7.4 guiNewSession Namespace Reference .......................... 14
  7.4.1 Detailed Description ........................................ 14

7.5 guiRecord Namespace Reference .............................. 14
  7.5.1 Detailed Description ........................................ 15

7.6 guiScript Namespace Reference ............................... 15
  7.6.1 Detailed Description ........................................ 15

7.7 lpard Namespace Reference .................................... 15
  7.7.1 Detailed Description ........................................ 15

7.8 lpardI2C Namespace Reference ................................ 16
  7.8.1 Detailed Description ........................................ 16

7.9 lpardXbee Namespace Reference ............................... 16
  7.9.1 Detailed Description ........................................ 16

7.10 nameValueObj Namespace Reference ......................... 16
  7.10.1 Detailed Description ....................................... 16

7.11 setup Namespace Reference .................................. 17
  7.11.1 Detailed Description ....................................... 17
  7.11.2 Variable Documentation .................................. 17
    7.11.2.1 opts .................................................. 17

7.12 utils Namespace Reference .................................. 17
  7.12.1 Detailed Description ....................................... 17

8 Class Documentation ............................................. 19

  8.1 ADXL335 Class Reference ................................... 19

  8.2 AtCommandRequest Class Reference ......................... 19
    8.2.1 Detailed Description .................................... 20
    8.2.2 Member Function Documentation ......................... 20
      8.2.2.1 clearCommandValue ................................ 20
      8.2.2.2 getFrameData ....................................... 20
      8.2.2.3 getFrameDataLength ............................... 21

  8.3 AtCommandResponse Class Reference ....................... 21
    8.3.1 Detailed Description .................................... 21
    8.3.2 Member Function Documentation ......................... 21
      8.3.2.1 getCommand ......................................... 21
      8.3.2.2 getStatus .......................................... 22
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.22</td>
<td>MagnetometerRaw Struct Reference</td>
<td>42</td>
</tr>
<tr>
<td>8.23</td>
<td>MagnetometerScaled Struct Reference</td>
<td>42</td>
</tr>
<tr>
<td>8.24</td>
<td>guiScript::Mode Class Reference</td>
<td>42</td>
</tr>
<tr>
<td>8.24.1</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>43</td>
</tr>
<tr>
<td>8.24.2</td>
<td>Member Function Documentation</td>
<td>43</td>
</tr>
<tr>
<td>8.24.3</td>
<td>Member Data Documentation</td>
<td>43</td>
</tr>
<tr>
<td>8.25</td>
<td>ModemStatusResponse Class Reference</td>
<td>44</td>
</tr>
<tr>
<td>8.26</td>
<td>nameValueObj::NameValueObj Class Reference</td>
<td>44</td>
</tr>
<tr>
<td>8.27</td>
<td>PayloadRequest Class Reference</td>
<td>45</td>
</tr>
<tr>
<td>8.27.1</td>
<td>Detailed Description</td>
<td>45</td>
</tr>
<tr>
<td>8.27.2</td>
<td>Member Function Documentation</td>
<td>45</td>
</tr>
<tr>
<td>8.28</td>
<td>RemoteAtCommandRequest Class Reference</td>
<td>46</td>
</tr>
<tr>
<td>8.28.1</td>
<td>Detailed Description</td>
<td>47</td>
</tr>
<tr>
<td>8.28.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>47</td>
</tr>
<tr>
<td>8.28.3</td>
<td>Member Function Documentation</td>
<td>47</td>
</tr>
<tr>
<td>8.29</td>
<td>RemoteAtCommandResponse Class Reference</td>
<td>48</td>
</tr>
<tr>
<td>8.29.1</td>
<td>Detailed Description</td>
<td>48</td>
</tr>
</tbody>
</table>
Chapter 1

LPARD - Lafayette Programmable Autonomous River Drone

1.1 Introduction

For the ECE492 Senior Design project, the ECE Class of 2012 is creating the Tracking and Data Fusion component of the three-year Lafayette Programmable Autonomous River Droid project. This is the API for the software and firmware.

1.2 Code Breakdown

1.2.1 Shore Software

Shore software consists of:

- The GUI
- Boat simulation model
- Freeze script to make executable
- Controller for I2C/XBee/GUI interface
- The AZEL control and sensor software

1.2.2 Boat Firmware

Boat firmware consists of:

- The main drone firmware
- Libraries for:
- XBee communication
- GPS communication
- Compass access and tilt compensation

1.3 Other

Note that compiling the PDF requires the package texlive-fonts-recommended.
Chapter 2

Bug List

File **AZELCode.h**  Main loop does not have fixed timing.
   Hard coded limits.

File **drone_firmware.pde**  Compass not tilt compensated.
   Compass iron offsets infinite loop, and have been disabled.
   Doesn't receive **XBee** packets properly. Could be a mode issue (needs API=2).
   Cannot set declination angle externally.
Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

- **BoatSim** (This package contains a single class that acts as a real time simulation of a boat) .................................................. 13
- **guiMainPage** ........................................................................... 13
- **guiManualPositioning** .......................................................... 14
- **guiNewSession** ................................................................. 14
- **guiRecord** ........................................................................... 14
- **guiScript** (This package contains two classes responsible for the creating, importing, and saving of automated script files) .... 15
- **lpard** .................................................................................. 15
- **lpardI2C** (This package contains a class that handles the interface to an AZEL Positioner through the I2C 2 PC Adapter) ........ 16
- **lpardXbee** (This package contains a class that communicates in real time with a boat using Xbee rf trancievers) ............ 16
- **nameValueObj** .................................................................... 16
- **setup** .................................................................................. 17
- **utils** (This package contains all utilities for general use in the software) .............................................................. 17
# Chapter 4

## Class Index

### 4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADXL335</td>
<td>19</td>
</tr>
<tr>
<td>lpard::interface::lpardXBee::Boat</td>
<td>22</td>
</tr>
<tr>
<td>lpard::interface::BoatSim::BoatSim</td>
<td>23</td>
</tr>
<tr>
<td>lpard::interface::BoatSimThreader::BoatSimThreader</td>
<td>27</td>
</tr>
<tr>
<td>Comp6DOF_n0m1</td>
<td>27</td>
</tr>
<tr>
<td>lpard::controller::lpardController::Controller</td>
<td>28</td>
</tr>
<tr>
<td>lpard::controller::lpardControllerThread::ControllerThread</td>
<td>30</td>
</tr>
<tr>
<td>GPS_NMEA_Class</td>
<td>31</td>
</tr>
<tr>
<td>guiAbout::GuiAbout</td>
<td>32</td>
</tr>
<tr>
<td>guiMainPage::GuiMainPage</td>
<td>33</td>
</tr>
<tr>
<td>guiManualPositioning::GuiManualPositioning</td>
<td>34</td>
</tr>
<tr>
<td>guiNewSession::GuiNewSession</td>
<td>35</td>
</tr>
<tr>
<td>guiRecord::GuiRecord</td>
<td>35</td>
</tr>
<tr>
<td>guiScript::GuiScript</td>
<td>35</td>
</tr>
<tr>
<td>HMC5883L</td>
<td>37</td>
</tr>
<tr>
<td>lpard::interface::lpardI2C::I2C2PC_MockObj</td>
<td>38</td>
</tr>
<tr>
<td>lpard::interface::lpardI2C::I2CInterface</td>
<td>38</td>
</tr>
<tr>
<td>lpard::logger::lpardLogger::Logger</td>
<td>41</td>
</tr>
<tr>
<td>MagnetometerRaw</td>
<td>42</td>
</tr>
<tr>
<td>MagnetometerScaled</td>
<td>42</td>
</tr>
<tr>
<td>guiScript::Mode</td>
<td>42</td>
</tr>
<tr>
<td>nameValueObj::NameValueObj</td>
<td>44</td>
</tr>
<tr>
<td>GUI_about_ui::Ui_about</td>
<td>60</td>
</tr>
<tr>
<td>GUI_mainPageReturn_ui::Ui_guIMain</td>
<td>61</td>
</tr>
<tr>
<td>GUI_manualPositioning_ui::Ui_manualPositioning</td>
<td>62</td>
</tr>
<tr>
<td>GUI_newSession_ui::Ui_newSession</td>
<td>62</td>
</tr>
<tr>
<td>GUI_record_ui::Ui_recordData</td>
<td>63</td>
</tr>
<tr>
<td>GUI_script_ui::Ui_Script</td>
<td>64</td>
</tr>
<tr>
<td>XBee</td>
<td>65</td>
</tr>
<tr>
<td>Class Name</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>XBeeAddress</td>
<td>67</td>
</tr>
<tr>
<td>XBeeAddress64</td>
<td>67</td>
</tr>
<tr>
<td>lpard::interface::lpardXbee::XbeeInterface</td>
<td>68</td>
</tr>
<tr>
<td>XBeeRequest</td>
<td>69</td>
</tr>
<tr>
<td>AtCommandRequest</td>
<td>19</td>
</tr>
<tr>
<td>RemoteAtCommandRequest</td>
<td>46</td>
</tr>
<tr>
<td>PayloadRequest</td>
<td>45</td>
</tr>
<tr>
<td>Tx16Request</td>
<td>56</td>
</tr>
<tr>
<td>Tx64Request</td>
<td>58</td>
</tr>
<tr>
<td>ZBTxRequest</td>
<td>78</td>
</tr>
<tr>
<td>XBeeResponse</td>
<td>71</td>
</tr>
<tr>
<td>FrameIdResponse</td>
<td>30</td>
</tr>
<tr>
<td>AtCommandResponse</td>
<td>21</td>
</tr>
<tr>
<td>RemoteAtCommandResponse</td>
<td>48</td>
</tr>
<tr>
<td>TxStatusResponse</td>
<td>59</td>
</tr>
<tr>
<td>ZBTxStatusResponse</td>
<td>80</td>
</tr>
<tr>
<td>ModemStatusResponse</td>
<td>44</td>
</tr>
<tr>
<td>RxDataResponse</td>
<td>52</td>
</tr>
<tr>
<td>RxResponse</td>
<td>55</td>
</tr>
<tr>
<td>Rx16Response</td>
<td>50</td>
</tr>
<tr>
<td>Rx64Response</td>
<td>52</td>
</tr>
<tr>
<td>RxIoSampleBaseResponse</td>
<td>54</td>
</tr>
<tr>
<td>Rx16IoSampleResponse</td>
<td>50</td>
</tr>
<tr>
<td>Rx64IoSampleResponse</td>
<td>51</td>
</tr>
<tr>
<td>ZBRxResponse</td>
<td>77</td>
</tr>
<tr>
<td>ZBRxIoSampleResponse</td>
<td>76</td>
</tr>
</tbody>
</table>
Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

- ADXL335 ................................................................. 19
- AtCommandRequest .................................................. 19
- AtCommandResponse .................................................. 19
- lpard::interface::lpardXBee::Boat .................................. 22
- lpard::interface::BoatSim::BoatSim .................................. 23
- lpard::interface::BoatSimThreader::BoatSimThreader .............. 27
- Comp6DOF_n0m1 .......................................................... 27
- lpard::controller::lpardController::Controller .................. 28
- lpard::controller::lpardControllerThread::ControllerThread ...... 30
- FrameIdResponse ...................................................... 30
- GPS_NMEA_Class ....................................................... 31
- guiAbout::GuiAbout (Controlling class for the About dialog ) .... 32
- guiMainPage::GuiMainPage .......................................... 33
- guiNewSession::GuiNewSession ..................................... 35
- guiRecord::GuiRecord ................................................. 35
- guiScript::GuiScript (Responsible for the creating, reading and writing of navigation scripts ) .................. 35
- HMC5883L .............................................................. 37
- lpard::interface::lpardI2C::I2C2PC_MockObj ...................... 38
- lpard::interface::lpardI2C::I2CInterface .......................... 38
- lpard::logger::lpardLogger::Logger ................................ 41
- MagnetometerRaw ..................................................... 42
- MagnetometerScaled .................................................. 42
- guiScript::Mode ....................................................... 42
- ModemStatusResponse ............................................... 44
- nameValueObj::NameValueObj ....................................... 44
- PayloadRequest ....................................................... 45
- RemoteAtCommandRequest ............................................ 46
<table>
<thead>
<tr>
<th>Class Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoteAtCommandResponse</td>
<td>48</td>
</tr>
<tr>
<td>Rx16IoSampleResponse</td>
<td>50</td>
</tr>
<tr>
<td>Rx16Response</td>
<td>50</td>
</tr>
<tr>
<td>Rx64IoSampleResponse</td>
<td>51</td>
</tr>
<tr>
<td>Rx64Response</td>
<td>52</td>
</tr>
<tr>
<td>RxDataResponse</td>
<td>52</td>
</tr>
<tr>
<td>RxIoSampleBaseResponse</td>
<td>54</td>
</tr>
<tr>
<td>RxResponse</td>
<td>55</td>
</tr>
<tr>
<td>Tx16Request</td>
<td>56</td>
</tr>
<tr>
<td>Tx64Request</td>
<td>58</td>
</tr>
<tr>
<td>TxStatusResponse</td>
<td>59</td>
</tr>
<tr>
<td>GUI_about_ui::Ui_about</td>
<td>60</td>
</tr>
<tr>
<td>GUI_mainPageReturn_ui::Ui_guiMain</td>
<td>61</td>
</tr>
<tr>
<td>GUI_manualPositioning_ui::Ui_manualPositioning</td>
<td>62</td>
</tr>
<tr>
<td>GUI_newSession_ui::Ui_newSession</td>
<td>62</td>
</tr>
<tr>
<td>GUI_record_ui::Ui_recordData</td>
<td>63</td>
</tr>
<tr>
<td>GUI_script_ui::Ui_Script</td>
<td>64</td>
</tr>
<tr>
<td>XBee</td>
<td>65</td>
</tr>
<tr>
<td>XBeeAddress</td>
<td>67</td>
</tr>
<tr>
<td>XBeeAddress64</td>
<td>67</td>
</tr>
<tr>
<td>lpard::interface::lpardXBee::XbeeInterface</td>
<td>68</td>
</tr>
<tr>
<td>XBeeRequest</td>
<td>69</td>
</tr>
<tr>
<td>XBeeResponse</td>
<td>71</td>
</tr>
<tr>
<td>ZBRxIoSampleResponse</td>
<td>76</td>
</tr>
<tr>
<td>ZBRxResponse</td>
<td>77</td>
</tr>
<tr>
<td>ZBTxRequest</td>
<td>78</td>
</tr>
<tr>
<td>ZBTxStatusResponse</td>
<td>80</td>
</tr>
</tbody>
</table>
Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

