



# YARP User Manual

Paul Fitzpatrick

Giorgio Metta

Lorenzo Natale

online at <http://yarp0.sourceforge.net>

19 Nov 2008



# Contents

<b>1</b>	<b>bottle/main.cpp</b>	<b>3</b>
<b>2</b>	<b>carrier/carrier_stub.cpp</b>	<b>7</b>
<b>3</b>	<b>cuda/cuda_gpu.cpp</b>	<b>11</b>
<b>4</b>	<b>dev/fake_motor.cpp</b>	<b>15</b>
<b>5</b>	<b>dev/file_grabber.cpp</b>	<b>21</b>
<b>6</b>	<b>dev/grabber_client.cpp</b>	<b>27</b>
<b>7</b>	<b>dev/motortest.cpp</b>	<b>31</b>
<b>8</b>	<b>external/nameclient.cpp</b>	<b>35</b>
<b>9</b>	<b>framerate/main.cpp</b>	<b>39</b>
<b>10</b>	<b>imagemagick/main.cpp</b>	<b>43</b>
<b>11</b>	<b>opencv/main.cpp</b>	<b>47</b>
<b>12</b>	<b>os/bottle_add.cpp</b>	<b>51</b>
<b>13</b>	<b>os/browse_bottle.cpp</b>	<b>53</b>
<b>14</b>	<b>os/buffered_port.cpp</b>	<b>57</b>

---

15 os/database.cpp	61
16 os/image_process.cpp	65
17 os/image_process_module.cpp	69
18 os/image_source.cpp	73
19 os/make_count.cpp	75
20 os/portable_pair.cpp	77
21 os/ratethread.cpp	81
22 os/simple_receiver.cpp	85
23 os/simple_sender.cpp	87
24 os/summer.cpp	89
25 os/threads.cpp	91
26 os/view_count.cpp	95
27 port_power/ex0000_receiver.cpp	97
28 port_power/ex0001_sender.cpp	99
29 port_power/ex0002_connector.cpp	101
30 port_power/ex0100_receiver.cpp	103
31 port_power/ex0101_sender.cpp	105
32 port_power/ex0200_polling.cpp	107
33 port_power/ex0300_port_callback.cpp	109
34 port_power/ex0301_buffered_callback.cpp	111

---

35	<a href="#">port_power/ex0302_buffered_ext_callback.cpp</a>	113
36	<a href="#">port_power/ex0400_expect_reply.cpp</a>	115
37	<a href="#">port_power/ex0401_give_reply.cpp</a>	119
38	<a href="#">port_power/ex0402_port_callback_reply.cpp</a>	121
39	<a href="#">port_power/ex0403_bufferedport_callback_reply.cpp</a>	125
40	<a href="#">port_power/ex0500_raw_target_receiver.cpp</a>	129
41	<a href="#">port_power/ex0501_raw_target_sender.cpp</a>	131
42	<a href="#">port_power/ex0502_raw_target_connector.cpp</a>	133
43	<a href="#">port_power/ex0503_serial_target_receiver.cpp</a>	135
44	<a href="#">port_power/ex0504_serial_target_sender.cpp</a>	137
45	<a href="#">port_power/ex0505_compliant_target_receiver.cpp</a>	139
46	<a href="#">port_power/ex0506_compliant_target_sender.cpp</a>	141
47	<a href="#">port_power/ex0507_alternative_compliant_target_receiver.cpp</a>	143
48	<a href="#">port_power/ex0508_alternative_compliant_target_sender.cpp</a>	145
49	<a href="#">port_power/TargetVer1.h</a>	147
50	<a href="#">port_power/TargetVer1b.h</a>	149
51	<a href="#">port_power/TargetVer2.h</a>	151
52	<a href="#">port_power/TargetVer3.h</a>	153
53	<a href="#">property/main.cpp</a>	155







# Chapter 1

## bottle/main.cpp

Experiment with bottles and properties.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <stdlib.h>

#include <yarp/os/Bottle.h>
#include <yarp/os/Property.h>

using namespace yarp::os;

int main(int argc, char *argv[]) {

    Bottle b;
    b.addString("color");
    b.addString("red");
    printf("Bottle b is: %s\n", b.toString().c_str());
    // should give: color red

    Bottle b2;
    b2.addString("height");
    b2.addInt(15);
    printf("Bottle b2 is: %s\n", b2.toString().c_str());
    // should give: height 15

    Bottle b3;
    b3.addList() = b;
    b3.addList() = b2;
    printf("Bottle b3 is: %s\n", b3.toString().c_str());
    // should give: (color red) (height 15)

    printf("color check: %s\n", b3.find("color").asString().c_str());
    printf("height check: %d\n", b3.find("height").asInt());

    Bottle b4;
    b4.addString("nested");
    b4.addList() = b3;
```

```
printf("Bottle b4 is: %s\n", b4.toString().c_str());
// should give: nested ((color red) (height 15))

// alternative way to create a Bottle from textual representation
Bottle b5("(pos left top) (size 10)");
printf("Bottle b5 is: %s\n", b5.toString().c_str());
// should give: (pos left top) (size 10)

Bottle b6;
b6 = b5;
b6.addList() = b4;
printf("Bottle b6 is: %s\n", b6.toString().c_str());
// should give: (pos left top) (size 10) (nested ((color red) (height 5))

printf("size check: %d\n", b6.find("size").asInt());
printf("pos check: %s\n", b6.find("pos").asString().c_str());
// find assumes key->value pairs; for lists, use findGroup
printf("pos group check: %s\n", b6.findGroup("pos").toString().c_str());
// see documentation for Bottle::findGroup
printf("nested check: %s\n", b6.find("nested").toString().c_str());
printf("nested height check: %d\n", b6.find("nested").find("height").asInt());

printf("\n");
printf("Relationship of Bottle and Property\n");
Property subProp;
subProp.put("hello", "there");
subProp.put("fortytwo", 42);
printf("subProp: %s\n", subProp.toString().c_str());

Value *lst = Value::makeList();
if (lst==NULL) {
    printf("Failed to allocate list\n");
    return 1;
}
lst->asList()->fromString(subProp.toString());
printf("lst: %s\n", lst->toString().c_str());

Property prop;
prop.put("height", 15);
prop.put("verbose", 1);
prop.put("sub", lst);
printf("prop: %s\n", prop.toString().c_str());

return 0;
}
```





## Chapter 2

# carrier/carrier\_stub.cpp

Example showing how to add a new carrier to YARP.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>

#include <yarp/os/impl/Carrier.h>
#include <yarp/os/impl/Carriers.h>
#include <yarp/os/impl/TextCarrier.h>

using namespace yarp::os;
using namespace yarp::os::impl;

class TestCarrier : public TextCarrier {
public:
    virtual String getName() {
        return "test";
    }

    virtual String getSpecifierName() {
        return "TESTTEST";
    }

    virtual Carrier *create() {
        return new TestCarrier();
    }
};

int main(int argc, char *argv[]) {
    Network yarp;
    Carriers::addCarrierPrototype(new TestCarrier);

    BufferedPort<Bottle> out, in;
    out.open("/test/out");
    in.open("/test/in");

    Network::connect("/test/out", "/test/in", "test");
}
```

```
out.prepare().fromString("1 2 3");
out.write();

Bottle * bot = in.read();
if (bot!=NULL) {
    printf("Got message %s\n", bot->toString().c_str());
}

out.close();
in.close();

return 0;
}
```





## Chapter 3

# cuda/cuda\_gpu.cpp

Example showing how to use the CUDA driver to execute custom code onto nVidia graphics cards (for supported cards, try to search on Google something like "cuda supported gpu")

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-  
  
/*  
 * Copyright (C) 2007 Giacomo Spigler  
 * CopyPolicy: Released under the terms of the GNU GPL v2.0.  
 */  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <yarp/dev/GPUInterface.h>  
#include <yarp/dev/PolyDriver.h>  
  
#include <yarp/dev/Drivers.h>  
  
#include <yarp/sig/all.h>  
#include <yarp/os/all.h>  
  
#include <cv.h>  
#include <cvaux.h>  
#include <highgui.h>  
  
#include <cutil.h> //only needed for timing..  
  
using namespace yarp::os;  
using namespace yarp::sig;  
using namespace yarp::sig::file;  
using namespace yarp::dev;  
  
int main(int argc, char *argv[]) {  
    //Instantiate the GPU driver  
    ImageOf<PixelRgb> img;  
    yarp::sig::file::read(img, "../dev/image/img0250.ppm");  
}
```

```
ImageOf<PixelRgb> out=img;

