ZomB Dashboard Bindings

Team 451 “The Cat Attack”

Revision 1.0

Table Of Contents

Overview

The ZomB Dashboard Bindings are the robot side connectors to enable you to easily send data via the ZomB protocol to any ZomB client (ex: a custom dashboard created with Visual ZomB)

Notes

This document is published under the GNU Free Documentation License. This document is a work in progress, and targets ZomB 0.7. This document assumes you have already downloaded and installed the ZomB Dashboard System from http://firstforge.wpi.edu/sf/sfmain/do/viewProject/projects.zombdashboard successfully. All paths are given from a default install on a x86 machine. If you install ZomB on a 64 bit machine, all references to Program Files are to be replaced with Program Files (x86).

Installation

To install the bindings to the robot, use the installer from *Start>All Programs>ZomB>Install Robot Bindings*. This program also opens when you install ZomB 0.7. When it comes up, enter your cRIO's IP address (10.xx.yy.2, xxyy is your team number), make sure your cRIO is booted and visible on the network, read the warning notice and click the *Install* button. This process takes anywhere from about 10 seconds to a minute or so. Once it has succeeded, reboot your cRIO and the ZomB binding should be installed on your cRIO.

If you want to manually install the bindings, or automatic install failed, follow the steps listed below.

1. Connect to *ftp://10.xx.yy.2/* using Windows explorer or any ftp program
2. Upload ZomB.out from *C:\Program Files\ZomB\Bindings\ZomB.out* to *ftp://10.xx.yy.2/ni-rt/system/ZomB.out*
3. Download *ftp://10.xx.yy.2/ni-rt.ini* to your computer and edit the StartupDlls line to include ZomB.out at the end, but before FRC_JavaVM.out or FRC_userProgram.out if they exist.
4. Re-upload ni-rt.ini
5. Reboot

Usage

C++ (Not DBPacket)

Using the C++ bindings is extremely simple. Start by opening the Bindings folder located at *Start>All Programs>ZomB>Bindings*. Drag ZomBDashboard.h onto your project in WindRiver. Example code:

```
#include "WPILib.h"
#include "ZomBDashboard.h"
class RobotDemo : public SimpleRobot
{
      Joystick stick;
      ZomBDashboard zb;

   public:
      RobotDemo() :
         stick(1),
         zb(ZomBDashboard::GetInstance(AllTCP))
      {
         GetWatchdog().SetEnabled(false);
      }

      void OperatorControl()
      {
         while (IsEnabled())
         {
            //Don't overload it!
            if (zb.CanSend())
            {
               zb.Add("taco", stick.GetY());
               zb.Add("val", zb.GetString("ival"));
               zb.Send();
            }
            Wait(0.02);
         }
      }
};
```

C++ (DBPacket workaround)

The way WPILib works, you must use this method to install ZomB if you are using DBPacket. Start by opening the Bindings folder located at *Start>All Programs>ZomB>Bindings*. Drag ZomBDashboard.h onto your project in WindRiver. Right click your project, and go to *Properties*. Select *Build Properties*, and in the *Libraries* tab, click *Add*. Select *Add full qualified library file*, and browse to your ZomB Bindings folder location (ex: *C:\Program Files\ZomB\Bindings*) and select *ZomB.a* and OK out of all the dialogs. Code example:

```
#include "WPILib.h"
#include "ZomBDashboard.h"
class RobotDemo : public SimpleRobot
{
      Joystick stick;
      ZomBDashboard zb;

   public:
      RobotDemo() :
         stick(1),
         zb(ZomBDashboard::GetInstance(DBPacket))
      {
         GetWatchdog().SetEnabled(false);
      }

      void OperatorControl()
      {
         while (IsEnabled())
         {
            //Don't overload it!
            if (zb.CanSend())
            {
               zb.Add("taco", stick.GetY());
               zb.Send();
            }
            Wait(0.02);
         }
      }
};
```

Java

To use the ZomB bindings in Java, begin by right clicking the src folder in NetBeans, and adding a new package with name *org.thecatattack.System451.Communication.Dashboard* and click Ok. Open the Bindings folder located at *Start>All Programs>ZomB>Bindings*. Drag ZomBDashboard.java and ZomBModes.java onto your newly created package. Code example:

```
package edu.wpi.first.wpilibj.templates;

import edu.wpi.first.wpilibj.*;
import org.thecatattack.System451.Communication.Dashboard.*;

public class RobotTemplate extends SimpleRobot
{

    Joystick stick = new Joystick(1);
    ZomBDashboard zomB = ZomBDashboard.getInstance(ZomBModes.AllTCP,
"10.4.51.5");

    public void operatorControl()
    {
```

```
        getWatchdog().setEnabled(false);
        while (isOperatorControl())
        {
            if (zomB.CanSend())
            {
                zomB.Add("taco", stick.getY());
                zomB.Add("val", zomB.GetString("ival"));
                zomB.Send();
            }
            Timer.delay(0.01);
        }
    }
}
```

LabVIEW

To use the ZomB bindings in LabVIEW, open the Bindings folder located at *Start>All Programs>ZomB>Bindings*. Drag ZomB.llb onto your project. All vi's are located in the ZomB folder, see C++ or Java for the order of operations.

Creating custom bindings

It is fairly easy to write a binding wrapper for a different language, simply call the exposed C functions listed below when ZomB.out is loaded on the cRIO. Java and LabVIEW use this method.

Public C functions (found in ZomBDashboard.h):

```
void ZomBDashboardInit(ZomBModes mode, char* ip);
int ZomBDashboardAdd(const char* name, const char* value);
int ZomBDashboardAddDebugVariable(const char* name, const char* v);
const char* ZomBDashboardGetString(const char* name);
void ZomBDashboardGetStringViaArg(const char* name, char* outValue);
int ZomBDashboardGetInt(const char* name);
float ZomBDashboardGetFloat(const char* name);
double ZomBDashboardGetDouble(const char* name);
int ZomBDashboardHasSpace();
int ZomBDashboardIsConnected();
int ZomBDashboardIsAnyConnected();
int ZomBDashboardCanSend();
int ZomBDashboardSend();
void ZomBDashboardResetCounter();
```

Some things worth pointing out: ZomBDashboardGetStringViaArg returns the exact same result as ZomBDashboardGetString but via a pointer as a second argument. ZomBModes is an enum that can be replaced with an int, see ZomBDashboard.h for the values.