- boat-systems/drone_firmware/drone_firmware.pde (Main Arduino code file for boat firmware) 81
- boat-systems/drone_firmware/libraries/ADXL35/ADXL335.h ??
- boat-systems/drone_firmware/libraries/Comp6DOF/Comp6DOF_n0m1.h ??
- boat-systems/drone_firmware/libraries/GPS_NMEA/GPS_NMEA.h ??
- boat-systems/drone_firmware/libraries/HMC5883L/HMC5883L.h ??
- boat-systems/drone_firmware/libraries/lpard_RF/lpard_rf.h ??
- boat-systems/drone_firmware/libraries/XBee/XBee.h ??
- shore-systems/AZEL/AZELCode/AZELCode.h (Firmware for the AZ/EL Positioner) 85
- shore-systems/AZEL/AZELCode/i2c.h ??
- shore-systems/AZEL/AZELCode/pwm.h ??
- shore-systems/AZEL/AZELCode/tracking.h ??
- shore-systems/AZEL/AZELCode/ultrasound.h ??
Chapter 7

Namespace Documentation

7.1 BoatSim Namespace Reference

This package contains a single class that acts as a real time simulation of a boat.

7.1.1 Detailed Description

This package contains a single class that acts as a real time simulation of a boat. It extends the Thread class so the program can be run as a separate thread from the main GUI thread.

Author

Aaron Springut

Date

2012

7.2 guiMainPage Namespace Reference

Classes

• class GuiMainPage

Functions

• def main
7.2.1 Detailed Description

Created on Mar 8, 2012
@author: This PC

7.3 guiManualPositioning Namespace Reference

Classes
- class GuiManualPositioning

Functions
- def main

7.3.1 Detailed Description

Created on Apr 12, 2012
@author: Charles Thomas

7.4 guiNewSession Namespace Reference

Classes
- class GuiNewSession

Functions
- def main

7.4.1 Detailed Description

Created on Apr 4, 2012
@author: Lauren Elizabeth

7.5 guiRecord Namespace Reference

Classes
- class GuiRecord

7.5.1 Detailed Description

Created on Apr 4, 2012
@author: Lauren Elizabeth
7.6 guiScript Namespace Reference

Functions

• def main

7.5.1 Detailed Description

Created on Mar 22, 2012
@author: This PC

7.6 guiScript Namespace Reference

This package contains two classes responsible for the creating, importing, and saving of automated script files.

Classes

• class GuiScript

    The GuiScript class is responsible for the creating, reading and writing of navigation scripts.

• class Mode

Functions

• def main

7.6.1 Detailed Description

This package contains two classes responsible for the creating, importing, and saving of automated script files. The classes are GuiScript and Mode.

Date

: Created on Mar 22, 2012

Author

: Michael Rupolo

7.7 lpard Namespace Reference

7.7.1 Detailed Description

print "** Loading Packages **

Loading logger:"
import lpard.logger
print "Loading gui:"
import lpard.gui

7.8 IpardI2C Namespace Reference

This package contains a class that handles the interface to an AZEL Positioner through the I2C 2 PC Adapter.

7.8.1 Detailed Description

This package contains a class that handles the interface to an AZEL Positioner through the I2C 2 PC Adapter. This class uses a configuration file to take commands and formulate I2C transmissions for them. There is also a commandline interface utility that can be called directly from this package as well as a mock object for the I2C 2 Pc adapter for testing.

7.9 IpardXbee Namespace Reference

This package contains a class that communicates in real time with a boat using Xbee rf tranceivers.

7.9.1 Detailed Description

This package contains a class that communicates in real time with a boat using Xbee rf tranceivers. The Boat class is a thread which shuffles data from the Xbee to two different queues, received packets and packets to send. The XbeeInterface class then used to read and write the packets to the queues.

7.10 nameValueObj Namespace Reference

Classes

• class NameValueObj

7.10.1 Detailed Description

Created on Apr 5, 2012
@author: Lauren Elizabeth
7.11 setup Namespace Reference

Variables

- list `data_files = []`
- list `f1 = sys.path[0]`
- string `f2 = './config'`
- list `dirpath = sys.path[0]`
- dictionary `opts`
- string `description = "LPARD-TDF-2012 Software"
- string `long_description = ""
- string `version = "0.1"
- list `windows = [{'script': "lpard_main.py", "dest_base": "lpard"]`
- options = opts,
- tuple `zf = zipfile.ZipFile('LPARD-TDF-2012 Software.zip', mode='w')`

7.11.1 Detailed Description

7.11.2 Variable Documentation

7.11.2.1 dictionary setup::opts

Initial value:

```python
{'py2exe': {
  # 'bundle_files': 2,
  'includes': "sip",
  'dist_dir': dirpath,
  'excludes': ['pdb', "unittest"],
  'dll_excludes': ['w9xpopen.exe']})
```

7.12 utils Namespace Reference

This package contains all utilities for general use in the software.

7.12.1 Detailed Description

This package contains all utilities for general use in the software.
Chapter 8

Class Documentation

8.1 ADXL335 Class Reference

Public Member Functions

- ADXL335(int pin_x, int pin_y, int pin_z, float aref)
- void setThreshold(float deadzone)
- boolean getFreefall()
- float getX()
- float getY()
- float getZ()
- float getRho()
- float getPhi()
- float getTheta()
- void update()

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/ADXL335/ADXL335.h
- boat-systems/drone_firmware/libraries/ADXL335/ADXL335.cpp

8.2 AtCommandRequest Class Reference

#include <XBee.h>

Inheritance diagram for AtCommandRequest:
Public Member Functions

- `AtCommandRequest (uint8_t *command)`
- `AtCommandRequest (uint8_t *command, uint8_t *commandValue, uint8_t commandValueLength)`
- `uint8_t getFrameData (uint8_t pos)`
- `uint8_t getFrameDataLength ()`
- `uint8_t *getCommand ()`
- `void setCommand (uint8_t *command)`
- `uint8_t *getCommandValue ()`
- `void setCommandValue (uint8_t *command)`
- `uint8_t getCommandValueLength ()`
- `void setCommandValueLength (uint8_t length)`
- `void clearCommandValue ()`

8.2.1 Detailed Description

Represents an AT Command TX packet. The command is used to configure the serially connected XBee radio.

8.2.2 Member Function Documentation

8.2.2.1 void AtCommandRequest::clearCommandValue ( )

Clears the optional commandValue and commandValueLength so that a query may be sent.

8.2.2.2 uint8_t AtCommandRequest::getFrameData ( uint8_t pos ) [virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum. Note: Unlike Digi’s definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implements `XBeeRequest`.

Reimplemented in `RemoteAtCommandRequest`.
8.3 AtCommandResponse Class Reference

8.2.2.3 uint8_t AtCommandRequest::getFrameDataLength () [virtual]

Returns the size of the api frame (not including frame id or api id or checksum).
Implements XBeeRequest.
Reimplemented in RemoteAtCommandRequest.
The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.3 AtCommandResponse Class Reference

#include <XBee.h>

Inheritance diagram for AtCommandResponse:

```
AtCommandResponse
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>
| FrameIdResponse
|               |
| XBeeResponse
|               |
|               |
|               |
| RemoteAtCommandResponse
```

Public Member Functions

- `uint8_t * getCommand ()`
- `uint8_t getStatus ()`
- `uint8_t * getValue ()`
- `uint8_t getValueLength ()`
- `bool isOk ()`

8.3.1 Detailed Description

Represents an AT Command RX packet

8.3.2 Member Function Documentation

8.3.2.1 uint8_t * AtCommandResponse::getCommand ()

Returns an array containing the two character command
8.3.2.2 uint8_t AtCommandResponse::getStatus ( )

Returns the command status code. Zero represents a successful command
Reimplemented in RemoteAtCommandResponse.

8.3.2.3 uint8_t * AtCommandResponse::getValue ( )

Returns an array containing the command value. This is only applicable to query com-
mands.
Reimplemented in RemoteAtCommandResponse.

8.3.2.4 uint8_t AtCommandResponse::getValueLength ( )

Returns the length of the command value array.
Reimplemented in RemoteAtCommandResponse.

8.3.2.5 bool AtCommandResponse::isOk ( )

Returns true if status equals AT_OK
Reimplemented in RemoteAtCommandResponse.

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.4 lpard::interface::lpardXBee::Boat Class Reference

Inherits Thread.

Public Member Functions

- def __init__

  Constructor for the Boat thread.

- def run

  This method runs the thread.
## Public Attributes

- inQ
- outQ
- zigbee
- alive

### Constructor & Destructor Documentation

#### Constructor & Destructor Documentation

8.4.1.1 def lpard::interface::lpardXBee::Boat::init ( self, inQ, outQ, zigbee )

Constructor for the Boat thread.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
</tr>
<tr>
<td>inQ</td>
</tr>
<tr>
<td>outQ</td>
</tr>
<tr>
<td>zigbee</td>
</tr>
</tbody>
</table>

#### Member Function Documentation

8.4.2.1 def lpard::interface::lpardXBee::Boat::run ( self )

This method runs the thread.

This method overrides the Thread.run() method. When the thread is started this method is called. In a loop, checks inQ for new messages to send and puts new messages from the xbee into outQ.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
</tr>
</tbody>
</table>

The documentation for this class was generated from the following file:

- shore-systems/lpard/interface/lpardXBee.py

---

### 8.5 lpard::interface::BoatSim::BoatSim Class Reference

#### Public Member Functions

- def __init__
  Constructor for the simulation.
- def run
  This method runs the simulation.
- def command
This method processes any input strings and then call the correct method to deal with them, depending on the header.

• def boatCommand
  This method processes all methods with a "BC" or boat command header.

• def update
  This method updates the boat model states through the transfer function described in the transition matrix.

• def getMessages
  This method checks if any messages have been put into the input queue and calls the command() if a message is present.

• def boatStatusMessage
  This method places the boat status message in the output queue.

• def boatCommandResponse
  This method send a command response.

Public Attributes

• inQ
• outQ
• startTime
• xPos
• yPos
• zPos
• theta
• fMov
• sMov
• oMov
• fMax
• sMax
• oMax
• forwardGain
• sideGain
• thetaGain
• xDest
• yDest
• mode
• calibrate
• fault
• timeStep
8.5.1 Constructor & Destructor Documentation

8.5.1.1 def lpard::interface::BoatSim::BoatSim::__init__( self, inQ, outQ )

Constructor for the simulation.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class instance.</td>
</tr>
<tr>
<td>inQ</td>
<td>A synchronized queue for passing commands to the simulation.</td>
</tr>
<tr>
<td>outQ</td>
<td>A synchronized queue for receiving commands from the GUI.</td>
</tr>
</tbody>
</table>

8.5.2 Member Function Documentation

8.5.2.1 def lpard::interface::BoatSim::BoatSim::boatCommand ( self, commandString )

This method processes all methods with a "BC" or boat command header.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class.</td>
</tr>
<tr>
<td>commandString</td>
<td>The message that has been sent to the boat.</td>
</tr>
</tbody>
</table>

8.5.2.2 def lpard::interface::BoatSim::BoatSim::boatCommandResponse ( self, queriedVariable )

This method send a command response.