// use YARP to create and configure an instance of CUDAGPU
Property config;
// no arguments, use a default
char str[80];
sprintf(str, "(device cuda) (w %d) (h %d) (bpp 3)", img.width(), img.height());
config.fromString(str);

PolyDriver dd(config);
if (!dd.isValid()) {
    printf("Failed to create and configure a device\n");
    exit(1);
}

IGPUDevice *gpu;
if (!dd.view(gpu)) {
    printf("Failed to view device through IGPUDevice interface\n");
    exit(1);
}

//Load and execute a program for the GPU
int prog;
if(argc!=2) {
    prog = gpu->load("progs/bgr.cubin");
} else {
    prog = gpu->load(argv[1]);
}
write(out, "test0.ppm");

unsigned int hTimer;
cutCreateTimer(&hTimer);
cutStartTimer(hTimer);

gpu->execute(prog, &img, &out);
write(out, "test1.ppm");

cutStopTimer(hTimer);
printf("Processing time: %fms\n", cutGetTimerValue(hTimer));

//Show the resulting image onto the screen using OpenCV
IplImage *cvImage = cvCreateImage(cvSize(out.width(), out.height()), IPL_DEPTH_8U, 3);
cvCvtColor((IplImage*)out.getIplImage(), cvImage, CV_RGB2BGR);

cvNamedWindow("CUDA", 1);
cvShowImage("CUDA", cvImage);

cvWaitKey(3000);
cvDestroyWindow("CUDA");

cvReleaseImage(&cvImage);
```

```
    dd.close();  
    return 0;  
}
```



# Chapter 4

## dev/fake\_motor.cpp

Some tips on how to create a device for a new motor control board.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-  
  
/*  
 * Copyright (C) 2006 Paul Fitzpatrick  
 * CopyPolicy: Released under the terms of the GNU GPL v2.0.  
 *  
 */  
  
#include <yarp/dev/ControlBoardInterfaces.h>  
#include <yarp/dev/Drivers.h>  
#include <yarp/dev/PolyDriver.h>  
  
#include <stdio.h>  
  
using namespace yarp::os;  
using namespace yarp::dev;  
  
class FakeMotor : public DeviceDriver, public IPositionControl {  
public:  
    virtual bool getAxes(int *ax) {  
        *ax = 2;  
        printf("FakeMotor reporting %d axes present\n", *ax);  
        return true;  
    }  
  
    virtual bool setPositionMode() {  
        return true;  
    }  
  
    virtual bool positionMove(int j, double ref) {  
        return true;  
    }  
  
    virtual bool positionMove(const double *refs) {
```

```
        return true;
    }

    virtual bool relativeMove(int j, double delta) {
        return true;
    }

    virtual bool relativeMove(const double *deltas) {
        return true;
    }

    virtual bool checkMotionDone(int j, bool *flag) {
        return true;
    }

    virtual bool checkMotionDone(bool *flag) {
        return true;
    }

    virtual bool setRefSpeed(int j, double sp) {
        return true;
    }

    virtual bool setRefSpeeds(const double *spds) {
        return true;
    }

    virtual bool setRefAcceleration(int j, double acc) {
        return true;
    }

    virtual bool setRefAccelerations(const double *accs) {
        return true;
    }

    virtual bool getRefSpeed(int j, double *ref) {
        return true;
    }

    virtual bool getRefSpeeds(double *spds) {
        return true;
    }

    virtual bool getRefAcceleration(int j, double *acc) {
        return true;
    }
}
```

```
virtual bool getRefAccelerations(double *accs) {
    return true;
}

virtual bool stop(int j) {
    return true;
}

virtual bool stop() {
    return true;
}

virtual bool open(Searchable& config) {
    return true;
}

virtual bool close() {
    return true;
}
};

void testMotor(PolyDriver& driver) {
    IPositionControl *pos;
    if (driver.view(pos)) {
        int ct = 0;
        pos->getAxes(&ct);
        printf(" number of axes is: %d\n", ct);
    } else {
        printf(" could not find IPositionControl interface\n");
    }
}

int main(int argc, char *argv[]) {
    Drivers::factory().add(new DriverCreatorOf<FakeMotor>("motor",
                                                         "controlboard",
                                                         "FakeMotor"));

    printf("=====\n");
    printf("check our device can be instantiated directly\n");

    PolyDriver direct("motor");
    if (direct.isValid()) {
        printf("Direct instantiation worked\n");
        testMotor(direct);
    } else {
        printf("Direct instantiation failed\n");
    }
    direct.close();

    // check our device can be wrapped in the controlboard network wrapper
    printf("\n\n");
}
```

```
printf("=====\n");
printf("check our device can be wrapped in controlboard\n");

PolyDriver indirect("(device controlboard) (subdevice motor)");
if (indirect.isValid()) {
    printf("Indirect instantiation worked\n");
} else {
    printf("Indirect instantiation failed\n");
}
indirect.close();

// check our device can be wrapped in the controlboard network wrapper
// and accessed remotely
printf("\n\n");
printf("=====\n");
printf("check our device can be accessed via remote_controlboard\n");

PolyDriver server("(device controlboard) (subdevice motor) (name /server)");
if (server.isValid()) {
    printf("Server instantiation worked\n");

    PolyDriver client("(device remote_controlboard) (local /client) (remote /server)");
    if (client.isValid()) {
        printf("Client instantiation worked\n");
        testMotor(client);
    } else {
        printf("Client instantiation failed\n");
    }
    client.close();
}
server.close();

return 0;
}
```





## Chapter 5

# dev/file\_grabber.cpp

This example shows how we would make a new device for a camera. We make a device that reads images from file. The only step missing here is to integrate the device into the YARP library.

Here's the header file, dev/FileFrameGrabber.h

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
#include <yarp/sig/all.h>

class FileFrameGrabber : public yarp::dev::IFrameGrabberImage,
                        public yarp::dev::DeviceDriver {
private:
    yarp::os::ConstString pattern, lastLoad;
    int first, last, at;
    int h, w;

    bool findImage(yarp::sig::ImageOf<yarp::sig::PixelRgb>& image) {
        bool triedFirst = false;
        char buf[1000];
        sprintf(buf, pattern.c_str(), at);
        while (!yarp::sig::file::read(image, buf)) {
            if (at==first) {
                if (triedFirst) {
                    return false;
                }
                triedFirst = true;
            }
            if (last==-1) {
                at = first;
            } else {
                at++;
                if (at>last) {
                    at = first;
                }
            }
        }
    }
};
```

```

        }
        sprintf(buf,pattern.c_str(),at);
    }
    lastLoad = buf;
    h = image.height();
    w = image.width();
    return true;
}

public:
    FileFrameGrabber() {
        pattern = "%d.ppm";
        first = last = -1;
        at = -1;
        h = w = 0;
    }

    bool open(const char *pattern, int first, int last) {
        this->pattern = pattern;
        this->first = first;
        this->last = last;
        at = first;
        yarp::sig::ImageOf<yarp::sig::PixelRgb> dummy;
        return findImage(dummy);
    }

    virtual bool open(yarp::os::Searchable& config) {
        yarp::os::ConstString pattern =
            config.check("pattern",yarp::os::Value("%d.ppm")).asString();
        int first = config.check("first",yarp::os::Value(0)).asInt();
        int last = config.check("last",yarp::os::Value(-1)).asInt();
        return open(pattern,first,last);
    }

    virtual bool close() {
        return true; // easy
    }

    virtual bool getImage(yarp::sig::ImageOf<yarp::sig::PixelRgb>& image) {
        bool ok = findImage(image);
        if (ok) {
            printf("showing image %s\n", lastLoad.c_str());
            at++;
            if (last!=-1 && at>last) {
                at = first;
            }
        }
        return ok;
    }

    virtual int height() const {
        return h;
    }

    virtual int width() const {
        return w;
    }
}

```

```
};
```

And here's how we can use it. In fact, we just grab the first image from the device.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <stdlib.h>
#include <yarp/dev/FrameGrabberInterfaces.h>
#include <yarp/dev/PolyDriver.h>
#include <yarp/dev/Drivers.h>
#include "FileFrameGrabber.h"
using namespace yarp::os;
using namespace yarp::sig;
using namespace yarp::dev;

int main(int argc, char *argv[]) {
    Network yarp;

    // give YARP a factory for creating instances of FileFrameGrabber
    DriverCreator *file_grabber_factory =
        new DriverCreatorOf<FileFrameGrabber>("file_grabber",
                                             "grabber",
                                             "FileFrameGrabber");
    Drivers::factory().add(file_grabber_factory); // hand factory over to YARP

    // use YARP to create and configure an instance of FileFrameGrabber
    Property config;
    if (argc==1) {
        // no arguments, use a default
        config.fromString("(device file_grabber) (pattern \"image/%03d.ppm\")");
    } else {
        // expect something like '--device file_grabber --pattern "image/%03d.ppm"'
        // or '--device dragonfly'
        // or '--device test_grabber --period 0.5 --mode [ball]'
        config.fromCommand(argc,argv);
    }
    PolyDriver dd(config);
    if (!dd.isValid()) {
        printf("Failed to create and configure a device\n");
        exit(1);
    }
    IFrameGrabberImage *grabberInterface;
    if (!dd.view(grabberInterface)) {
        printf("Failed to view device through IFrameGrabberImage interface\n");
        exit(1);
    }

    ImageOf<PixelRgb> img;
    grabberInterface->getImage(img);
    printf("Got a %dx%d image\n", img.width(), img.height());

    dd.close();

    return 0;
}
```

```
}
```





