORB_init(_argc, _argv);
  CorbaNaming naming(orb, "localhost:9876");
```

```
  CorbaConsumer<RTObject> conin, conout;
  PortList_var pin;
  PortList_var pout;

  // find ConsoleIn0 component
  conin.setObject(naming.resolve("UrgDevice0.rtc"));

  // get ports
  pin = conin->get_ports();
  pin[0]->disconnect_all();
  assert(pin->length() > 0);
  // activate ConsoleIn0
  ExecutionContextServiceList_var eclisti;
  eclisti = conin->get_execution_context_services();
  eclisti[0]->activate_component(RTObject::_duplicate(conin._ptr()));

  // find ConsoleOut0 component
  conout.setObject(naming.resolve("UrgHandlerCsv0.rtc"));
  // get ports
  pout = conout->get_ports();
  pout[0]->disconnect_all();
  assert(pout->length() > 0);
  // activate ConsoleOut0
  ExecutionContextServiceList_var eclisto;
  eclisto = conout->get_execution_context_services();
  eclisto[0]->activate_component(RTObject::_duplicate(conout._ptr()));

  // connect ports
  ConnectorProfile prof;
  prof.connector_id = "";
  prof.name = CORBA::string_dup("connector0");
  prof.ports.length(2);
  prof.ports[0] = pin[0];
  prof.ports[1] = pout[0];
  CORBA_SeqUtil::push_back(prof.properties,
                           NVUtil::newNV("dataport.interface_type",
                                         "CORBA_Any"));
  CORBA_SeqUtil::push_back(prof.properties,
                           NVUtil::newNV("dataport.dataflow_type",
                                         "Push"));
  CORBA_SeqUtil::push_back(prof.properties,
                           NVUtil::newNV("dataport.subscription_type",
                                         subs_type.c_str()));
  if (period != "")
  CORBA_SeqUtil::push_back(prof.properties,
                           NVUtil::newNV("dataport.push_interval",
                                         period.c_str()));
  ReturnCode_t ret;
  ret = pin[0]->connect(prof);
  assert(ret == RTC::RTC_OK);

  std::cout << "Connector ID: " << prof.connector_id << std::endl;
  NVUtil::dump(prof.properties);

  orb->destroy();
  exit(1);
}


#!/bin/sh
#
# @file run.sh
# @brief SimpleIO example startup script
# @date $Date: 2007/04/27 06:12:58 $
#
# Copyright (c) 2003-2007 Noriaki Ando <n-ando@aist.go.jp>
#          Task-intelligence Research Group,
#          Intelligent System Research Institute,
```

```
#          National Institute of Industrial Science (AIST), Japan
#          All rights reserved.
#
# Edit
#          Satofumi KAMIMURA
#
# $Id: run.sh,v 1.4 2007/04/27 06:12:58 n-ando Exp $
#

nsport='9876'
hostname=`hostname`


term=`which kterm`

if test "x$term" = "x" ; then
    term=`which xterm`
fi

if test "x$term" = "x" ; then
    term=`which uxterm`
fi

if test "x$term" = "x" ; then
    term=`which gnome-terminal`
fi

if test "x$term" = "x" ; then
    echo "No terminal program (kterm/xterm/gnome-terminal) exists."
    exit
fi

rtm-naming $nsport

echo 'corba.nameservers: '$hostname':'$nsport > ./rtc.conf
echo 'naming.formats: %n.rtc' >> ./rtc.conf
echo 'logger.log_level: TRACE' >> ./rtc.conf

$term -e ./UrgDeviceComp &
sleep 1
$term -e ./UrgHandlerCsvComp &

#sleep 3
#./ConnectorComp &
gdb ./ConnectorComp

#sleep 10
#$nspid=`ps -ax | grep 9876 | awk '{print $1}'`
#kill $nspid
#echo 'Naming service was stopped.'
```