It is normally called when the boatCommand() method determines that a response is neccessary.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class.</td>
</tr>
<tr>
<td>queriedVariable</td>
<td>The boat state that is being queried. This can be any one of the variables for command response specified in the boat protocol document.</td>
</tr>
</tbody>
</table>

8.5.2.3 def lpard::interface::BoatSim::BoatSim::boatStatusMessage ( self )

This method places the boat status message in the output queue.

This message gets called every 1000 timesteps in the run method. This corresponds to about 1 second.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class.</td>
</tr>
</tbody>
</table>
8.5.2.4 def lpard::interface::BoatSim::BoatSim::command ( self, commandString )

This method processes any input strings and then call the correct method to deal with them, depending on the header.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class.</td>
</tr>
<tr>
<td>commandString</td>
<td>The message that has been sent to the boat.</td>
</tr>
</tbody>
</table>

8.5.2.5 def lpard::interface::BoatSim::BoatSim::getMessages ( self )

This method checks if any messages have been put into the input queue and calls the command() if a message is present.

This method should be called consistantly during the run() method.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class.</td>
</tr>
</tbody>
</table>

8.5.2.6 def lpard::interface::BoatSim::BoatSim::run ( self )

This method runs the simulation.

This method overrides the Thread.run() method. When the thread is started this method is called. It continually loops and updates the transfer model and deals with any input messages.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class.</td>
</tr>
</tbody>
</table>

8.5.2.7 def lpard::interface::BoatSim::BoatSim::update ( self )

This method updates the boat model states through the transfer function described in the transition matrix.

This matrix can be found in the boat model analysis. Currently this is not implemented correctly and is instead replaced with a simplified version.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class.</td>
</tr>
</tbody>
</table>

The documentation for this class was generated from the following file:

- shore-systems/lpard/interface/BoatSim.py
8.6 Ipard::interface::BoatSimThreader::BoatSimThreader Class Reference

Public Member Functions

- def __init__
- def sendMessage
- def getMessage

Public Attributes

- inQ
- outQ

The documentation for this class was generated from the following file:

- shore-systems/lpard/interface/BoatSimThreader.py

8.7 Comp6DOF_n0m1 Class Reference

Public Member Functions

- void compCompass (int xMagAxis, int yMagAxis, int zMagAxis, int xAccel, int yAccel, int zAccel, boolean lowpass)
- int roll ()
- int pitch ()
- int yaw ()
- float rollf ()
- float pitchf ()
- float yawf ()
- int xAxisComp ()
- int yAxisComp ()
- int zAxisComp ()
- int xHardOff ()
- int yHardOff ()
- int zHardOff ()
- boolean deviantSpread (int XAxis, int YAxis, int ZAxis)
- boolean calOffsets ()
- int atan2Int (int y, int x)

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/Comp6DOF/Comp6DOF_n0m1.h
- boat-systems/drone_firmware/libraries/Comp6DOF/Comp6DOF_n0m1.cpp

Generated on Tue May 8 2012 13:27:22 for LPARD-TDF-2012 by Doxygen
8.8 lpard::controller::lpardController::Controller Class Reference

Public Member Functions

- `def __init__`
  Constructor, initializes the logger, also populates default sensor values and default "listOfValues" values.

- `def getDegreesPerPwm`
  Calculate values required for degree calculations.

- `def getDegrees`
  Returns the azimuth angle and elevation angle in degrees based off the passed PWM values.

- `def getPWM`
  Returns the azimuth angle and elevation angle in PWM based off of the passed degree values.

- `def initSerialInterface`
  Ensures the AZEL(s) are connected via USB - Finds the AZEL(s) and start the USB communications.

- `def manualAZELPositioning`
- `def autoAZELPositioning`
- `def recieveUltrasonic`
- `def transmiteUltrasonic`
- `def logCurrentAZELStatus`
- `def processRawAZEL`
- `def processAZELData`
- `def startSearch`
- `def contSearch`
- `def moveSensorsLeft`
- `def moveSensorsRight`
- `def moveSensorsUp`
- `def moveSensorsDown`
- `def initCalibrate`
- `def updateData`
- `def translateSensorPos`
- `def setSensorPos`
- `def updateSensorPos`
- `def calSOS`
- `def shutdown`

Public Attributes

- `prev_sensor`
- `xValue`
- `yValue`
- `thetaValue`
8.8 lpard::controller::lpardController::Controller Class Reference

- I2C_port
- r1
- r2
- baseline
- sos

Static Public Attributes

- logger = None
- dictionary I2C_port = {}
- ser_itrfc = None
- xbee_itrfc = None
- dictionary system_values = {}
- list listOfValues = []
- i2c_config = None
- xbee_config = None
- mode = REAL_TIME
- float r1 = 1.0
- float r2 = 1.0
- int baseline = 1
- float sos = 0.344

8.8.1 Constructor & Destructor Documentation

8.8.1.1 def lpard::controller::lpardController::Controller::__init__( self, sensors = [1, 2] )

Constructor, initializes the logger, also populates default sensor values and default "listOfValues" values.

Parameters

<table>
<thead>
<tr>
<th>num_sensors</th>
<th>List of AZEL sensors (Defaults to [1,2])</th>
</tr>
</thead>
</table>

8.8.2 Member Function Documentation

8.8.2.1 def lpard::controller::lpardController::Controller::getDegrees( self, index, azimuth, elevation )

Returns the azimuth angle and elevation angle in degrees based off of the passed PWM values.

Parameters

<table>
<thead>
<tr>
<th>index</th>
<th>The index of the sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>azimuth</td>
<td>The PWM value for the horizontal axis</td>
</tr>
<tr>
<td>elevation</td>
<td>The PWM value for the vertical axis</td>
</tr>
</tbody>
</table>

Generated on Tue May 8 2012 13:27:22 for LPARD-TDF-2012 by Doxygen
8.8.2.2 def lpard::controller::lpardController::Controller::getDegreesPerPwm (self, pwm_left, pwm_right, pwm_up, pwm_down, deg_left, deg_right, deg_up, deg_down)

Calculate values required for degree calculations.
Data is stored in AZEL_dgr_per_pwm

**Parameters**

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwm_left</td>
<td>The PWM value for soft left limit</td>
</tr>
<tr>
<td>pwm_right</td>
<td>The PWM value for soft right limit</td>
</tr>
<tr>
<td>pwm_up</td>
<td>The PWM value for soft up limit</td>
</tr>
<tr>
<td>pwm_down</td>
<td>The PWM value for soft down limit</td>
</tr>
<tr>
<td>deg_left</td>
<td>The degree value for soft left limit</td>
</tr>
<tr>
<td>deg_right</td>
<td>The degree value for soft right limit</td>
</tr>
<tr>
<td>deg_up</td>
<td>The degree value for soft up limit</td>
</tr>
<tr>
<td>deg_down</td>
<td>The degree value for soft down limit</td>
</tr>
</tbody>
</table>

The documentation for this class was generated from the following file:

- shore-systems/lpard/controller/lpardController.py

8.9 lpard::controller::lpardControllerThread::ControllerThread Class Reference

**Public Member Functions**

- def run

**Public Attributes**

- controller

The documentation for this class was generated from the following file:

- shore-systems/lpard/controller/lpardControllerThread.py

8.10 FrameldResponse Class Reference

```
#include <XBee.h>
```

Inheritance diagram for FrameldResponse:
Public Member Functions

- uint8_t getFrameId ()

8.10.1 Detailed Description

This class is extended by all Responses that include a frame id

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.11 GPS_NMEA_Class Class Reference

Public Member Functions

- void Init ()
- void Read ()

Public Attributes

- long Time
- long Latitude
- long Longitude
- long Altitude
- long Ground_Speed
- long Speed_3d
- long Ground_Course
- uint8_t Type
- uint8_t NumSats
- uint8_t Fix
- uint8_t Quality
• `uint8_t` `NewData`
• `uint8_t` `PrintErrors`
• `int` `HDOP`

The documentation for this class was generated from the following files:

• `boat-systems/drone_firmware/libraries/GPS_NMEA/GPS_NMEA.h`
• `boat-systems/drone_firmware/libraries/GPS_NMEA/GPS_NMEA.cpp`

8.12 `guiAbout::GuiAbout` Class Reference

Controlling class for the About dialog.

Public Member Functions

• `def __init__`
  Constructor for the about dialog.
• `def showGui`
  Make the dialog visible.

Public Attributes

• `ui`
  The UI object for the about dialog.

8.12.1 Detailed Description

Controlling class for the About dialog.

Author

Mike Rupolo

Date

22 Mar 2012

8.12.2 Constructor & Destructor Documentation

8.12.2.1 `def guiAbout::GuiAbout::__init__ ( self )`

Constructor for the about dialog.
8.13 guiMainPage::GuiMainPage Class Reference

8.12.3 Member Function Documentation

8.12.3.1 def guiAbout::GuiAbout::showGui ( self )
Make the dialog visible.

8.12.4 Member Data Documentation

8.12.4.1 guiAbout::GuiAbout::ui
The UI object for the about dialog.
The documentation for this class was generated from the following file:

• shore-systems/lpard/gui/guiAbout.py

8.13 guiMainPage::GuiMainPage Class Reference

Public Member Functions

• def __init__
• def showGui

Public Attributes

• ui
• searchTimer
• updateTimer
• controller
• record
• script
• newSession
• about
• boatHeight
• boatWidth
• boat
• scene
• frameHeight
• frameWidth
• divisionX
• gridX
• divisionY
• gridY
• x
• y
Testing Setting up Tables ####.

The documentation for this class was generated from the following file:

- shore-systems/lpard/gui/guiMainPage.py

### guiManualPositioning::GuiManualPositioning Class Reference

#### Public Member Functions

- def __init__
- def showGui

#### Public Attributes

- controller
- dTimer
- ui

The documentation for this class was generated from the following file:

- shore-systems/lpard/gui/guiManualPositioning.py
8.15 guiNewSession::GuiNewSession Class Reference

Public Member Functions

- def __init__
- def showGui

Public Attributes

- controller
- dTimer
- uTimer
- ui
- manual

The documentation for this class was generated from the following file:

- shore-systems/lpard/gui/guiNewSession.py

8.16 guiRecord::GuiRecord Class Reference

Public Member Functions

- def __init__
- def showGui

Public Attributes

- ui
- controller

The documentation for this class was generated from the following file:

- shore-systems/lpard/gui/guiRecord.py

8.17 guiScript::GuiScript Class Reference

The GuiScript class is responsible for the creating, reading and writing of navigation scripts.
Public Member Functions

- def __init__
  Constructor for the scripting class.
- def keyPressEvent
  This method handles key press events.
- def showGui
  Method to show the window The method will initially show the GuiScript dialog box.

Public Attributes

- controller
  The I2C controller.
- ui
  The UI object being controlled.
- scriptList
  List holding the script modes.
- i
  ...
- file_opt
  Holds options for opening or saving a file.

8.17.1 Detailed Description

The GuiScript class is responsible for the creating, reading and writing of navigation scripts.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 def guiScript::GuiScript::__init__( self, controller )

Constructor for the scripting class.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class instance.</td>
</tr>
<tr>
<td>controller</td>
<td>Underlying handler for commands external to lpardGUI</td>
</tr>
</tbody>
</table>

8.17.3 Member Function Documentation

8.17.3.1 def guiScript::GuiScript::keyPressEvent( self, e )

This method handles key press events.
This method handles any key press events while the window is open. As of now its only function is to check for the 'delete' key. Can be expanded using 'if statements' to encompass other keys if desired.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class.</td>
</tr>
<tr>
<td>e</td>
<td>The key event which occurred</td>
</tr>
</tbody>
</table>

8.17.3.2 `def guiScript::GuiScript::showGui (` *self `)

Method to show the window The method will initially show the `GuiScript` dialog box.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class.</td>
</tr>
</tbody>
</table>

8.17.4 Member Data Documentation

8.17.4.1 `guiScript::GuiScript::controller`

The I2C controller.

8.17.4.2 `guiScript::GuiScript::scriptList`

List holding the script modes.