## Chapter 6

# dev/grabber\_client.cpp

How to grab images from a remote source using the `yarp::dev::IFrameGrabber` interface.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <stdlib.h>
#include <yarp/os/all.h>
#include <yarp/sig/all.h>
#include <yarp/dev/FrameGrabberInterfaces.h>
#include <yarp/dev/PolyDriver.h>
#include <yarp/dev/Drivers.h>
using namespace yarp::os;
using namespace yarp::sig;
using namespace yarp::dev;

/*
 * Read an image from a remote source using the "device" view of
 * camera-like sources.
 *
 * Remote source could be, for example:
 *   yarpdev --device test_grabber --name /fakey
 *
 */

int main() {
    Network yarp;

    Property config;
    config.put("device","remote_grabber"); // device type
    config.put("local","/client");        // name of local port to use
    config.put("remote","/fakey");        // name of remote port to connect to

    PolyDriver dd(config);
    if (!dd.isValid()) {
        printf("Failed to create and configure device\n");
        exit(1);
    }
}
```

```
    }
    IFrameGrabberImage *grabberInterface;
    if (!dd.view(grabberInterface)) {
        printf("Failed to view device through IFrameGrabberImage interface\n");
        exit(1);
    }

    ImageOf<PixelRgb> img;
    grabberInterface->getImage(img);
    printf("Got a %dx%d image\n", img.width(), img.height());

    dd.close();

    return 0;
}
```





# Chapter 7

## dev/motortest.cpp

Example showing how to use the SerialServoBoard driver AND how to configure and use the IPositionControl interface./\*\*

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <yarp/os/Network.h>
#include <yarp/os/Port.h>
#include <yarp/os/Bottle.h>
#include <yarp/os/Time.h>
#include <stdio.h>
#include <stdlib.h>

#include <yarp/dev/PolyDriver.h>

#include <yarp/dev/ControlBoardInterfaces.h>

using namespace yarp::os;
using namespace yarp::dev;

int main(int argc, char *argv[]) {
    Network yarp;

    if(argc!=7) {
        printf(" Error: correct usage is %s --board BOARD --comport COM --baudrate BAUDRATE\n", argv[0]);
        printf("          BOARD is one out of ssc32, minissc, pontech_sv203x, mondotronic_smi, pololu_usb_16servo, picop");
        printf("          COM is COMx or /dev/ttyS0\n");
        printf("          BAUDRATE is the baud rate, eg: 38400");

        exit(1);
    }

    Property config;
    config.fromCommand(argc, argv);
```

```
Property options;
//options.put("robot", "icub"); // typically from the command line.
options.put("device", "SerialServoBoard");
options.put("board", config.check("board", yarp::os::Value("ssc32")).asString().c_str());
options.put("comport", config.check("comport", yarp::os::Value("/dev/ttyS0")).asString().c_str());
options.put("baudrate", config.check("baudrate", yarp::os::Value(38400)).asInt());

PolyDriver dd(options);
if(!dd.isValid()) {
    printf("Device not available.\n");
    Network::fini();
    return 1;
}

IPositionControl *pos;

dd.view(pos);

int jnts = 0;
pos->getAxes(&jnts);

//printf("axes: %d\n", jnts);

//Time::delay(1);

pos->positionMove(0, -45);

Time::delay(1);

dd.close();

return 0;
}
```





## Chapter 8

# external/nameclient.cpp

How to get data from YARP's name server without using any YARP code in your own program.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-  
  
// This program based on contributions from Paul Fitzpatrick, Jose Gaspar  
  
// If you are on Windows, make sure WIN32 is defined.  
// otherwise, we assume UNIX.  
  
// This simple demo program doesn't deal with fragmentation of tcp  
// reads/writes. If you're doing something serious, make sure you  
// deal with cases where socket reads/writes are only partially completed.  
// in one call.  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <sys/types.h>  
#include <string.h>  
  
#ifdef WIN32  
#include <winsock2.h>  
#else  
#include <unistd.h>  
#include <sys/socket.h>  
#include <netinet/in.h>  
#include <netdb.h>  
#endif  
  
// port number for yarp server  
#define PORT 10000  
  
// IP address or host name for yarp server  
#define HOST "127.0.0.1"  
  
// buffer size for responses from name server
```

```
#define BUFSIZE      1000

#ifdef WIN32
void windowsNetStart(void) {
    int    wsaRc;
    WSADATA wsaData;
    if(wsaRc=WSAStartup(0x0101, &wsaData)) {
        perror("WinSock init");
    }
    if(wsaData.wVersion != 0x0101) {
        WSACleanup();
        perror("wsaData.wVersion");
    }
}
#endif

int main(int argc, char *argv[]) {
    char buf[BUFSIZE];
    int i;
#ifdef WIN32
    SOCKET sd;
#else
    int sd;
#endif
    //struct sockaddr_in sin;
    struct sockaddr_in pin;
    struct hostent *hp;

    if (argc<=1) {
        printf("Please supply a message to send to the nameserver. ");
        printf(" Examples:\n");
        printf("  help\n");
        printf("  list\n");
        printf("  query /portname\n");
        exit(1);
    }

#ifdef WIN32
    // start windows networking
    windowsNetStart();
#endif

    // get host information
    if ((hp = gethostbyname(HOST)) == 0) {
        perror("gethostbyname");
        exit(1);
    }

    // set up socket structure
    memset(&pin, 0, sizeof(pin));
    pin.sin_family = AF_INET;
    pin.sin_addr.s_addr = ((struct in_addr *) (hp->h_addr))->s_addr;
    pin.sin_port = htons(PORT);

    // get a socket
    if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
```

```
        perror("socket");
        exit(1);
    }

    // connect to PORT on HOST
    if (connect(sd,(struct sockaddr *) &pin, sizeof(pin)) == -1) {
        perror("connect");
        exit(1);
    }

    // Commands sent to the yarp server must begin with
    // "NAME_SERVER " and be terminated with "\n"
    buf[0] = '\0';
    strcat(buf,"NAME_SERVER",sizeof(buf));
    for (i=1; i<argc; i++) {
        strcat(buf, " ",sizeof(buf));
        strcat(buf,argv[i],sizeof(buf));
    }
    strcat(buf,"\n",sizeof(buf));
    printf("Message to send to name server:\n");
    printf("%s", buf);

    // send a message to the server PORT on machine HOST
    if (send(sd, buf, strlen(buf), 0) == -1) {
        perror("send");
        exit(1);
    }

    // wait for a message to come back from the server
    for (i=0; i<BUFSIZE; i++) {
        buf[i] = '\0';
    }
    if (recv(sd, buf, BUFSIZE, 0) == -1) {
        perror("recv");
        exit(1);
    }

    // print out the results
    printf("Response from yarp server:\n");
    printf("%s", buf);