The documentation for this class was generated from the following file:

- shore-systems/lpard/gui/guiScript.py

8.18 HMC5883L Class Reference

**Public Member Functions**

- `MagnetometerRaw` `ReadRawAxis` ()
- `MagnetometerScaled` `ReadScaledAxis` ()
- int `SetMeasurementMode` (uint8_t mode)
- int `SetScale` (float gauss)
- char * `GetErrorText` (int errorCode)

**Protected Member Functions**

- void `Write` (int address, int byte)
- uint8_t * `Read` (int address, int length)
The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/HMC5883L/HMC5883L.h
- boat-systems/drone_firmware/libraries/HMC5883L/HMC5883L.cpp

## 8.19 lpard::interface::lpardI2C::I2C2PC MockObj Class Reference

### Public Member Functions

- `def __init__`
- `def write`
- `def read`
- `def close`

### Public Attributes

- `verbose`
- `cmds`
- `last_cmd`
- `timeout`

The documentation for this class was generated from the following file:

- shore-systems/lpard/interface/lpardI2C.py

## 8.20 lpard::interface::lpardI2C::I2CInterface Class Reference

### Public Member Functions

- `def __init__`
- `def __del__`
- `def status`
  
  Gets sensor mode info.
- `def data`
  
  Tells sensor to ping and then gets relevent control data.
- `def AZEL_Mode`
  
  Toggles AZ/EL Positioner mode between Manual and Tracking.
- `def US_Mode`
  
  Toggles Ultrasonic sensor mode between Receive and Transmit.
- `def changeAZ`
  
  Changes the Azimuth positon to the value of pwmVal.
- `def changeEL`
  
  Changes the Elevation positon to the value of pwmVal.
Public Attributes

- i2c2pc
- config
- verbose

Static Public Attributes

- sensorData = data
- sensorAZEL_Mode = AZEL_Mode
- sensorUS_Mode = US_Mode
- sensorStatus = status

8.20.1 Detailed Description

I2CInterface handles the interface to an AZEL Positioner through the
I2C 2 PC Adapter. This class uses a configuration file to take commands and
formulate I2C transmissions for them.

Compatible with firmware version 1.02

Bugs:
- No firmware checks for written PWM values (?)

8.20.2 Constructor & Destructor Documentation

8.20.2.1 def lpard::interface::lpardI2C::I2CInterface::

Constructor.

Parameters

self | The pointer to the class instance.
I2C_port | A PySerial object which interfaces with the I2C 2 PC adapter. Default baud
rate should be 57600. Use utils.find_ports to get the PySerial object for the
I2C port.
protocolConfig | A ConfigParser object which has been loaded with the appropriate configu-
ration file.

Params:
I2C_port - a PySerial object which interfaces with the I2C 2 PC adapter. Default baud rate should be 57600. Use utils.find_ports to get the PySerial object for the I2C port.
protocolConfig - a ConfigParser object which has been loaded with the appropriate configuration file.

8.20.3 Member Function Documentation
8.20.3.1 def lpard::interface::lpardI2C::I2CInterface::AZEL_Mode ( self, sensorID )
Toggles AZ/EL Positioner mode between Manual and Tracking.

8.20.3.2 def lpard::interface::lpardI2C::I2CInterface::changeAZ ( self, sensorID, pwmVal )
Changes the Azimuth positon to the value of pwmVal.
Limited on firmware from 800-2200.

8.20.3.3 def lpard::interface::lpardI2C::I2CInterface::changeEL ( self, sensorID, pwmVal )
Changes the Elevation positon to the value of pwmVal.
Limited on firmware from 600-1600.

8.20.3.4 def lpard::interface::lpardI2C::I2CInterface::data ( self, sensorID )
Tells sensor to ping and then gets relevent control data.
Data format is specified in config

Tells sensor to ping and then gets relevent control data.
Data format is as follows:
   Azimuth: PWM freq. in us
   Elevation: PWM freq. in us
   Time: Round trip ping travel time in us

8.20.3.5 def lpard::interface::lpardI2C::I2CInterface::status ( self, sensorID )
Gets sensor mode info.
Returns a dict based on config.

Gets sensor mode info. Returns a dict based on config.
8.2.3.6 \texttt{lpard::interface::lpardI2C::I2CInterface::US\textunderscore Mode (self, sensorID )}

Toggles Ultrasonic sensor mode between Receive and Transmit.

The documentation for this class was generated from the following file:

- shore-systems/lpard/interface/lpardI2C.py

8.21 \texttt{lpard::logger::lpardLogger::Logger Class Reference}

Public Member Functions

- \texttt{def \_\_init\_}
- \texttt{def printTime}
- \texttt{def printDate}
- \texttt{def ensureDir}
- \texttt{def addLog}
- \texttt{def write}

Public Attributes

- ifile
- dictFiles
- echo\textunderscore out
- log\textunderscore echo
- session\textunderscore time

8.21.1 Detailed Description

The Logger class is used by the LPARD system to log various system and sensor output.

All logs can be found in the Logs directory. Each log element also has its own directory within the Logs directory i.e. the system log is found in the directory "Logs/system/"

Finally each new session of logging (each time the program starts) creates a new file of the format: "Logs/<log element name>/<log element name> - <date> <time>"

Methods:

- \texttt{addLog(name, fields) - Adds a new object to be logged}
  name - Takes the name of the new log element
  fields - The name of the separate fields to be logged (e.g for logging position the fields would be x, y)

- \texttt{write(name, data) - Writes given data to the log element}
  name - The name of the log element to log to
  data - The data to be written (e.g for logging position: write(‘position’, 120, 30) NOTE: Can add as many data parameters as necessary)

The documentation for this class was generated from the following file:

- shore-systems/lpard/logger/lpardLogger.py

Generated on Tue May 8 2012 13:27:22 for LPARD-TDF-2012 by Doxygen
8.22 MagnetometerRaw Struct Reference

Public Attributes

- int XAxis
- int YAxis
- int ZAxis

The documentation for this struct was generated from the following file:

- boat-systems/drone_firmware/libraries/HMC5883L/HMC5883L.h

8.23 MagnetometerScaled Struct Reference

Public Attributes

- float XAxis
- float YAxis
- float ZAxis

The documentation for this struct was generated from the following file:

- boat-systems/drone_firmware/libraries/HMC5883L/HMC5883L.h

8.24 guiScript::Mode Class Reference

Public Member Functions

- def __init__
  Constructor for the Mode class.
- def __repr__
  Overridden toString() equivalent.

Public Attributes

- mode
  Identifier for the command mode.
- x
  Either the x-component for NAVIGATE or the HOLD duration.
- y
  Either the y-component for NAVIGATE or None if HOLD.
- record
  The indicator for whether we're recording command execution data.
8.24 guiScript::Mode Class Reference

8.24.1 Constructor & Destructor Documentation

8.24.1.1 def guiScript::Mode::__init__ ( self, mode, x, y, record )

Constructor for the Mode class.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>The pointer to the class instance.</td>
</tr>
<tr>
<td>mode</td>
<td>The identifier of the command mode.</td>
</tr>
<tr>
<td>x</td>
<td>The desired x-component of navigation in NAVIGATE mode or the desired duration of holding in HOLD mode.</td>
</tr>
<tr>
<td>y</td>
<td>The desired y-component of navigation in NAVIGATE mode or NULL in HOLD mode.</td>
</tr>
<tr>
<td>record</td>
<td>The indicator for whether or not to record all data during the execution of the desired command.</td>
</tr>
</tbody>
</table>

8.24.2 Member Function Documentation

8.24.2.1 def guiScript::Mode::__repr__ ( self )

Overridden toString() equivalent.

Uses the internal _toString method.

8.24.3 Member Data Documentation

8.24.3.1 guiScript::Mode::mode

Identifier for the command mode.

8.24.3.2 guiScript::Mode::record

The indicator for whether we're recording command execution data.

8.24.3.3 guiScript::Mode::x

Either the x-component for NAVIGATE or the HOLD duration.

8.24.3.4 guiScript::Mode::y

Either the y-component for NAVIGATE or None if HOLD.

The documentation for this class was generated from the following file:

- shore-systems/lpard/gui/guiScript.py
8.25 ModemStatusResponse Class Reference

#include <XBee.h>

Inheritance diagram for ModemStatusResponse:

```
 XBeeResponse
    
 ModemStatusResponse
```

Public Member Functions

- `uint8_t getStatus()`

8.25.1 Detailed Description

Represents a Modem Status RX packet

The documentation for this class was generated from the following files:

- `boat-systems/drone_firmware/libraries/XBee/XBee.h`
- `boat-systems/drone_firmware/libraries/XBee/XBee.cpp`

8.26 nameValueObj::NameValueObj Class Reference

Public Member Functions

- `def __init__`
- `def returnName`
- `def returnValue`

Public Attributes

- `name`
- `value`

The documentation for this class was generated from the following file:

- `shore-systems/lpard/gui/nameValueObj.py`
#include `<XBee.h>

Inheritance diagram for PayloadRequest:

![Inheritance Diagram]

### Public Member Functions

- `PayloadRequest(uint8_t apiId, uint8_t frameId, uint8_t *payload, uint8_t payloadLength)`
- `uint8_t * getPayload()`
- `void setPayload(uint8_t *payloadPtr)`
- `uint8_t getPayloadLength()`
- `void setPayloadLength(uint8_t payloadLength)`

### 8.27.1 Detailed Description

All TX packets that support payloads extend this class

### 8.27.2 Member Function Documentation

#### 8.27.2.1 uint8_t PayloadRequest::getPayload()  
Returns the payload of the packet, if not null

#### 8.27.2.2 uint8_t PayloadRequest::getPayloadLength()  
Returns the length of the payload array, as specified by the user.

#### 8.27.2.3 void PayloadRequest::setPayload(uint8_t *payloadPtr)  
Sets the payload array
8.27.2.4 void PayloadRequest::setPayloadLength ( uint8_t payloadLength )

Sets the length of the payload to include in the request. For example if the payload array is 50 bytes and you only want the first 10 to be included in the packet, set the length to 10. Length must be <= to the array length.

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.28 RemoteAtCommandRequest Class Reference

#include <XBee.h>

Inheritance diagram for RemoteAtCommandRequest:

```
<table>
<thead>
<tr>
<th>XBeeRequest</th>
</tr>
</thead>
<tbody>
<tr>
<td>AtCommandRequest</td>
</tr>
<tr>
<td>RemoteAtCommandRequest</td>
</tr>
</tbody>
</table>
```

Public Member Functions

- RemoteAtCommandRequest (uint16_t remoteAddress16, uint8_t *command, uint8_t *commandValue, uint8_t commandValueLength)
- RemoteAtCommandRequest (uint16_t remoteAddress16, uint8_t *command)
- RemoteAtCommandRequest (XBeeAddress64 &remoteAddress64, uint8_t *command, uint8_t *commandValue, uint8_t commandValueLength)
- RemoteAtCommandRequest (XBeeAddress64 &remoteAddress64, uint8_t *command)
- uint16_t getRemoteAddress16 ()
- void setRemoteAddress16 (uint16_t remoteAddress16)
- XBeeAddress64 & getRemoteAddress64 ()
- void setRemoteAddress64 (XBeeAddress64 &remoteAddress64)
- bool getApplyChanges ()
- void setApplyChanges (bool applyChanges)
- uint8_t getFrameData (uint8_t pos)
- uint8_t getFrameDataLength ()

Static Public Attributes

- static XBeeAddress64 broadcastAddress64 = XBeeAddress64(0x0, BROADCAST_ADDRESS)
8.28 RemoteAtCommandRequest Class Reference

8.28.1 Detailed Description

Represents an Remote AT Command TX packet. The command is used to configure a remote XBee radio.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 RemoteAtCommandRequest::RemoteAtCommandRequest ( uint16_t remoteAddress16, uint8_t* command, uint8_t* commandValue, uint8_t commandValueLength )

Creates a RemoteAtCommandRequest with 16-bit address to set a command. 64-bit address defaults to broadcast and applyChanges is true.

8.28.2.2 RemoteAtCommandRequest::RemoteAtCommandRequest ( uint16_t remoteAddress16, uint8_t* command )

Creates a RemoteAtCommandRequest with 16-bit address to query a command. 64-bit address defaults to broadcast and applyChanges is true.

8.28.2.3 RemoteAtCommandRequest::RemoteAtCommandRequest ( XBeeAddress64 & remoteAddress64, uint8_t* command, uint8_t* commandValue, uint8_t commandValueLength )

Creates a RemoteAtCommandRequest with 64-bit address to set a command. 16-bit address defaults to broadcast and applyChanges is true.

8.28.2.4 RemoteAtCommandRequest::RemoteAtCommandRequest ( XBeeAddress64 & remoteAddress64, uint8_t* command )

Creates a RemoteAtCommandRequest with 16-bit address to query a command. 16-bit address defaults to broadcast and applyChanges is true.

8.28.3 Member Function Documentation

8.28.3.1 uint8_t RemoteAtCommandRequest::getFrameData ( uint8_t pos ) [virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum. Note: Unlike Digi’s definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Reimplemented from AtCommandRequest.
8.28.3.2 constexpr uint8_t RemoteAtCommandRequest::getFrameDataLength () [virtual]

Returns the size of the API frame (not including frame id or api id or checksum).
Reimplemented from AtCommandRequest.