#ifdef WIN32
    closesocket(sd);
#else
    close(sd);
#endif
}
```



## Chapter 9

# framerate/main.cpp

Measure the framerate of a data source. This example was originally designed for images but should work with anything.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-

#include <stdio.h>

#include <ace/OS.h>
#include <ace/Log_Msg.h>

#include <yarp/String.h>
#include <yarp/sig/Image.h>
#include <yarp/os/BufferedPort.h>
#include <yarp/os/Property.h>
#include <yarp/os/Network.h>
#include <yarp/os/Time.h>
#include <yarp/os/Network.h>

using namespace yarp;
using namespace yarp::os;
using namespace yarp::sig;

static bool terminated=false;

static void handler (int) {
    static int ct = 0;
    ct++;

    fprintf(stderr, "[try %d of %d] Asking to shut down smoothly\n",ct, 3);
    terminated = true;
    if (ct>2)
    {
        fprintf(stderr, "[try %d of %d] OK asking to abort...\n", ct, 3);
        exit(-1);
    }
}
```

```
int main(int argc, char *argv[]) {
    Network yarp;

    ACE_OS::signal(SIGINT, (ACE_SignalHandler) handler);
    ACE_OS::signal(SIGTERM, (ACE_SignalHandler) handler);

    if (argc==1) {
        printf("This program checks the framerate of an output port\n");
        printf("Call as:\n");
        printf("  framerate --remote /port_name --local /local_name --prot protocol\n");
        printf("protocol can be for example tcp,udp,mcast\n");
        exit(0);
    }

    BufferedPort<Bottle> port;

    // get options
    Property opt;
    opt.fromCommand(argc,argv);

    // name port
    Value *val;
    Value *prot;
    String local = "/get_image";
    if (opt.check("local",val)) {
        local = val->asString().c_str();
    }
    port.open(local.c_str());

    // connect port
    if (opt.check("remote", val))
    {
        if (opt.check("prot", prot))
            Network::connect(val->asString(), local.c_str(), prot->asString().c_str());
        else
            Network::connect(val->asString(), local.c_str());
    }
    // read
    double first = Time::now();
    double prev = 0;
    int ct = 0;
    bool spoke = false;
    while (!terminated) {
        Bottle *bot = port.read(true);
        double now = Time::now()-first;
        ct++;
        if (now-prev>=2) {
            double period = (now-prev)/ct;
            printf("Period is %g ms per message, freq is %g (%d mgs in %g secs)\n",
                period*1000, 1/period, ct, now-prev);
            ct = 0;
            prev = now;
            spoke = false;
        }
        if (bot!=NULL) {
```

```
        if (!spoke) {
            printf("Got something with %d top-level elements\n", bot->size());
            spoke = true;
        }
    }
}

return 0;
}
```



## Chapter 10

# imagemagick/main.cpp

Using ImageMagick and YARP

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <Magick++.h>
#include <iostream>

#include <yarp/sig/Image.h>

using namespace std;

void copyImage(yarp::sig::ImageOf<yarp::sig::PixelRgb>& src,
              Magick::Image& dest) {
    int h = src.height();
    int w = src.width();
    dest.size(Magick::Geometry(w,h));
    dest.depth(8);
    for (int i=0; i<h; i++) {
        // must transfer row by row, since YARP may use padding in representation
        Magick::PixelPacket *packet = dest.setPixels(0,i,w,1);
        dest.readPixels(Magick::RGBQuantum,(unsigned char *)&src.pixel(0,i));
    }
    dest.syncPixels();
}

void copyImage(Magick::Image& src,
              yarp::sig::ImageOf<yarp::sig::PixelRgb>& dest) {
    Magick::Geometry g = src.size();
    int h = g.height();
    int w = g.width();
    src.depth(8);
    dest.resize(w,h);
    for (int i=0; i<h; i++) {
        // must transfer row by row, since YARP may use padding in representation
        Magick::PixelPacket *packet = src.getPixels(0,i,w,1);
        src.writePixels(Magick::RGBQuantum,(unsigned char *)&dest.pixel(0,i));
    }
}
```

```
    src.syncPixels();
}

int main(int argc, char **argv)
{
    try {
        yarp::sig::ImageOf<yarp::sig::PixelRgb> yimg1, yimg2;

        yimg1.resize(255,127);
        for (int i=0; i<yimg1.width(); i++) {
            for (int j=0; j<yimg1.height(); j++) {
                yarp::sig::PixelRgb& pix = yimg1.pixel(i,j);
                pix.r = ((i+j)/2)%256;
                pix.g = i%256;
                pix.b = j%256;
            }
        }

        printf("Creating a YARP image, and showing the value of one pixel\n");

        yarp::sig::PixelRgb& pixel1 = yimg1.pixel(10,20);
        printf("rgb %d %d %d\n", pixel1.r, pixel1.g, pixel1.b);

        Magick::Image mimg;
        copyImage(yimg1,mimg);

        printf("Copying image to Magick, and tracking the value of the same pixel\n");

        Magick::Color c = mimg.pixelColor(10, 20);
        printf("rgb %d %d %d\n", c.redQuantum(),c.greenQuantum(),c.blueQuantum());

        printf("Saving image as test.gif\n");
        mimg.write("test.gif");

        copyImage(mimg,yimg2);

        printf("Copying image back to YARP, and tracking the value of the same pixel\n");

        yarp::sig::PixelRgb& pixel2 = yimg2.pixel(10,20);
        printf("rgb %d %d %d\n", pixel2.r, pixel2.g, pixel2.b);
    }
    catch( Magick::Exception &error_ )
    {
        cout << "Caught exception: " << error_.what() << endl;
        return 1;
    }
    return 0;
}
```







```
cvCvtColor((IplImage*)yarpImage.getIplImage(), cvImage, CV_RGB2BGR);

printf("Showing OpenCV/IPL image\n");
cvNamedWindow("test",1);
cvShowImage("test",cvImage);

printf("Taking image back into YARP...\n");
ImageOf<PixelBgr> yarpReturnImage;
yarpReturnImage.wrapIplImage(cvImage);
yarp::sig::file::write(yarpReturnImage,"test.ppm");
printf("Saving YARP image to test.ppm\n");

cvWaitKey(3000);

cvDestroyWindow("test");

cvReleaseImage(&cvImage);

printf("...done\n");
return 0;
}
```





## Chapter 12

### os/bottle\_add.cpp

How to construct a very simple bottle. See also [os/browse\\_bottlecpp](#) example.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/Bottle.h>
using namespace yarp::os;

int main() {
    // create a bottle representing the list (5,"plus",2,"is")
    Bottle b("5 plus 2 is");
    // add an integer that is the sum of element 0 and 2
    b.addInt(b.get(0).asInt()+b.get(2).asInt());
    // print the result -- "result: 5 plus 2 is 7"
    printf("result: %s\n", b.toString().c_str());
    return 0;
}
```



# Chapter 13

## os/browse\_bottle.cpp

Demonstrates one way to access bottle objects. See also [os/bottle\\_addecpp](#)

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/Bottle.h>
#include <yarp/os/Vocab.h>
using namespace yarp::os;

void showBottle(Bottle& anUnknownBottle, int indentation = 0) {
    for (int i=0; i<anUnknownBottle.size(); i++) {
        for (int j=0; j<indentation; j++) { printf(" "); }
        printf("[%d]: ", i);
        Value& element = anUnknownBottle.get(i);
        switch (element.getCode()) {
            case BOTTLE_TAG_INT:
                printf("int %d\n", element.asInt());
                break;
            case BOTTLE_TAG_DOUBLE:
                printf("float %g\n", element.asDouble());
                break;
            case BOTTLE_TAG_STRING:
                printf("string \"%s\"\n", element.asString().c_str());
                break;
            case BOTTLE_TAG_BLOB:
                printf("binary blob of length %d\n", element.asBlobLength());
                break;
            case BOTTLE_TAG_VOCAB:
                printf("vocab [%s]\n", Vocab::decode(element.asVocab()).c_str());
                break;
            default:
                if (element.isList()) {
                    Bottle *lst = element.asList();
                    printf("list of %d elements\n", lst->size());
                    showBottle(*lst,indentation+2);
                } else {
                    printf("unrecognized type\n");
                }
        }
    }
}
```

```
        }
        break;
    }
}

int main() {
    Bottle anUnknownBottle("equals 7 (add (add 2 3) 2)");
    showBottle(anUnknownBottle);
    return 0;
}
```





# Chapter 14

## os/buffered\_port.cpp

This example shows how to communicate between a pair of buffered ports.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <yarp/os/Network.h>
#include <yarp/os/BufferedPort.h>
#include <yarp/os/Bottle.h>
#include <yarp/os/Time.h>
#include <stdio.h>

using namespace yarp::os;

int main() {
    // Initialize YARP - some OSes need network and time service initialization
    Network yarp;

    // Work locally - don't rely on name server (just for this example).
    // If you have a YARP name server running, you can remove this line.
    Network::setLocalMode(true);

    // Create two ports that we'll be using to transmit "Bottle" objects.
    // The ports are buffered, so that sending and receiving can happen
    // in the background.
    BufferedPort<Bottle> in;
    BufferedPort<Bottle> out;

    // we will want to read every message, with no skipping of "old" messages
    // when new ones come in
    in.setStrict();

    // Name the ports
    in.open("/in");
    out.open("/out");

    // Connect the ports so that anything written from /out arrives to /in
    Network::connect("/out","/in");

    // Send one "Bottle" object. The port is responsible for creating
```

```
// and reusing/destroying that object, since it needs to be sure
// it exists until communication to all recipients (just one in
// this case) is complete.
Bottle& outBot1 = out.prepare(); // Get the object
outBot1.fromString("hello world"); // Set it up the way we want
printf("Writing bottle 1 (%s)\n", outBot1.toString().c_str());
out.write(); // Now send it on its way

// Send another "Bottle" object
Bottle& outBot2 = out.prepare();
outBot2.fromString("2 3 5 7 11");
printf("Writing bottle 2 (%s)\n", outBot2.toString().c_str());
out.writeStrict(); // writeStrict() will wait for any
// previous communication to finish;
// write() would skip sending if
// there was something being sent

// Read the first object
Bottle *inBot1 = in.read();
printf("Bottle 1 is: %s\n", inBot1->toString().c_str());

// Read the second object
Bottle *inBot2 = in.read();
printf("Bottle 2 is: %s\n", inBot2->toString().c_str());

return 0;
}
```