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.29 RemoteAtCommandResponse Class Reference

#include <XBee.h>

Inheritance diagram for RemoteAtCommandResponse:

```
  XBeeResponse
   |
   FrameIdResponse
      |
      AtCommandResponse
         |
         RemoteAtCommandResponse
```

Public Member Functions

- constexpr uint8_t *getCommand ()
- constexpr uint8_t getStatus ()
- constexpr uint8_t *getValue ()
- constexpr uint8_t getValueLength ()
- constexpr uint16_t getRemoteAddress16 ()
-XBeeAddress64 & getRemoteAddress64 ()
- constexpr bool isOk ()

8.29.1 Detailed Description

Represents a Remote AT Command RX packet

8.29.2 Member Function Documentation
RemoteAtCommandResponse Class Reference

8.29.2.1 uint8_t RemoteAtCommandResponse::getCommand()

Returns an array containing the two character command
Reimplemented from AtCommandResponse.

8.29.2.2 uint16_t RemoteAtCommandResponse::getRemoteAddress16()

Returns the 16-bit address of the remote radio

8.29.2.3 XBeeAddress64 & RemoteAtCommandResponse::getRemoteAddress64()

Returns the 64-bit address of the remote radio

8.29.2.4 uint8_t RemoteAtCommandResponse::getStatus()

Returns the command status code. Zero represents a successful command
Reimplemented from AtCommandResponse.

8.29.2.5 uint8_t RemoteAtCommandResponse::getValue()

Returns an array containing the command value. This is only applicable to query commands.
Reimplemented from AtCommandResponse.

8.29.2.6 uint8_t RemoteAtCommandResponse::getValueLength()

Returns the length of the command value array.
Reimplemented from AtCommandResponse.

8.29.2.7 bool RemoteAtCommandResponse::isOk()

Returns true if command was successful
Reimplemented from AtCommandResponse.

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp
8.30 Rx16IoSampleResponse Class Reference

Inheritance diagram for Rx16IoSampleResponse:

- XBeeResponse
- RxDataResponse
- RxResponse
- RxIoSampleBaseResponse
- Rx16IoSampleResponse

Public Member Functions

- uint16_t getRemoteAddress16()
- uint8_t getRssiOffset()

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.31 Rx16Response Class Reference

#include <XBee.h>

Inheritance diagram for Rx16Response:

- XBeeResponse
- RxDataResponse
- RxResponse
- Rx16Response
Public Member Functions

- `uint8_t getRssiOffset ()`
- `uint16_t getRemoteAddress16 ()`

Protected Attributes

- `uint16_t _remoteAddress`

8.31.1 Detailed Description

Represents a Series 1 16-bit address RX packet

The documentation for this class was generated from the following files:

- `boat-systems/drone_firmware/libraries/XBee/XBee.h`
- `boat-systems/drone_firmware/libraries/XBee/XBee.cpp`

8.32 Rx64IoSampleResponse Class Reference

Inheritance diagram for Rx64IoSampleResponse:

```
  XBeeResponse
  |
  RxDataResponse
  |
  RxResponse
  |
 RxIoSampleBaseResponse
  |
 Rx64IoSampleResponse
```

Public Member Functions

- `XBeeAddress64 & getRemoteAddress64 ()`
- `uint8_t getRssiOffset ()`

The documentation for this class was generated from the following files:

- `boat-systems/drone_firmware/libraries/XBee/XBee.h`
- `boat-systems/drone_firmware/libraries/XBee/XBee.cpp`
8.33 Rx64Response Class Reference

#include <XBee.h>

Inheritance diagram for Rx64Response:

```
    XBeeResponse
     |
     V
    RxDataResponse
     |
     V
    RxResponse
     |
     V
    Rx64Response
```

Public Member Functions

- `uint8_t getRssiOffset ()`
- `XBeeAddress64 & getRemoteAddress64 ()`

8.33.1 Detailed Description

Represents a Series 1 64-bit address RX packet

The documentation for this class was generated from the following files:

- `boat-systems/drone_firmware/libraries/XBee/XBee.h`
- `boat-systems/drone_firmware/libraries/XBee/XBee.cpp`

8.34 RxDataResponse Class Reference

#include <XBee.h>

Inheritance diagram for RxDataResponse:
8.34 RxDataResponse Class Reference

Public Member Functions

- `uint8_t getData (int index)`
- `uint8_t * getData ()`
- `virtual uint8_t getDataLength ()=0`
- `virtual uint8_t getDataOffset ()=0`

8.34.1 Detailed Description

Common functionality for both Series 1 and 2 data RX data packets

8.34.2 Member Function Documentation

8.34.2.1 `uint8_t RxDataResponse::getData ( int index )`

Returns the specified index of the payload. The index may be 0 to `getDataLength() - 1`
This method is deprecated; use `uint8_t * getData()`

8.34.2.2 `uint8_t * RxDataResponse::getData ()`

Returns the payload array. This may be accessed from index 0 to `getDataLength() - 1`

8.34.2.3 `virtual uint8_t RxDataResponse::getDataLength ( ) [pure virtual]`

Returns the length of the payload
Implemented in `ZBRxResponse`, and `RxResponse`.

8.34.2.4 `virtual uint8_t RxDataResponse::getDataOffset ( ) [pure virtual]`

Returns the position in the frame data where the data begins
Implemented in `ZBRxResponse`, and `RxResponse`.

Generated on Tue May 8 2012 13:27:22 for LPARD-TDF-2012 by Doxygen
The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

### 8.35 RxIoSampleBaseResponse Class Reference

`#include <XBee.h>`

Inheritance diagram for RxIoSampleBaseResponse:

```
XBeResponse
  RxDataResponse
    RxResponse
      RxIoSampleBaseResponse
      Rx16IoSampleResponse
      Rx64IoSampleResponse
```

#### Public Member Functions

- `uint8_t getSampleSize ()`
- `bool containsAnalog ()`
- `bool containsDigital ()`
- `bool isAnalogEnabled (uint8_t pin)`
- `bool isDigitalEnabled (uint8_t pin)`
- `uint16_t getAnalog (uint8_t pin, uint8_t sample)`
- `bool isDigitalOn (uint8_t pin, uint8_t sample)`
- `uint8_t getSampleOffset ()`

### 8.35.1 Detailed Description

Represents a Series 1 RX I/O Sample packet

### 8.35.2 Member Function Documentation

#### 8.35.2.1 `uint16_t RxIoSampleBaseResponse::getAnalog ( uint8_t pin, uint8_t sample )`

Returns the 10-bit analog reading of the specified pin. Valid pins include ADC:0-5. Sample index starts at 0.
8.35.2.2  

`uint8_t RxIoSampleBaseResponse::getSampleSize()`  

Returns the number of samples in this packet.

8.35.2.3  

`bool RxIoSampleBaseResponse::isAnalogEnabled(uint8_t pin)`  

Returns true if the specified analog pin is enabled.

8.35.2.4  

`bool RxIoSampleBaseResponse::isDigitalEnabled(uint8_t pin)`  

Returns true if the specified digital pin is enabled.

8.35.2.5  

`bool RxIoSampleBaseResponse::isDigitalOn(uint8_t pin, uint8_t sample)`  

Returns true if the specified pin is high/on. Valid pins include DIO:0-8. Sample index starts at 0.

The documentation for this class was generated from the following files:

- `boat-systems/drone_firmware/libraries/XBee/XBee.h`
- `boat-systems/drone_firmware/libraries/XBee/XBee.cpp`

---

8.36  RxResponse Class Reference

#include <XBee.h>

Inheritance diagram for RxResponse:

```
XBeResponse

RxDataResponse

RxResponse

Rx16Response  Rx64Response  RxIoSampleBaseResponse

Rx16IoSampleResponse  Rx64IoSampleResponse
```

Public Member Functions

- `uint8_t getRssi()`  
- `uint8_t getOption()`  
- `bool isAddressBroadcast()`
8.36.1 Detailed Description

Represents a Series 1 RX packet

8.36.2 Member Function Documentation

8.36.2.1 uint8_t RxResponse::getDataLength ( ) [virtual]

Returns the length of the payload
Implements RxDataResponse.

8.36.2.2 uint8_t RxResponse::getDataOffset ( ) [virtual]

Returns the position in the frame data where the data begins
Implements RxDataResponse.

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.37 Tx16Request Class Reference

#include <XBee.h>

Inheritance diagram for Tx16Request:

```
XBeeRequest

PayloadRequest

Tx16Request
```

Public Member Functions

- **Tx16Request** (uint16_t addr16, uint8_t option, uint8_t *payload, uint8_t payloadLength, uint8_t frameId)
8.37 Tx16Request Class Reference

- Tx16Request (uint16_t addr16, uint8_t *payload, uint8_t payloadLength)
- Tx16Request()
- uint16_t getAddress16()
- void setAddress16(uint16_t addr16)
- uint8_t getOption()
- void setOption(uint8_t option)
- uint8_t getFrameData(uint8_t pos)
- uint8_t getFrameDataLength()

8.37.1 Detailed Description

Represents a Series 1 TX packet that corresponds to Api Id: TX_16_REQUEST

Be careful not to send a data array larger than the max packet size of your radio. This class does not perform any validation of packet size and there will be no indication if the packet is too large, other than you will not get a TX Status response. The datasheet says 100 bytes is the maximum, although that could change in future firmware.

8.37.2 Constructor & Destructor Documentation

8.37.2.1 Tx16Request::Tx16Request (uint16_t addr16, uint8_t *payload, uint8_t payloadLength) [virtual]

Creates a Unicast Tx16Request with the ACK option and DEFAULT_FRAME_ID

8.37.2.2 Tx16Request::Tx16Request() [virtual]

Creates a default instance of this class. At a minimum you must specify a payload, payload length and a destination address before sending this request.

8.37.3 Member Function Documentation

8.37.3.1 uint8_t Tx16Request::getFrameData(uint8_t pos) [virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi’s definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implements XBeeRequest.

8.37.3.2 uint8_t Tx16Request::getFrameDataLength() [virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Implements XBeeRequest.
The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

### 8.38 Tx64Request Class Reference

```cpp
#include <XBee.h>
```

Inheritance diagram for Tx64Request:

```
 XBeeRequest
  |
  v
PayloadRequest
  |
  v
Tx64Request
```

#### Public Member Functions

- `Tx64Request(XBeeAddress64 &addr64, uint8_t option, uint8_t *payload, uint8_t payloadLength, uint8_t frameId)`
- `Tx64Request(XBeeAddress64 &addr64, uint8_t *payload, uint8_t payloadLength)`
- `Tx64Request()`
- `XBeeAddress64 & getAddress64()`
- `void setAddress64(XBeeAddress64 &addr64)`
- `uint8_t getOption()`
- `void setOption(uint8_t option)`
- `uint8_t getFrameData(uint8_t pos)`
- `uint8_t getFrameDataLength()`

#### 8.38.1 Detailed Description

Represents a Series 1 TX packet that corresponds to Api Id: TX_64_REQUEST

Be careful not to send a data array larger than the max packet size of your radio. This class does not perform any validation of packet size and there will be no indication if the packet is too large, other than you will not get a TX Status response. The datasheet says 100 bytes is the maximum, although that could change in future firmware.

#### 8.38.2 Constructor & Destructor Documentation
8.38.2.1 \texttt{Tx64Request::Tx64Request ( XBeeAddress64 \& addr64, uint8 \_t \_payload, uint8 \_t payloadLength )}

Creates a unicast \texttt{Tx64Request} with the ACK option and DEFAULT_FRAME_ID.

8.38.2.2 \texttt{Tx64Request::Tx64Request ( )}

Creates a default instance of this class. At a minimum you must specify a payload, payload length and a destination address before sending this request.

8.38.3 Member Function Documentation

8.38.3.1 \texttt{uint8 \_t Tx64Request::getFrameData ( uint8 \_t pos ) [virtual]}

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implements \texttt{XBeeRequest}.

8.38.3.2 \texttt{uint8 \_t Tx64Request::getFrameDataLength ( ) [virtual]}

Returns the size of the api frame (not including frame id or api id or checksum).

Implements \texttt{XBeeRequest}.