# Chapter 15

## os/database.cpp

A toy "database" program for storing and fetching key-values, accessible from a port.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>

using namespace yarp::os;

#define VOCAB_SET VOCAB3('s','e','t')
#define VOCAB_GET VOCAB3('g','e','t')
#define VOCAB_NOT VOCAB3('n','o','t')
#define VOCAB_IS VOCAB2('i','s')
#define VOCAB_REMOVE VOCAB2('r','m')

int main(int argc, char *argv[]) {
    if (argc<=1) {
        printf("This is a very simple database\n");
        printf("Call as: %s --name /database\n", argv[0]);
        printf("Then you can test it by running:\n");
        printf("  yarp rpc /database\n");
        printf("And typing things like:\n");
        printf("  set x 24\n");
        printf("  get x\n");
        printf("  get y\n");
        printf("  rm x\n");
        printf("  get x\n");
        printf("  set \"my favorite numbers\" (5 10 16)\n");
        printf("  get \"my favorite numbers\"\n");
    }

    Network yarp;

    Property option;
    option.fromCommand(argc,argv);
```

```
Property state;

Port port;
port.open(option.check("name", Value("/database")).asString());

while (true) {
    Bottle cmd;
    Bottle response;
    port.read(cmd, true); // true -> will reply

    Bottle tmp;
    tmp.add(cmd.get(1));
    ConstString key = tmp.toString();

    switch (Vocab::encode(cmd.get(0).toString())) {
    case VOCAB_SET:
        state.put(key, cmd.get(2));
        break;
    case VOCAB_GET:
        break;
    case VOCAB_REMOVE:
        state.unput(key);
        break;
    }
    Value& v = state.find(key);
    response.addVocab(v.isNull()?VOCAB_NOT:VOCAB_IS);
    response.add(cmd.get(1));
    if (!v.isNull()) {
        response.add(v);
    }
    port.reply(response);
}

return 0;
}
```





# Chapter 16

## os/image\_process.cpp

An example of how to receive, process, and output images to and from a port.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <ace/config.h>

#include <yarp/os/all.h>
#include <yarp/sig/all.h>

using namespace yarp::os;
using namespace yarp::sig;
using namespace yarp::sig::draw;

/*
  This example adds a moving circle to an image stream.

  Suppose we have an image source on port /source such as:
  yarpdev --device test_grabber --name /source --mode line --framerate 10

  And suppose we have an image viewer on port /view:
  yarpview --name /view

  Then we can hook this program up in between as follows:
  ./image_process --name /worker
  yarp connect /source /worker
  yarp connect /worker /view

  You should see the normal scrolling line of test_grabber, with a moving
  circle overlaid.

*/

int main(int argc, char *argv[]) {

  // Initialize network
  Network yarp;

  // Make a port for reading and writing images
```

```
BufferedPort<ImageOf<PixelRgb> > port;

// Get command line options
Property options;
options.fromCommand(argc,argv);

// Set the name of the port (use "/worker" if there is no --name option)
ConstString portName = options.check("name",Value("/worker")).asString();
port.open(portName);

int ct = 0;
while (true) {
    // read an image from the port
    ImageOf<PixelRgb> *img = port.read();
    if (img==NULL) continue;

    // add a blue circle
    PixelRgb blue(0,0,255);
    addCircle(*img,blue,ct,50,10);
    ct = (ct+5)%img->width();

    // output the image
    port.prepare() = *img;
    port.write();
}

return 0;
}
```





## Chapter 17

# os/image\_process\_module.cpp

This is the [os/image\\_process](#) example rewritten as a "module". It inherits better starting up, configuring, and shutting down behavior.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <ace/config.h>

#include <yarp/os/all.h>
#include <yarp/sig/all.h>

using namespace yarp::os;
using namespace yarp::sig;
using namespace yarp::sig::draw;

/*
   This example adds a moving circle to an image stream.
   It is the same as image_process.cpp, except it is built
   as a Yarp "Module". This makes it a bit cleaner to start/stop.

   Suppose we have an image source on port /source such as:
       yarpdev --device test_grabber --name /source --mode line --framerate 10

   And suppose we have an image viewer on port /view:
       yarpview --name /view

   Then we can hook this program up in between as follows:
       ./image_process --name /worker
       yarp connect /source /worker
       yarp connect /worker /view

   You should see the normal scrolling line of test_grabber, with a moving
   circle overlaid.
```

```
*/

class ImageProcessModule : public Module {
private:
    // Make a port for reading and writing images
    BufferedPort<ImageOf<PixelRgb> > port;
    Port cmdPort;
    int ct;
public:
    bool open(Searchable& config) {
        ct = 0;
        port.open(getName());
        cmdPort.open(getName("cmd")); // optional command port
        attach(cmdPort); // cmdPort will work just like terminal
        return true;
    }

    // try to interrupt any communications or resource usage
    bool interruptModule() {
        port.interrupt();
        return true;
    }

    bool updateModule() {
        ImageOf<PixelRgb> *img = port.read();
        if (img==NULL) return false;;

        // add a blue circle
        PixelRgb blue(0,0,255);
        addCircle(*img,blue,ct,50,10);
        ct = (ct+5)%img->width();

        // output the image
        port.prepare() = *img;
        port.write();

        return true;
    }
};

int main(int argc, char *argv[]) {
    // Initialize the yarp network
    Network yarp;

    // Create and run our module
    ImageProcessModule module;
    module.setName("/worker");
    return module.runModule(argc,argv);
}
```





# Chapter 18

## os/image\_source.cpp

An example of how to create a source of images as a port.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <ace/config.h>

#include <yarp/os/all.h>
#include <yarp/sig/all.h>

using namespace yarp::os;
using namespace yarp::sig;
using namespace yarp::sig::draw;

int main(int argc, char *argv[]) {
    Network yarp;
    BufferedPort<ImageOf<PixelRgb> > port;

    Property options;
    options.fromCommand(argc,argv);
    port.open(options.check("name",Value("/image")).asString());

    int ct = 0;
    while (true) {
        ImageOf<PixelRgb>& img = port.prepare();
        img.resize(100,100);
        img.zero();
        PixelRgb blue(0,0,255);
        addCircle(img,blue,ct,50,10);
        ct = (ct+5)%100;
        port.write();
        Time::delay(0.25);
    }

    return 0;
}
```



## Chapter 19

### os/make\_count.cpp

Send countdown messages from a port (paired with [os/view\\_count.cpp](#) example).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <yarp/os/all.h>
#include <stdio.h>
using namespace yarp::os;

int main(int argc, char *argv[]) {
    if (argc!=2) return 1;
    Network yarp;

    BufferedPort<Bottle> out;
    out.open(argv[1]);

    for (int i=10; i>=0; i--) {
        printf("at %d\n", i);
        Bottle& msg = out.prepare();
        msg.clear();
        msg.addString("countdown");
        msg.addInt(i);
        out.write();
        Time::delay(1);
    }
    return 0;
}
```



## Chapter 20

# os/portable\_pair.cpp

Show use of the `yarp::os::PortablePair` class, to send messages of mixed type. If the component types follow the YARP network data format, then their composition will also.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <ace/config.h>
#include <yarp/os/all.h>
#include <yarp/sig/all.h>

using namespace yarp::os;
using namespace yarp::sig;

int main(int argc, char *argv[]) {
    Network yarp;
    Port port;
    PortablePair<Bottle,Vector> pp;
    port.open("/pp");
    pp.head.fromString("this is the bottle part");
    int ct = 1;
    int ct2 = 1;
    while (true) {
        Vector v(3);
        v[0] = ct;
        v[1] = ct2;
        v[2] = ct*ct2;
        pp.body = v;
        port.write(pp);
        printf("Sent output to %s...\n", port.getName().c_str());
        ct++;
        if (ct>10) {
            ct2 = 1+(ct2+1)%10;
            ct = 1;
        }
        Time::delay(0.25);
    }
    return 0;
}
```

```
}
```