The documentation for this class was generated from the following files:

- \texttt{boat-systems/drone_firmware/libraries/XBee/XBee.h}
- \texttt{boat-systems/drone_firmware/libraries/XBee/XBee.cpp}

8.39 \texttt{TxStatusResponse} Class Reference

\texttt{#include <XBee.h>}

Inheritance diagram for \texttt{TxStatusResponse}:

\begin{center}
\begin{tikzcd}
XBeeResponse \\
FrameIdResponse \\
TxStatusResponse
\end{tikzcd}
\end{center}

Generated on Tue May 8 2012 13:27:22 for LPARD-TDF-2012 by Doxygen
Public Member Functions

- `uint8_t getStatus ()`
- `bool isSuccess ()`

8.39.1 Detailed Description

Represents a Series 1 TX Status packet

The documentation for this class was generated from the following files:

- `boat-systems/drone_firmware/libraries/XBee/XBee.h`
- `boat-systems/drone_firmware/libraries/XBee/XBee.cpp`

8.40 GUI::Ui_about Class Reference

Public Member Functions

- `def setupUi`
- `def retranslateUi`

Public Attributes

- `frame_6`
- `line_4`
- `discardbutton`
- `notes`
- `frame_4`
- `lafayette`
- `created`
- `engineers`
- `software`
- `team`
- `recordUserName`
- `class_name`
- `members`
- `recordUserName_4`
- `date`
- `frame_5`
- `line_3`
- `about_lpard`
- `graphicsView`

The documentation for this class was generated from the following file:

- `shore-systems/lpard/gui/GUI_about_ui.py`
8.41 GUI\_mainPageReturn\_ui::Ui\_guiMain Class Reference

Public Member Functions

- def setupUi
- def retranslateUi

Public Attributes

- centralwidget
- frame
- groupBox
- scriptButton
- recordButton
- label_4
- yUpdate
- label_7
- stopButton
- label_6
- line_2
- returnToOrigin
- line_3
- doubleSpinBox\_x
- doubleSpinBox\_y
- scriptButton\_2
- groupBox\_4
- PayloadSensorTable
- groupBox\_5
- boatStatTable
- label\_5
- boatStatusLabel
- frame\_2
- groupBox\_3
- systemLog
- graph
- zoom
- label\_8
- line
- menubar
- menuFile
- menuSettings
- menuGraph\_Settings
- menuHelp
- menuCalibrate
- actionBaseline
- actionNew\_Session
• actionNew_Origin
• actionAbout
• actionUser_Manual
• actionClear_Graph

The documentation for this class was generated from the following file:

• shore-systems/lpard/gui/GUI_mainPageReturn_ui.py

8.42 GUI_manualPositioning_ui::Ui_manualPositioning Class Reference

Public Member Functions

• def setupUi
• def retranslateUi

Public Attributes

• frame_5
• doneButton
• cancelButton
• buttonUp
• buttonDown
• buttonRight
• buttonLeft
• frame
• line_3
• label_8

The documentation for this class was generated from the following file:

• shore-systems/lpard/gui/GUI_manualPositioning_ui.py

8.43 GUI_newSession_ui::Ui_newSession Class Reference

Public Member Functions

• def setupUi
• def retranslateUi
Public Attributes

- frame_6
- calibrateButton
- closeButton
- groupBox
- sosNew
- sosDefault
- sosPrevious
- distanceLabel
- baselineEdit
- groupBox_2
- label_3
- label_2
- userName
- location
- simulationModeBox
- label_4
- frame_5
- line_3
- label_7

The documentation for this class was generated from the following file:

- shore-systems/lpard/gui/GUI_newSession_ui.py

8.44 GUI_record_ui::Ui_recordData Class Reference

Public Member Functions

- def setupUi
- def retranslateUi

Public Attributes

- frame_6
- line_4
- discardbutton
- saveButton
- label_11
- observationNotes
- frame_4
- label_5
- label_6
- label_9
The documentation for this class was generated from the following file:

- shore-systems/lpard/gui/GUI_record_ui.py

8.45 GUI_script.ui::Ui_Script Class Reference

Public Member Functions

- `def setupUi`
- `def retranslateUi`

Public Attributes

- `frame_5`
- `line_3`
- `label_7`
- `frame_6`
- `verticalLayoutWidget`
- `verticalLayout`
- `horizontalLayout_2`
- `edit1_mode`
- `edit2_x`
- `edit3_y`
- `edit4`
- `edit5`
- `horizontalLayout`
- `comboBox_mode`
- `doubleSpinBox_x`
- `doubleSpinBox_y`
- `checkBox_record`
- `pushButton_insert`
- `listWidget`
8.46 XBee Class Reference

#include <XBee.h>

Public Member Functions

- void readPacket ()
- bool readPacket (int timeout)
- void readPacketUntilAvailable ()
- void begin (long baud)
- void getResponse (XBeeResponse &response)
- XBeeResponse & getResponse ()
- void send (XBeeRequest &request)
- uint8_t getNextFrameId ()
- void setSerial (HardwareSerial &serial)

8.46.1 Detailed Description

Primary interface for communicating with an XBee Radio. This class provides methods for sending and receiving packets with an XBee radio via the serial port. The XBee radio must be configured in API (packet) mode (AP=2) in order to use this software.

Since this code is designed to run on a microcontroller, with only one thread, you are responsible for reading the data off the serial buffer in a timely manner. This involves a call to a variant of readPacket(...). If your serial port is receiving data faster than you are reading, you can expect to lose packets. Arduino only has a 128 byte serial buffer so it can easily overflow if two or more packets arrive without a call to readPacket(...)

In order to conserve resources, this class only supports storing one response packet in memory at a time. This means that you must fully consume the packet prior to calling readPacket(...), because calling readPacket(...) overwrites the previous response.

This class creates an array of size MAX_FRAME_DATA_SIZE for storing the response packet. You may want to adjust this value to conserve memory.

Author

Andrew Rapp
8.46.2 Member Function Documentation

8.46.2.1 void XBee::begin (long baud)

Starts the serial connection at the supplied baud rate.

8.46.2.2 uint8_t XBee::getNextFrameId()

Returns a sequential frame id between 1 and 255.

8.46.2.3 XBeeResponse & XBee::getResponse()

Returns a reference to the current response. Note: once readPacket is called again this response will be overwritten!

8.46.2.4 bool XBee::readPacket (int timeout)

Waits a maximum of timeout milliseconds for a response packet before timing out; returns true if packet is read. Returns false if timeout or error occurs.

8.46.2.5 void XBee::readPacket()

Reads all available serial bytes until a packet is parsed, an error occurs, or the buffer is empty. You may call `xbee.getResponse().isAvailable()` after calling this method to determine if a packet is ready, or `xbee.getResponse().isError()` to determine if an error occurred.

This method should always return quickly since it does not wait for serial data to arrive. You will want to use this method if you are doing other timely stuff in your loop, where a delay would cause problems. NOTE: calling this method resets the current response, so make sure you first consume the current response.

8.46.2.6 void XBee::readPacketUntilAvailable()

Reads until a packet is received or an error occurs. Caution: use this carefully since if you don’t get a response, your Arduino code will hang on this call forever!! often it’s better to use a timeout: readPacket(int)

8.46.2.7 void XBee::send (XBeeRequest & request)

Sends a XBeeRequest (TX packet) out the serial port.
8.46.2.8 void XBee::setSerial ( HardwareSerial & serial )

Specify the serial port. Only relevant for Arduinos that support multiple serial ports (e.g. Mega)

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.47 XBeeAddress Class Reference

Inheritance diagram for XBeeAddress:

```
XBeeAddress

XBeeAddress64
```

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.48 XBeeAddress64 Class Reference

```c
#include <XBee.h>
```

Inheritance diagram for XBeeAddress64:

```
XBeeAddress

XBeeAddress64
```

Public Member Functions

- `XBeeAddress64 (uint32_t msb, uint32_t lsb)`
- `uint32_t getMsb ()`
- `uint32_t getLsb ()`
- `void setMsb (uint32_t msb)`
- `void setLsb (uint32_t lsb)`
8.48.1 Detailed Description

Represents a 64-bit XBe Address

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.49 Ipard::interface::IpardXBee::XbeeInterface Class Reference

Public Member Functions

- def __init__
  Constructor for XbeeInterface.
- def sendMessage
  Sends a packet with the specified message.
- def getMessage
  Gets the message from the next available packet.
- def closePort
  Closes the serial port to the XBee.
- def getPacket

Public Attributes

- config
- inQ
- outQ
- boat

8.49.1 Detailed Description

The XbeeInterface class is used by the LPARD system to connect to the boat system through the XBee RF transcievers.

8.49.2 Constructor & Destructor Documentation

8.49.2.1 def Ipard::interface::IpardXBee::XbeeInterface::__init__(self, zigbeePort, protocolConfig, simulate=False)

Constructor for XbeeInterface.

Parameters
### 8.49.3 Member Function Documentation

#### 8.49.3.1 `def lpard::interface::lpardXBee::XbeeInterface::closePort ( self )`

Closes the serial port to the XBee.

This must be called when the program is being shutdown.

**Parameters**

- `self` The pointer to the class instance.

#### 8.49.3.2 `def lpard::interface::lpardXBee::XbeeInterface::getMessage ( self )`

Gets the message from the next available packet.

**Parameters**

- `self` The pointer to the class instance.

#### 8.49.3.3 `def lpard::interface::lpardXBee::XbeeInterface::sendMessage ( self, message )`

Sends a packet with the specified message.

**Parameters**

- `self` The pointer to the class instance.
- `message` A string which contains the packet data to send.

The documentation for this class was generated from the following file:

- `shore-systems/lpard/interface/lpardXBee.py`

### 8.50 XBeeRequest Class Reference

```
#include <XBee.h>
```
Inheritance diagram for XBeeRequest:

```
XBeeRequest
   |___AtCommandRequest
   |___PayloadRequest
   |    |___RemoteAtCommandRequest
   |    |___Tx16Request
   |    |___Tx64Request
   |    |___ZBTxRequest
```

Public Member Functions

- `XBeeRequest (uint8_t apiId, uint8_t frameId)`
- `void setFrameId (uint8_t frameId)`
- `uint8_t getFrameId ()`
- `uint8_t getApiId ()`
- `virtual uint8_t getFrameData (uint8_t pos)=0`
- `virtual uint8_t getFrameDataLength ()=0`

Protected Member Functions

- `void setApiId (uint8_t apiId)`

### 8.50.1 Detailed Description

Super class of all XBee requests (TX packets) Users should never create an instance of this class; instead use a subclass of this class It is recommended to reuse Subclasses of the class to conserve memory

This class allocates a buffer to

### 8.50.2 Constructor & Destructor Documentation

#### 8.50.2.1 XBeeRequest::XBeeRequest ( uint8_t apiId, uint8_t frameId )

Constructor TODO make protected

### 8.50.3 Member Function Documentation

#### 8.50.3.1 uint8_t XBeeRequest::getApiId ( )

Returns the API id
8.50.3.2 virtual uint8_t XBeeRequest::getFrameData ( uint8_t pos ) [pure virtual]

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The reason for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.

Implemented in Tx16Request, Tx64Request, ZBTxRequest, AtCommandRequest, and RemoteAtCommandRequest.

8.50.3.3 virtual uint8_t XBeeRequest::getFrameDataLength ( ) [pure virtual]

Returns the size of the api frame (not including frame id or api id or checksum).

Implemented in Tx16Request, Tx64Request, ZBTxRequest, AtCommandRequest, and RemoteAtCommandRequest.

8.50.3.4 uint8_t XBeeRequest::getFrameId ( )

Returns the frame id

8.50.3.5 void XBeeRequest::setFrameId ( uint8_t frameId )

Sets the frame id. Must be between 1 and 255 inclusive to get a TX status response.

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.51 XBeeResponse Class Reference

#include <XBee.h>

Inheritance diagram for XBeeResponse:

![Inheritance Diagram]

Public Member Functions

- XBeeResponse ()
• uint8_t getApид ()
  • void setApид (uint8_t apid)
• uint8_t getMsbLength ()
  • void setMsbLength (uint8_t msbLength)
• uint8_t getLsbLength ()
  • void setLsbLength (uint8_t lsbLength)
• uint8_t getChecksum ()
  • void setChecksum (uint8_t checksum)
• uint8_t getFrameDataLength ()
  • void setFrameData (uint8_t *frameDataPtr)
• uint8_t *getFrameData ()
  • void setFrameLength (uint8_t frameLength)
• uint16_t getPacketLength ()
  • void reset ()
• void init ()
  • void getZBTxStatusResponse (XBeeResponse &response)
  • void getZBRxResponse (XBeeResponse &response)
  • void getZBRxIoSampleResponse (XBeeResponse &response)
• void getTxStatusResponse (XBeeResponse &response)
• void getRx16Response (XBeeResponse &response)
• void getRx64Response (XBeeResponse &response)
• void getRx16IoSampleResponse (XBeeResponse &response)
• void getRx64IoSampleResponse (XBeeResponse &response)
• void getAtCommandResponse (XBeeResponse &responses)
• void getRemoteAtCommandResponse (XBeeResponse &response)
• void getModemStatusResponse (XBeeResponse &response)
• bool isAvailable ()
  • void setAvailable (bool complete)
• bool isError ()
  • uint8_t getErrorCode ()
  • void setErrorCode (uint8_t errorCode)

Protected Attributes

• uint8_t * _frameDataPtr

8.51.1 Detailed Description

The super class of all XBee responses (RX packets) Users should never attempt to create an instance of this class; instead create an instance of a subclass It is recommend to reuse subclasses to conserve memory
8.51 XBeeResponse Class Reference

8.51.2 Constructor & Destructor Documentation

8.51.2.1 XBeeResponse::XBeeResponse ( )

Default constructor

Copyright (c) 2009 Andrew Rapp. All rights reserved.