# Chapter 21

## os/ratethread.cpp

Show a class for running some processing at a specified interval.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-

// rate thread example -nat

#include <stdio.h>

#include <yarp/os/RateThread.h>
#include <yarp/os/Time.h>
#include <yarp/os/Thread.h>

using namespace yarp::os;

class Thread1 : public RateThread {
public:
    Thread1(int r):RateThread(r){}
    virtual bool threadInit()
    {
        printf("Starting thread1\n");
        return true;
    }

    //called by start after threadInit, s is true iff the thread started
    //successfully
    virtual void afterStart(bool s)
    {
        if (s)
            printf("Thread1 started successfully\n");
        else
            printf("Thread1 did not start\n");
    }

    virtual void run()
    {
        printf("Hello, from thread1\n");
    }
};
```

```
    }

    virtual void threadRelease()
    {
        printf("Goodbye from thread1\n");
    }
};

class Thread2: public RateThread {
public:
    Thread2(int r):RateThread(r){}
    virtual bool threadInit()
    {
        printf("Starting thread2\n");
        return true;
    }

    //called by start after threadInit, s is true iff the thread started
    //successfully
    virtual void afterStart(bool s)
    {
        if (s)
            printf("Thread2 started successfully\n");
        else
            printf("Thread2 did not start\n");
    }

    virtual void run()
    {
        printf("Hello, from thread2\n");
    }

    virtual void threadRelease()
    {
        printf("Goodbye from thread2\n");
    }
};

int main() {
    Thread1 t1(500); //run every 500ms
    Thread2 t2(1000); //run every 1s
    printf("thread1 rate is %d[ms]\n", 500);
    printf("thread2 rate is %d[ms]\n", 1000);

    printf("Starting threads...\n");
    bool ok=t1.start();
    ok = ok&& t2.start();
    if (!ok)
    {
        printf("One of the thread failed to initialize, returning\n");
        return -1;
    }

    Time::delay(3);
    printf("Thread1 ran %d times, estimated rate: %.1f[ms]\n", t1.getIterations(), t1.getEstPeriod());
    printf("Thread2 ran %d times, estimated rate: %.1f[ms]\n", t2.getIterations(), t2.getEstPeriod());
}
```

---

```
printf("suspending threads...\n");
    t1.suspend();
    t2.suspend();

    printf("Waiting some time");
    for(int k=1;k<20;k++)
    {
        printf(".");
        fflush(stdout);
        Time::delay(0.2); //200[ms]
    }
    printf("\n");

    printf("Changing thread1 rate to %d[ms]\n", 250);
    printf("Changing thread2 rate to %d[ms]\n", 500);

    t1.setRate(250);
    t2.setRate(500);

    printf("Resuming threads...\n");
    t1.resetStat();
    t2.resetStat();
    t1.resume();
    t2.resume();

    Time::delay(3);

    printf("Thread1 ran %d times, estimated rate: %.1f[ms]\n", t1.getIterations(), t1.getEstPeriod());
    printf("Thread2 ran %d times, estimated rate: %.1f[ms]\n", t2.getIterations(), t2.getEstPeriod());
    t1.stop();
    t2.stop();
    printf("stopped\n");

    return 0;
}
```



## Chapter 22

### os/simple\_receiver.cpp

This reads some data from a port. Designed to be used with [os/simple\\_sendercpp](#) example.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <yarp/os/Network.h>
#include <yarp/os/Port.h>
#include <yarp/os/Bottle.h>
#include <stdio.h>

using namespace yarp::os;

int main() {
    Network yarp;
    Bottle bot;
    Port input;
    input.open("/receiver");
    // usually, we create connections externally, but just for this example...
    Network::connect("/sender", "/receiver");
    input.read(bot);
    printf("Got message: %s\n", bot.toString().c_str());
    input.close();
    return 0;
}
```



## Chapter 23

### os/simple\_sender.cpp

This outputs some data to a port. Designed to be used with [os/simple\\_receivercpp](#) example.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <yarp/os/Network.h>
#include <yarp/os/Port.h>
#include <yarp/os/Bottle.h>
#include <yarp/os/Time.h>
#include <stdio.h>

using namespace yarp::os;

int main() {
    Network yarp;
    Port output;
    output.open("/sender");
    int top = 100;
    for (int i=1; i<=top; i++) {
        // prepare a message
        Bottle bot;
        bot.addString("testing");
        bot.addInt(i);
        bot.addString("of");
        bot.addInt(top);
        // send the message
        output.write(bot);
        printf("Sent message: %s\n", bot.toString().c_str());
        // wait a while
        Time::delay(1);
    }
    output.close();
    return 0;
}
```



# Chapter 24

## os/summer.cpp

Add up numbers received on a port, and send the result back out.

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <yarp/os/all.h>
#include <iostream>

using namespace std;
using namespace yarp::os;

int main(int argc, char *argv) {
    Network yarp;
    BufferedPort<Bottle> port;
    port.open("/summer");
    while (true) {
        cout << "waiting for input" << endl;
        Bottle *input = port.read();
        if (input!=NULL) {
            cout << "got " << input->toString().c_str() << endl;
            double total = 0;
            for (int i=0; i<input->size(); i++) {
                total += input->get(i).asDouble();
            }
            Bottle& output = port.prepare();
            output.clear();
            output.addString("total");
            output.addDouble(total);
            cout << "writing " << output.toString().c_str() << endl;
            port.write();
        }
    }
    return 0;
}
```



## Chapter 25

# os/threads.cpp

Demonstrate the basic use of threads. See also [os/ratethreadcpp](#)

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
// Show thread basic functionalities, you may want to have a look at the
// ratethread example.

// added initThread/releaseThread example -nat

#include <stdio.h>

#include <yarp/os/Thread.h>
#include <yarp/os/Time.h>

using namespace yarp::os;

class Thread1 : public Thread {
public:
    virtual bool threadInit()
    {
        printf("Starting thread1\n");
        return true;
    }

    virtual void run() {
        while (!isStopping()) {
            printf("Hello, from thread1\n");
            Time::delay(1);
        }
    }

    virtual void threadRelease()
    {
        printf("Goodbye from thread1\n");
    }
};
```

```
class Thread2 : public Thread {
public:
    virtual bool threadInit()
    {
        printf("Starting thread2\n");
        return true;
    }

    virtual void run() {
        Time::delay(0.5);
        while (!isStopping()) {
            printf("Hello from thread2\n");
            Time::delay(1);
        }
    }

    virtual void threadRelease()
    {
        printf("Goodbye from thread2\n");
    }
};

int main() {
    Thread1 t1;
    Thread2 t2;
    printf("Starting threads...\n");
    t1.start();
    t2.start();
    printf("started\n");
    Time::delay(3);
    printf("stopping threads...\n");
    t1.stop();
    t2.stop();
    printf("stopped\n");
    return 0;
}
```





## Chapter 26

### os/view\_count.cpp

View countdown messages coming from a port (paired with [os/make\\_countcpp](#) example).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <yarp/os/all.h>
#include <stdio.h>
using namespace yarp::os;

int main(int argc, char *argv[]) {
    if (argc!=2) return 1;
    Network yarp;

    BufferedPort<Bottle> in;
    in.open(argv[1]);

    int count = 1;
    while (count>0) {
        Bottle *msg = in.read();
        count = msg->get(1).asInt();
        printf("at %d\n", count);
    }

    return 0;
}
```



## Chapter 27

# port\_power/ex0000\_receiver.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

int main() {
    Network yarp;

    Port p;           // Create a port.
    p.open("/in");    // Give it a name on the network.
    Bottle b;         // Make a place to store things.
    while (true) {
        p.read(b);    // Read from the port.  Waits until data arrives.
        // Do something with data.
        printf("Got %s\n", b.toString().c_str());
    }

    return 0;
}
```



## Chapter 28

# port\_power/ex0001\_sender.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

int main() {
    Network yarp;

    int ct = 0;
    Port p;          // Create a port.
    p.open("/out"); // Give it a name on the network.
    while (true) {
        Bottle b;          // Make a place to store things.
        b.clear();
        b.add("hello");
        b.add("world");
        b.add(ct);
        ct++;
        p.write(b);       // Send the data.
        printf("Sent %s\n", b.toString().c_str());
        Time::delay(1);
    }

    return 0;
}
```



## Chapter 29

# port\_power/ex0002\_- connector.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

int main() {
    Network yarp;