This file is part of XBee-Arduino.

XBee-Arduino is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

XBee-Arduino is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with XBee-Arduino. If not, see <http://www.gnu.org/licenses/>.

8.51.3 Member Function Documentation

8.51.3.1 uint8_t XBeeResponse::getApiId ( )

Returns Api Id of the response

8.51.3.2 void XBeeResponse::getAtCommandResponse ( XBeeResponse & responses )

Call with instance of AtCommandResponse only if getApiId() == AT_COMMAND_RESPONSE

8.51.3.3 uint8_t XBeeResponse::getChecksum ( )

Returns the packet checksum

8.51.3.4 uint8_t XBeeResponse::getErrorCode ( )

Returns an error code, or zero, if successful. Error codes include: CHECKSUM_FAILURE, PACKET_EXCEEDS_BYTE_ARRAY_LENGTH, UNEXPECTED_START_BYTE

8.51.3.5 uint8_t * XBeeResponse::getFrameData ( )

Returns the buffer that contains the response. Starts with byte that follows API ID and includes all bytes prior to the checksum Length is specified by getFrameDataLength() Note: Unlike Digi's definition of the frame data, this does not start with the API ID. The
reason for this is all responses include an API ID, whereas my frame data includes only the API specific data.

8.51.3.6 uint8_t XBeeResponse::getFrameDataLength ( )

Returns the length of the frame data: all bytes after the api id, and prior to the checksum. Note up to release 0.1.2, this was incorrectly including the checksum in the length.

8.51.3.7 uint8_t XBeeResponse::getLsbLength ( )

Returns the LSB length of the packet.

8.51.3.8 void XBeeResponse::getModemStatusResponse ( XBeeResponse & response )

Call with instance of ModemStatusResponse only if getApiId() == MODEM_STATUS_RESPONSE.

8.51.3.9 uint8_t XBeeResponse::getMsbLength ( )

Returns the MSB length of the packet.

8.51.3.10 uint16_t XBeeResponse::getPacketLength ( )

Returns the length of the packet.

8.51.3.11 void XBeeResponse::getRemoteAtCommandResponse ( XBeeResponse & response )

Call with instance of RemoteAtCommandResponse only if getApiId() == REMOTE_AT_COMMAND_RESPONSE.

8.51.3.12 void XBeeResponse::getRx16IoSampleResponse ( XBeeResponse & response )

Call with instance of Rx16IoSampleResponse only if getApiId() == RX_16_IO_RESPONSE.

8.51.3.13 void XBeeResponse::getRx16Response ( XBeeResponse & response )

Call with instance of Rx16Response only if getApiId() == RX_16_RESPONSE.
8.51.3.14 void XBeeResponse::getRx64IoSampleResponse (XBeeResponse & response)
Call with instance of Rx64IoSampleResponse only if getApilid() == RX_64_IO_RESPONSE

8.51.3.15 void XBeeResponse::getRx64Response (XBeeResponse & response)
Call with instance of Rx64Response only if getApilid() == RX_64_RESPONSE

8.51.3.16 void XBeeResponse::getTxStatusResponse (XBeeResponse & response)
Call with instance of TxStatusResponse only if getApilid() == TX_STATUS_RESPONSE

8.51.3.17 void XBeeResponse::getZBRxIoSampleResponse (XBeeResponse & response)
Call with instance of ZBRxIoSampleResponse class only if getApilid() == ZB_IO_SAMPLE_RESPONSE to populate response

8.51.3.18 void XBeeResponse::getZBRxResponse (XBeeResponse & response)
Call with instance of ZBRxResponse class only if getApilid() == ZB_RX_RESPONSE to populate response

8.51.3.19 void XBeeResponse::getZBTxStatusResponse (XBeeResponse & response)
Call with instance of ZBTxStatusResponse class only if getApilid() == ZB_TX_STATUS_RESPONSE to populate response

8.51.3.20 void XBeeResponse::init()
Initializes the response

8.51.3.21 bool XBeeResponse::isAvailable()
Returns true if the response has been successfully parsed and is complete and ready for use

8.51.3.22 bool XBeeResponse::isError()
Returns true if the response contains errors

Generated on Tue May 8 2012 13:27:22 for LPARD-TDF-2012 by Doxygen
8.51.3.23 void XBeeResponse::reset()

Resets the response to default values

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.52 ZBRxIoSampleResponse Class Reference

```cpp
#include <XBee.h>
```

Inheritance diagram for ZBRxIoSampleResponse:

```
XBeeResponse
   |
   V
RxDataResponse
   |
   V
ZBRxResponse
   |
   V
ZBRxIoSampleResponse
```

Public Member Functions

- bool `containsAnalog` ()
- bool `containsDigital` ()
- bool `isAnalogEnabled` (uint8_t pin)
- bool `isDigitalEnabled` (uint8_t pin)
- uint16_t `getAnalog` (uint8_t pin)
- bool `isDigitalOn` (uint8_t pin)
- uint8_t `getDigitalMaskMsb` ()
- uint8_t `getDigitalMaskLsb` ()
- uint8_t `getAnalogMask` ()

8.52.1 Detailed Description

Represents a Series 2 RX I/O Sample packet
8.52.2 Member Function Documentation

8.52.2.1 uint16_t ZBRxIoSampleResponse::getAnalog ( uint8_t pin )

Returns the 10-bit analog reading of the specified pin. Valid pins include ADC:xxx.

8.52.2.2 bool ZBRxIoSampleResponse::isAnalogEnabled ( uint8_t pin )

Returns true if the pin is enabled.

8.52.2.3 bool ZBRxIoSampleResponse::isDigitalEnabled ( uint8_t pin )

Returns true if the pin is enabled.

8.52.2.4 bool ZBRxIoSampleResponse::isDigitalOn ( uint8_t pin )

Returns true if the specified pin is high/on. Valid pins include DIO:xxx.

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.53 ZBRxResponse Class Reference

#include <XBee.h>

Inheritance diagram for ZBRxResponse:

```
+-------------------------+
| XBeeResponse            |
|                         |
| RxDataResponse          |
|                         |
| ZBRxResponse            |
|                         |
| ZBRxIoSampleResponse    |
```

Public Member Functions

- XBeeAddress64 & getRemoteAddress64 ()
- uint16_t getRemoteAddress16 ()
- uint8_tgetOption ()
8.53.1 Detailed Description

Represents a Series 2 RX packet

8.53.2 Member Function Documentation

8.53.2.1 `uint8_t ZBRxResponse::getDataLength () [virtual]`

Returns the length of the payload

Implements RxDataResponse.

8.53.2.2 `uint8_t ZBRxResponse::getDataOffset () [virtual]`

Returns the position in the frame data where the data begins

Implements RxDataResponse.

The documentation for this class was generated from the following files:

- `boat-systems/drone_firmware/libraries/XBee/XBee.h`
- `boat-systems/drone_firmware/libraries/XBee/XBee.cpp`

8.54 ZBTxRequest Class Reference

#include `<XBee.h>`

Inheritance diagram for ZBTxRequest:

```
   XBeeRequest
      |
      v
PayloadRequest
      |
      v
ZBTxRequest
```

Public Member Functions

- `ZBTxRequest (XBeeAddress64 &addr64, uint8_t *payload, uint8_t payloadLength)`
- `ZBTxRequest (XBeeAddress64 &addr64, uint16_t addr16, uint8_t broadcastRadius, uint8_t option, uint8_t *payload, uint8_t payloadLength, uint8_t frameld)`
8.54 ZBTxRequest Class Reference

- ZBTxRequest()
- XBeeAddress64 & getAddress64()
- uint16_t getAddress16()
- uint8_t getAddressRadius()
- uint8_t getOption()
- void setAddress64(XBeeAddress64 &addr64)
- void setAddress16(uint16_t addr16)
- void setBroadcastRadius(uint8_t broadcastRadius)
- void setOption(uint8_t option)

Protected Member Functions

- uint8_t setFrameData(uint8_t pos)
- uint8_t setFrameDataLength()

8.54.1 Detailed Description

Represents a Series 2 TX packet that corresponds to Api Id: ZB_TX_REQUEST

Be careful not to send a data array larger than the max packet size of your radio. This class does not perform any validation of packet size and there will be no indication if the packet is too large, other than you will not get a TX Status response. The datasheet says 72 bytes is the maximum for ZNet firmware and ZB Pro firmware provides the ATNP command to get the max supported payload size. This command is useful since the maximum payload size varies according to certain settings, such as encryption. ZB Pro firmware provides a PAYLOAD_TOO_LARGE that is returned if payload size exceeds the maximum.

8.54.2 Constructor & Destructor Documentation

8.54.2.1 ZBTxRequest::ZBTxRequest(XBeeAddress64 &addr64, uint8_t *payload, uint8_t payloadLength)

Creates a unicast ZBTxRequest with the ACK option and DEFAULT_FRAME_ID

8.54.2.2 ZBTxRequest::ZBTxRequest()

Creates a default instance of this class. At a minimum you must specify a payload, payload length and a destination address before sending this request.

8.54.3 Member Function Documentation

8.54.3.1 uint8_t ZBTxRequest::getFrameData(uint8_t pos)

Starting after the frame id (pos = 0) and up to but not including the checksum Note: Unlike Digi’s definition of the frame data, this does not start with the API ID. The reason
for this is the API ID and Frame ID are common to all requests, whereas my definition of frame data is only the API specific data.
Implements XBeeRequest.

8.54.3.2 uint8_t ZBTxRequest::getFrameDataLength ( ) [protected, virtual]
Returns the size of the api frame (not including frame id or api id or checksum).
Implements XBeeRequest.
The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp

8.55 ZBTxStatusResponse Class Reference

#include <XBee.h>

Inheritance diagram for ZBTxStatusResponse:

```
XBeeResponse
 |
↓
FrameIdResponse
 |
↓
ZBTxStatusResponse
```

Public Member Functions

- uint16_t getRemoteAddress ( )
- uint8_t getTxRetryCount ( )
- uint8_t getDeliveryStatus ( )
- uint8_t getDiscoveryStatus ( )
- bool isSuccess ( )

8.55.1 Detailed Description

Represents a Series 2 TX status packet

The documentation for this class was generated from the following files:

- boat-systems/drone_firmware/libraries/XBee/XBee.h
- boat-systems/drone_firmware/libraries/XBee/XBee.cpp
Chapter 9

File Documentation

9.1 boat-systems/drone_firmware/drone_firmware.pde File Reference

Main Arduino code file for boat firmware.

```c
#include <Wire.h>
#include <XBee.h>
#include <GPS_NMEA.h>
#include <Comp6DOF_n0m1.h>
#include <HMC5883L.h>
```

Functions

- `void setup ()`
- `void loop ()`
- `void calibrate ()`
- `boolean init_compass ()`
- `void transmit (String str)`
- `void receivePacket (int timeout)`
- `void parseIncoming (uint8_t *data)`
- `void send_status ()`
- `void check_gps ()`
- `void check_compass ()`
- `float nonCompHeading (MagnetometerRaw raw)`

Variables

- `long timer`

  *Used to track the start time of the most recent loop.*
- `long diff`
Used to determine the time between loops and limit to 1 Hz.