    Network::connect("/out","/in"); // connect two ports.
    // can do the same thing from command line with "yarp connect /out /in"

    return 0;
}
```



## Chapter 30

# port\_power/ex0100\_receiver.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

int main() {
    Network yarp;

    BufferedPort<Bottle> p; // Create a port.
    p.open("/in");         // Give it a name on the network.
    while (true) {
        Bottle *b = p.read(); // Read from the port.  Waits until data arrives.
        if (b==NULL) continue;
        printf("Got %s\n", b->toString().c_str());
    }

    return 0;
}
```



## Chapter 31

# port\_power/ex0101\_sender.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

int main() {
    Network yarp;

    int ct = 0;
    BufferedPort<Bottle> p; // Create a port.
    p.open("/out");        // Give it a name on the network.
    while (true) {
        Bottle& b = p.prepare(); // Get a place to store things.
        b.clear(); // clear is important - b might be a reused object
        b.add("hello");
        b.add("world");
        b.add(ct);
        ct++;
        printf("Sending %s\n", b.toString().c_str());
        p.write(); // Send the data.
        // after write() is called, user should not touch b.
        Time::delay(1);
    }

    return 0;
}
```



## Chapter 32

# port\_power/ex0200 - polling.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

int main() {
    Network yarp;

    BufferedPort<Bottle> p; // Create a port.
    p.open("/in");         // Give it a name on the network.
    while (true) {
        Bottle *b = p.read(false); // Read from the port. Don't wait
        if (b!=NULL) {
            printf("Got %s\n", b->toString().c_str());
        } else {
            printf("No data yet...\n");
            Time::delay(0.5);
        }
    }

    return 0;
}
```



## Chapter 33

# port\_power/ex0300\_port\_callback.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

class DataProcessor : public PortReader {
    virtual bool read(ConnectionReader& connection) {
        Bottle b;
        b.read(connection);
        // process data in b
        printf("Got %s\n", b.toString().c_str());
    }
};

DataProcessor processor;

int main() {
    Network yarp;

    Port p;           // Create a port.
    p.open("/in");    // Give it a name on the network.
    p.setReader(processor); // no need to call p.read() on port any more.

    while (true) {
        printf("main thread free to do whatever it wants\n");
        Time::delay(10);
    }

    return 0;
}
```



## Chapter 34

# port\_power/ex0301\_buffered\_callback.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

class DataPort : public BufferedPort<Bottle> {
    virtual void onRead(Bottle& b) {
        // process data in b
        printf("Got %s\n", b.toString().c_str());
    }
};

int main() {
    Network yarp;

    DataPort p;
    p.useCallback(); // input should go to onRead() callback
    p.open("/in"); // Give it a name on the network.
    while (true) {
        printf("main thread free to do whatever it wants\n");
        Time::delay(10);
    }

    return 0;
}
```



## Chapter 35

# port\_power/ex0302\_buffered\_ext\_callback.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

class DataProcessor : public TypedReaderCallback<Bottle> {
    virtual void onRead(Bottle& b) {
        // process data in b
        printf("Got %s\n", b.toString().c_str());
    }
};

DataProcessor processor;

int main() {
    Network yarp;

    DataProcessor processor;
    BufferedPort<Bottle> p;
    p.useCallback(processor); // input should go to processor.onRead()
    p.open("/in");           // Give it a name on the network.
    while (true) {
        printf("main thread free to do whatever it wants\n");
        Time::delay(10);
    }

    return 0;
}
```



## Chapter 36

# port\_power/ex0400 - expect\_reply.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

int main() {
    Network yarp;

    int ct = 0;
    Port p;          // Create a port.
    p.open("/out"); // Give it a name on the network.
    while (true) {
        Bottle in,out; // Make places to store things.
        // prepare command "out".
        out.clear();
        out.add("hello");
        out.add("world");
        out.add(ct);
        ct++;
        p.write(out,in); // send command, wait for reply.
        // process response "in".
        if (in.size()>0) {
            printf("Got response: %s\n", in.toString().c_str());
        } else {
            printf("No response\n");
        }
        Time::delay(1);
    }

    return 0;
}
```

```
}
```





## Chapter 37

# port\_power/ex0401\_give\_reply.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

int main() {
    Network yarp;

    Port p;           // Create a port.
    p.open("/in");    // Give it a name on the network.
    Bottle in, out;   // Make places to store things.
    while (true) {
        p.read(in,true); // Read from the port, warn that we'll be replying.
        printf("Got %s\n", in.toString().c_str());
        out.clear();
        out.add("acknowledge");
        out.append(in);
        p.reply(out);    // send reply.
    }
}
```



## Chapter 38

# port\_power/ex0402\_port\_callback\_reply.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

class DataProcessor : public PortReader {
    virtual bool read(ConnectionReader& connection) {
        Bottle in, out;
        in.read(connection);
        // process data "in", prepare "out"
        printf("Got %s\n", in.toString().c_str());
        out.clear();
        out.add("acknowledge");
        out.append(in);
        ConnectionWriter *returnToSender = connection.getWriter();
        if (returnToSender!=NULL) {
            out.write(*returnToSender);
        }
    }
};

DataProcessor processor;

int main() {
    Network yarp;

    Port p;          // Create a port.
    p.open("/in");   // Give it a name on the network.
    p.setReader(processor); // no need to call p.read() on port any more.

    while (true) {
```

```
        printf("main thread free to do whatever it wants\n");
        Time::delay(10);
    }

    return 0;
}
```





## Chapter 39

# port\_power/ex0403\_bufferedport\_callback\_reply.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

class DataProcessor : public TypedReaderCallback<Bottle>, public PortReader {
    virtual void onRead(Bottle& b) {
        // process data in b
        printf("Got one-way message: %s\n", b.toString().c_str());
    }

    virtual bool read(ConnectionReader& connection) {
        Bottle in, out;
        in.read(connection);
        // process data "in", prepare "out"
        printf("Got message to reply to: %s\n", in.toString().c_str());
        out.clear();
        out.add("acknowledge");
        out.append(in);
        ConnectionWriter *returnToSender = connection.getWriter();
        if (returnToSender!=NULL) {
            out.write(*returnToSender);
        }
    }
};

DataProcessor processor;
```

```
int main() {
    Network yarp;

    DataProcessor processor;
    BufferedPort<Bottle> p;
    p.useCallback(processor); // input should go to processor.onRead()
    p.setReplier(processor); // input with replt goes to processor.read()
    p.open("/in");           // Give it a name on the network.
    while (true) {
        printf("main thread free to do whatever it wants\n");
        Time::delay(10);
    }

    return 0;
}
```





## Chapter 40

# port\_power/ex0500\_raw\_target\_receiver.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

// define the Target class
#include "TargetVer1.h"

int main() {
    Network yarp;

    Port p;          // Create a port.
    p.open("/target/raw/in"); // Give it a name on the network.
    BinPortable<Target> b; // Make a place to store things.
    while (true) {
        p.read(b); // Read from the port. Waits until data arrives.
        // Do something with data.
        printf("Got (%d,%d)\n", b.content().x, b.content().y);
    }

    return 0;
}
```



## Chapter 41

# port\_power/ex0501\_raw\_target\_sender.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

// define the Target class
#include "TargetVer1.h"

int main() {
    Network yarp;

    int ct = 0;
    Port p;          // Create a port.
    p.open("/target/raw/out"); // Give it a name on the network.
    while (true) {
        BinPortable<Target> b; // Make a place to store things.
        b.content().x = ct;
        b.content().y = 42;
        ct++;
        p.write(b); // Send the data.
        printf("Sent (%d,%d)\n", b.content().x, b.content().y);
        Time::delay(1);
    }

    return 0;
}
```



## Chapter 42

# port\_power/ex0502\_raw\_target\_connector.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

int main() {
    Network yarp;

    Network::connect("/target/raw/out", "/target/raw/in"); // connect ports.
    // can do the same thing from command line with
    // "yarp connect /target/raw/out /target/raw/in"

    return 0;
}
```



## Chapter 43

# port\_power/ex0503\_serial\_target\_receiver.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

// define the Target class
#include "TargetVer2.h"

int main() {
    Network yarp;

    Port p;          // Create a port.
    p.open("/target/raw/in"); // Give it a name on the network.
    Target t;        // Make a place to store things.
    while (true) {
        p.read(t);   // Read from the port.  Waits until data arrives.
        // Do something with data.
        printf("Got (%d,%d)\n", t.x, t.y);
    }

    return 0;
}
```



## Chapter 44

# port\_power/ex0504\_serial\_target\_sender.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

// define the Target class
#include "TargetVer2.h"

int main() {
    Network yarp;

    int ct = 0;
    Port p;          // Create a port.
    p.open("/target/raw/out"); // Give it a name on the network.
    while (true) {
        Target t;    // Make a place to store things.
        t.x = ct;
        t.y = 42;
        ct++;
        p.write(t);  // Send the data.
        printf("Sent (%d,%d)\n", t.x, t.y);
        Time::delay(1);
    }

    return 0;
}
```