- `const char * DELIM = " "`
  Packet delimiter.
- `XBee xbee = XBee()`
  Our XBee object.
- `ZBTxRequest zbTx`
  Our transmit request.
- `XBeeResponse response = XBeeResponse()`
  For incoming packets.
- `ZBRxResponse rx = ZBRxResponse()`
  For incoming ZB packets.
- `ModemStatusResponse msr = ModemStatusResponse()`
  For modem status response.
- `XBeeAddress64 addr64 = XBeeAddress64(0, 0)`
  Coordinator address.
- `Comp6DOF_n0m1 sixDOF`
  Object used for tilt compensation.
- `HMC5883L compass`
  The actual compass.
- `float declination`
  Declination angle for the operating area.
- `int pos [3]`
  Boat positions in the order X, Y, Z in cm.
- `int head`
  Heading in 100ths of degrees.
- `long comp_read`
  Most recent compass reading in 100ths of degrees.
- `int accel_read [3]`
  Most recent accelerometer readings going X, Y, Z.
- `int gyro_read [3]`
  Most recent gyroscope readings going X, Y, Z.
- `int volt_read`
  Subsystem voltage readings.
- `int amp_read`
  Subsystem current readings.
- `int power_read`
  Subsystem power readings.
- `int temp_read`
  Subsystem temperature readings.
- `char ctrl_mode = 'A'`
  Control state where A->auto and M->manual.
- `long gps_time`
  Time from epoch (from the GPS) in ms.
• long gps_lat
  
  Latitude multiplied by 10^7 to clear the decimal.
• long gps_long
  
  Longitude multiplied by 10^7 to clear the decimal.
• long gps_speed
  
  GPS speed over ground field. In m/s * 100?
• long gps_angle
  
  GPS track angle. Degrees * 100?

9.1.1 Detailed Description

Main Arduino code file for boat firmware. Contains both the setup and loop Arduino functions, as well as convenience methods for XBee communication and boat sensor controls. Note that this is the second code branch - the first has more sensors operating but does not have working XBee communication.

Author

Sam Courtney <courtnes@lafayette.edu>

Version

1.0

Date

2012

Bug

Compass not tilt compensated.
Compass iron offsets infinite loop, and have been disabled.
Doesn’t receive XBee packets properly. Could be a mode issue (needs API=2).
Cannot set declination angle externally.

9.1.2 Function Documentation

9.1.2.1 void calibrate ( )

Method handling calibration.

9.1.2.2 void check_compass ( )

Convenience method that updates compass data and the relevant boat state fields.

9.1.2.3 void check_gps ( )

Convenience method that checks for new GPS data and updates the relevant boat state fields.
9.1.2.4 boolean init_compass ( )

Method handling compass initialization. Note that this also handles the calculation of hard iron offsets.

**Returns**

true if no error was detected

9.1.2.5 void loop ( )

Main loop for the boat firmware. The ArduPilot automatically loops through this code.

9.1.2.6 float nonCompHeading (MagnetometerRaw raw)

Convenience method that updates calculates the non-compensated heading value using the HMC5883L compass. In other words, we're assuming the compass is level.

**Returns**

the heading value

9.1.2.7 void parseIncoming (uint8_t * data)

Convenience method that handles parsing of our protocol's packets.

**Parameters**

| data | Pointer to the frame data that is to be parsed. |

9.1.2.8 void receivePacket (int timeout)

Convenience method allowing reception of an XBee packet.

**Parameters**

| timeout | The time to wait for an incoming packet |

9.1.2.9 void send_status ( )

Convenience method that handles construction of a boat status packet. Also calls the send method.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Pointer to the frame data that is to be parsed.</td>
</tr>
</tbody>
</table>

9.1.2.10 void setup ( )

Setup loop for firmware. Automatically run any time the ArduPilot receives power.

9.1.2.11 void transmit ( String str )

Convenience method allowing transmission to the stored address.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>The character string to send</td>
</tr>
</tbody>
</table>

9.2 shore-systems/AZ/EL/Code/AZELCode.h File Reference

Firmware for the AZ/EL Positioner.

```c
#include <avr/io.h>
#include <avr-interrupt.h>
#include "pwm.h"
#include "i2c.h"
#include "ultrasound.h"
#include "tracking.h"
#include <util/delay.h>
```

Defines

- `#define F_CPU 8000000UL`

Enumerations

- `enum clock_div_t {
  clock_div_1 = 0, clock_div_2 = 1, clock_div_4 = 2, clock_div_8 = 3,
  clock_div_16 = 4, clock_div_32 = 5, clock_div_64 = 6, clock_div_128 = 7,
  clock_div_256 = 8, clock_div_1 = 0, clock_div_2 = 1, clock_div_4 = 2,
  clock_div_8 = 3, clock_div_16 = 4, clock_div_32 = 5, clock_div_64 = 6,
  clock_div_128 = 7, clock_div_256 = 8, clock_div_1 = 0, clock_div_2 = 1,
  clock_div_4 = 2, clock_div_8 = 3, clock_div_16 = 4, clock_div_32 = 5,
} `
clock\textsubscript{div\_64} = 6, clock\textsubscript{div\_128} = 7, clock\textsubscript{div\_256} = 8 \}

Various Clock Divider settings.

9.2.1 Detailed Description

Firmware for the AZ/EL Positioner. This program runs in the ATmega 328 of the AZ/EL Positioner to manage operations. Current features include

1. PWM Drive of the AZ and EL servos
2. Manual joystick mode
3. Automatic analog position input mode
4. Soft limits (Hard Coded)
5. I2C Interface (incomplete)

Author

Chris Nadovich <nadovicc@lafayette.edu>
David Salter <salterd@lafayette.edu>

Version

1.02

Date

31 March 2012

Bug

Main loop does not have fixed timing.
Hard coded limits.
Index

__init__
  guiAbout::GuiAbout, 32
guiScript::GuiScript, 36
guiScript::Mode, 43
lpard::controller::lpardController::Controller, 29
lpard::interface::BoatSim::BoatSim, 25
lpard::interface::lpardI2C::I2CInterface, 39
lpard::interface::lpardXBee::Boat, 23
lpard::interface::lpardXBee::XbeeInterface, 68
__repr__
guiScript::Mode, 43

ADXL335, 19
AtCommandRequest, 19
clearCommandValue, 20
getFrameData, 20
getFrameDataLength, 20
AtCommandResponse, 21
getCommand, 21
getStatus, 22
getValue, 22
getValueLength, 22
isOk, 22
AZEL_Mode
  lpard::interface::lpardI2C::I2CInterface, 39

begin
  XBee, 66
boat-systems/drone_firmware/drone_firmware.pde, 81
boatCommand
  lpard::interface::BoatSim::BoatSim, 25
boatCommandResponse
  lpard::interface::BoatSim::BoatSim, 25
BoatSim, 13
boatStatusMessage
  lpard::interface::BoatSim::BoatSim, 25

getAnalog

check_compass
  drone_firmware.pde, 83
clearCommandValue
  AtCommandRequest, 20
closePort
  lpard::interface::lpardXBee::XbeeInterface, 69
command
  lpard::interface::BoatSim::BoatSim, 25
Comp6DOF_n0m1, 27
clearCommandValue
controller
guiScript::GuiScript, 37
data
  lpard::interface::lpardI2C::I2CInterface, 40

check_gps
  drone_firmware.pde, 83
clearCommandValue
  AtCommandRequest, 20
closePort
  lpard::interface::lpardXBee::XbeeInterface, 69
command
  lpard::interface::BoatSim::BoatSim, 25
clearCommandValue
controller
guiScript::GuiScript, 37
data
  lpard::interface::lpardI2C::I2CInterface, 40

isOk

check_gps
  drone_firmware.pde, 83
clearCommandValue
  AtCommandRequest, 20
closePort
  lpard::interface::lpardXBee::XbeeInterface, 69
command
  lpard::interface::BoatSim::BoatSim, 25
clearCommandValue
controller
guiScript::GuiScript, 37
data
  lpard::interface::lpardI2C::I2CInterface, 40

isOk

check_compass
  drone_firmware.pde, 83
clearCommandValue
  AtCommandRequest, 20
closePort
  lpard::interface::lpardXBee::XbeeInterface, 69
command
  lpard::interface::BoatSim::BoatSim, 25
clearCommandValue
controller
guiScript::GuiScript, 37
data
  lpard::interface::lpardI2C::I2CInterface, 40

isOk

check_gps
  drone_firmware.pde, 83
init_compass, 83
loop, 84

RxIoSampleBaseResponse, 54
ZBRxIoSampleResponse, 77
getApiId
XBeeRequest, 70
XBeeResponse, 73
getAtCommandResponse
XBeeResponse, 73
getChecksum
XBeeResponse, 73
getCommand
AtCommandResponse, 21
RemoteAtCommandResponse, 48
data
RXDataResponse, 53
getDataLength
RXDataResponse, 53
RXResponse, 56
ZBRxResponse, 78
dataOffset
RXDataResponse, 53
RXResponse, 56
ZBRxResponse, 78
degrees
lpard::controller::lpardController::Controller, 29
degreesPerPwm
lpard::controller::lpardController::Controller, 30
dGetErrorData
XBeeResponse, 73
dataFrameCommand
AtCommandRequest, 20
RemoteAtCommandRequest, 47
Tx16Request, 57
Tx64Request, 59
XBeeRequest, 70
XBeeResponse, 73
ZBTxRequest, 79
getFrameDataLength
AtCommandRequest, 20
RemoteAtCommandRequest, 47
Tx16Request, 57
Tx64Request, 59
XBeeRequest, 71
XBeeResponse, 74
ZBTxRequest, 80
dataframeId
XBeeRequest, 71
getLsbLength
XBeeResponse, 74
getMessage
lpard::interface::lpardXBee::XbeeInterface, 69
getMessages
lpard::interface::BoatSim::BoatSim, 26
getModemStatusResponse
XBeeResponse, 74
getMsbLength
XBeeResponse, 74
getNextFrameId
XBeeResponse, 74
dataPacketLength
XBeeResponse, 74
dataPayload
PayloadRequest, 45
dataPayloadLength
PayloadRequest, 45
getRemoteAddress16
RemoteAtCommandResponse, 49
getRemoteAddress64
RemoteAtCommandResponse, 49
getRemoteAtCommandResponse
XBeeResponse, 74
dataResponse
XBee, 66
dataResponse
getRx16IoSampleResponse
XBeeResponse, 74
dataResponse
getRx16Response
XBee, 66
dataResponse
getRx16IoSampleResponse
XBeeResponse, 74
dataResponse
getRx64IoSampleResponse
XBeeResponse, 74
dataResponse
getRx64Response
XBeeResponse, 75
dataResponse
getSampleSize
RXIoSampleBaseResponse, 54
dataStatus
AtCommandResponse, 22
RemoteAtCommandResponse, 49
dataResponse
getTStatusResponse
XBeeResponse, 75
dataValue
AtCommandResponse, 22
RemoteAtCommandResponse, 49
dataValueLength
AtCommandResponse, 22
RemoteAtCommandResponse, 49
dataResponse
getZBRxIoSampleResponse
XBeeResponse, 75
dataResponse
getZBRxResponse
XBeeResponse, 75
### transimt
- drone_firmware.pde, 85
- Tx16Request, 56
  - getFrameData, 57
  - getFrameDataLength, 57
- Tx16Request, 57
- Tx64Request, 58
  - getFrameData, 59
  - getFrameDataLength, 59
- Tx64Request, 58, 59
- TxStatusResponse, 59

### ui
- guiAbout::GuiAbout, 33

### update
- lpard::interface::BoatSim::BoatSim, 26
- US_Mode
  - lpard::interface::lpardI2C::I2CInterface, 40
- utils, 17

### X
- guiScript::Mode, 43
- XBee, 65
  - begin, 66
  - getNextFrameld, 66
  - onResponse, 66
  - readPacket, 66
  - readPacketUntilAvailable, 66
  - send, 66
  - setSerial, 66
- XBeeAddress, 67
- XBeeAddress64, 67
- XBeeRequest, 69
  - getApid, 70
  - getFrameData, 70
  - getFrameDataLength, 71
  - getFrameId, 71
  - setFrameId, 71
- XBeeResponse, 70
- XBeeResponse, 71
  - getApid, 73
  - getAtCommandResponse, 73
  - getChecksum, 73
  - getColorCode, 73
  - getFrameData, 73
  - getFrameDataLength, 74
  - getLsbLength, 74
  - getModemStatusResponse, 74
  - getMsbLength, 74
  - getPacketLength, 74
  - getRemoteAtCommandResponse, 74
  - getRx16IoSampleResponse, 74
  - getRx16Response, 74
  - getRx64IoSampleResponse, 74
  - getRx64Response, 75
  - getTxStatusResponse, 75
  - getZBRxIoSampleResponse, 75
  - getZBRxResponse, 75
  - getZBTxStatusResponse, 75
  - init, 75
  - isAvailable, 75
  - isError, 75
  - reset, 75
- XBeeResponse, 73
- guiScript::Mode, 43
- ZBRxIoSampleResponse, 76
  - getAnalog, 77
  - isAnalogEnabled, 77
  - isDigitalEnabled, 77
  - isDigitalOn, 77
- ZBRxResponse, 77
  - getDataLength, 78
  - getDataOffset, 78
- ZBTxRequest, 78
  - getFrameData, 79
  - getFrameDataLength, 80
  - ZBTxRequest, 79
  - ZBTxStatusResponse, 80

---

Generated on Tue May 8 2012 13:27:22 for LPARD-TDF-2012 by Doxygen