## Chapter 45

# port\_power/ex0505\_compliant\_target\_receiver.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

// define the Target class
#include "TargetVer3.h"

int main() {
    Network yarp;

    Port p;          // Create a port.
    p.open("/in");   // Give it a name on the network.
    Target t;        // Make a place to store things.
    while (true) {
        p.read(t);   // Read from the port.  Waits until data arrives.
        // Do something with data.
        printf("Got (%d,%d)\n", t.x, t.y);
    }

    return 0;
}
```



## Chapter 46

# port\_power/ex0506\_compliant\_target\_sender.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

// define the Target class
#include "TargetVer3.h"

int main() {
    Network yarp;

    int ct = 0;
    Port p;           // Create a port.
    p.open("/out");  // Give it a name on the network.
    while (true) {
        Target t;    // Make a place to store things.
        t.x = ct;
        t.y = 42;
        ct++;
        p.write(t);  // Send the data.
        printf("Sent (%d,%d)\n", t.x, t.y);
        Time::delay(1);
    }

    return 0;
}
```



## Chapter 47

# port\_power/ex0507\_alternative\_compliant\_target\_receiver.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

// define the Target class
#include "TargetVer1b.h"

int main() {
    Network yarp;

    Port p;          // Create a port.
    p.open("/target/raw/in"); // Give it a name on the network.
    BinPortable<Target> b; // Make a place to store things.
    while (true) {
        p.read(b); // Read from the port. Waits until data arrives.
        // Do something with data.
        printf("Got (%d,%d)\n", b.content().x, b.content().y);
    }

    return 0;
}
```

14port\_power/ex0507\_alternative\_compliant\_target\_receiver.cpp

## Chapter 48

# port\_power/ex0508\_alternative\_compliant\_target\_sender.cpp

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <yarp/os/all.h>
using namespace yarp::os;

// define the Target class
#include "TargetVer1b.h"

int main() {
    Network yarp;

    int ct = 0;
    Port p;          // Create a port.
    p.open("/target/raw/out"); // Give it a name on the network.
    while (true) {
        BinPortable<Target> b; // Make a place to store things.
        b.content().x = ct;
        b.content().y = 42;
        ct++;
        p.write(b); // Send the data.
        printf("Sent (%d,%d)\n", b.content().x, b.content().y);
        Time::delay(1);
    }

    return 0;
}
```

146 port\_power/ex0508\_alternative\_compliant\_target\_sender.cpp

## Chapter 49

# port\_power/TargetVer1.h

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-

#ifndef TARGETVER1_INC
#define TARGETVER1_INC

#include <yarp/os/begin_pack_for_net.h>
class Target {
public:
    NetInt32 x;
    NetInt32 y;
} PACKED_FOR_NET;
#include <yarp/os/end_pack_for_net.h>

#endif
```



## Chapter 50

# port\_power/TargetVer1b.h

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-

#ifndef TARGETVER1B_INC
#define TARGETVER1B_INC

#include <yarp/os/Bottle.h>

#include <yarp/os/begin_pack_for_net.h>
class Target {
public:
    Target() {
        tag = BOTTLE_TAG_LIST + BOTTLE_TAG_INT;
        len = 2;
    }

    NetInt32 tag;
    NetInt32 len;
    NetInt32 x;
    NetInt32 y;
} PACKED_FOR_NET;
#include <yarp/os/end_pack_for_net.h>

#endif
```



## Chapter 51

# port\_power/TargetVer2.h

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-

#ifndef TARGETVER1_INC
#define TARGETVER1_INC

#include <yarp/os/Portable.h>

class Target : public yarp::os::Portable {
public:
    int x;
    int y;
    virtual bool write(yarp::os::ConnectionWriter& connection) {
        connection.appendInt(x);
        connection.appendInt(y);
        return true;
    }
    virtual bool read(yarp::os::ConnectionReader& connection) {
        x = connection.expectInt();
        y = connection.expectInt();
        return true;
    }
};

#endif
```



## Chapter 52

# port\_power/TargetVer3.h

Part of a series of examples on the different ways of using ports. See [Port power tutorial](#).

```
// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-

#ifndef TARGETVER1_INC
#define TARGETVER1_INC

#include <yarp/os/Portable.h>

class Target : public yarp::os::Portable {
public:
    int x;
    int y;
    virtual bool write(yarp::os::ConnectionWriter& connection) {
        connection.appendInt(BOTTLE_TAG_LIST+BOTTLE_TAG_INT);
        connection.appendInt(2); // two elements
        connection.appendInt(x);
        connection.appendInt(y);
        connection.convertTextMode(); // if connection is text-mode, convert!
        return true;
    }
    virtual bool read(yarp::os::ConnectionReader& connection) {
        connection.convertTextMode(); // if connection is text-mode, convert!
        int tag = connection.expectInt();
        x = y = -1;
        if (tag!=BOTTLE_TAG_LIST+BOTTLE_TAG_INT) return false;
        int ct = connection.expectInt();
        if (ct!=2) return false;
        x = connection.expectInt();
        y = connection.expectInt();
        return true;
    }
};

#endif
```



# Chapter 53

## property/main.cpp

How to read a configuration file. This example is designed to read a file like the following (property/config.txt):

```
//
// Initialization file for robot head, 8 dof on can bus controller.
//

[GENERAL]
Joints 8
MaxDAC 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0

AxisMap 1 7 0 6 2 3 5 4
Encoder 360.0 360.0 360.0 360.0 360.0 360.0 360.0 360.0
Zeros 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Signs 0 0 0 0 0 0 0 0
Stiff 1 1 1 1 1 1 1 1
CurrentLimits 1220.0 1220.0 1220.0 1220.0 10000.0 10000.0 10000.0 10000.0

CanAddresses 14 13 12 11 128 128 128 128 128 128 128 128 128 128 128 128

// limits are: neck pan, neck tilt, eye tilt, version, vergence
[LIMITS]
Max 250000 250000 250000 250000 180000 180000 180000
Min -250000 -250000 -250000 -250000 -250000 -180000 -180000 -180000

[HIGHPID]
Pid0 32 128 2 0 0 1333 0 1333 4 0
Pid1 32 128 2 0 0 1333 0 1333 4 0
Pid2 32 128 2 0 0 1333 0 1333 4 0
Pid3 32 128 2 0 0 1333 0 1333 4 0
Pid4 32 128 4 0 0 1333 0 1333 4 0
Pid5 32 128 1 0 0 1333 0 1333 4 0
Pid6 32 128 1 0 0 1333 0 1333 4 0
Pid7 32 128 1 0 0 1333 0 1333 4 0
```

```
[LOWPID]
Pid0 0 0 0 0 0 32767 0 32767 0 0
Pid1 0 0 0 0 0 32767 0 32767 0 0
Pid2 0 0 0 0 0 32767 0 32767 0 0
Pid3 0 0 0 0 0 32767 0 32767 0 0
Pid4 0 0 0 0 0 32767 0 32767 0 0
Pid5 0 0 0 0 0 32767 0 32767 0 0
Pid6 0 0 0 0 0 32767 0 32767 0 0
Pid7 0 0 0 0 0 32767 0 32767 0 0

// -*- mode:C++; tab-width:4; c-basic-offset:4; indent-tabs-mode:nil -*-
#include <stdio.h>
#include <stdlib.h>

#include <yarp/os/Property.h>

using namespace yarp::os;

int main(int argc, char *argv[]) {
    Property cmdLine;
    cmdLine.fromCommand(argc,argv);

    if (!cmdLine.check("file")) {
        printf("Please call with: --file config.txt\n");
        exit(1);
    }
    ConstString fname = cmdLine.find("file").asString();
    printf("Working with config file %s\n", fname.c_str());

    Property robot;
    robot.fromConfigFile(fname.c_str());
    if (!robot.check("GENERAL")) {
        printf("Cannot understand config file\n");
        exit(1);
    }

    int joints = robot.findGroup("GENERAL").find("Joints").asInt();
    printf("Robot has %d joints\n", joints);

    Bottle& maxes = robot.findGroup("LIMITS").findGroup("Max");
    printf("Robot has limits: ");
    for (int i=1; i<maxes.size(); i++) {
        printf("%d ", maxes.get(i).asInt());
    }
    printf("\n");

    //printf("If you wanted to transmit this configuration, here it is in Bottle format:\n");
    //printf("%s\n", robot.toString().c_str());

    return 0;
}
```