

Vilas Kumar Chitrakaran

Mon Aug 28 15:21:10 2006

Contents

1	Logitech Wingman 3D Class Driver for QNX Hierarchical Index	1
	1.1 Logitech Wingman 3D Class Driver for QNX Class Hierarchy]
2	Logitech Wingman 3D Class Driver for QNX Class Index	3
	2.1 Logitech Wingman 3D Class Driver for QNX Class List	é
3	Logitech Wingman 3D Class Driver for QNX File Index	ţ
	3.1 Logitech Wingman 3D Class Driver for QNX File List	ļ
4	Logitech Wingman 3D Class Driver for QNX Class Documentation	7
	4.1 Wingman3D Class Reference	7
	4.2 wingman_data Struct Reference	12
	4.3 wingman_device Struct Reference	14
	4.4 wingman_report Struct Reference	15
5	Logitech Wingman 3D Class Driver for QNX File Documentation	17
	5.1 Wingman3D.hpp File Reference	17

Logitech Wingman 3D Class Driver for QNX Hierarchical Index

1.1 Logitech Wingman 3D Class Driver for QNX Class Hierarchy

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

Wingman3D	
wingman_data	15
wingman_device	1
wingman report	\dots 1

2	$\mathbf{Logitech}$	Wingman	3D Cla	ss Driver	for QN	X Hierarchica	l Index

Logitech Wingman 3D Class Driver for QNX Class Index

2.1 Logitech Wingman 3D Class Driver for QNX Class List

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

Wingman3D (The class Wingman3D provides an interface to the following Logitech (Ven-	
dor ID: 0x046D) USB HID compliant joysticks:)	1
wingman data (A structure for holding wingman joystick data)	2
wingman device (A structure for wingman devices (for driver internal use)) 1	4
wingman report (A structure for device reports (for driver internal use)) 1	į

4	Logitech	Wingman 3D	Class	Driver for	QNX	Class 1	Index

Logitech Wingman 3D Class Driver for QNX File Index

3.1	Logitech	Wingman	3D	Class	Driver	for	ONX	File	List
υ. Ι	Logiccii	vv iiigiiiaii	$\sigma \mathbf{D}$	Class	DIIVCI	101	$\omega_{\perp} \cdot \nabla_{\perp}$	1110	LISU

Here is a list of all file	s with brief descriptions:	
Wingman3D.hpp		1

6	Logitech	Wingman	3D	Class	Driver	for	QNX	File l	$\underline{\mathbf{ndex}}$

Logitech Wingman 3D Class Driver for QNX Class Documentation

4.1 Wingman3D Class Reference

The class Wingman3D provides an interface to the following Logitech (Vendor ID: 0x046D) USB HID compliant joysticks:.

#include <Wingman3D.hpp>

Public Member Functions

- Wingman3D (int devNum=0)
- ~Wingman3D ()
- int getX () const
- int getMaxX () const
- int getY () const
- int getMaxY () const
- int getTwist () const
- int getMaxTwist () const
- int getNumButtons () const
- bool isButtonPressed (int button) const
- int getSliderValue () const
- int getMaxSliderValue () const
- int getHatSwitchStatus () const
- bool isStatusOk () const
- char * getStatusMessage () const
- void printDeviceInfo (int verbosity=1) const

Protected Member Functions

- wingman data t getWingmanData () const
- wingman data t getWingmanMaxData () const

4.1.1 Detailed Description

The class Wingman3D provides an interface to the following Logitech (Vendor ID: 0x046D) USB HID compliant joysticks:.

- Wingman Extreme 3D Digital (Product ID: 0xC212)
- Force3D (Product ID: 0xC283) (no force feedback support)
- Extreme 3D Pro (Product ID: 0xc215)

To use the joystick please ensure that the USB manager (devu-uhci or devu-ohci) and manager for HID devices (io-hid) are running. Your application will also need to be linked with the HID library libhiddi.so and the library created by this class. You will need to be root to use this class. Here are the commands to start the device managers (as root):

```
/sbin/devu-uhci & (or /sbin/devu-ohci &)
/sbin/io-hid &
mount -Tio-hid devh-usb.so &
```

Example Program:

```
//-----
// Wingman3D.t.cpp : Example program for Logitech Wingman HID Joystick
//
// Author
                : Vilas Kumar Chitrakaran < cvilas@ces.clemson.edu>
// Date
                 : March 2, 2004
// Compiler
                : GNU GCC 2.95.3qnx-nto
// Operating System : QNX Momentics 6.2.1
#include "Wingman3D.hpp"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main()
Wingman3D joystick(0);
if( !joystick.isStatusOk() ) {
 fprintf(stderr, "%s\n", joystick.getStatusMessage());
 return EXIT_FAILURE;
sleep(1);
joystick.printDeviceInfo(3);
while(1)
 fprintf(stdout, "\nx: %03d y: %03d z: %03d slide: %03d hat: %03d buttons: ",
  joystick.getX(), joystick.getY(), joystick.getTwist(),
  joystick.getSliderValue(), joystick.getHatSwitchStatus());
 for(int i = 1; i < joystick.getNumButtons(); i++)</pre>
  if( joystick.isButtonPressed(i) )
   fprintf(stdout, "%01d ", i);
 if( !joystick.isStatusOk() )
  return -1;
return EXIT_SUCCESS;
}
```

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Wingman3D::Wingman3D (int devNum = 0)

The default constructor. Establishes connection with the HID driver.

Parameters:

devNum The address of the joystick device to connect to (default = 0).

4.1.2.2 Wingman3D::~Wingman3D ()

The default destructor disconnects from the HID driver and cleans up.

4.1.3 Member Function Documentation

4.1.3.1 int Wingman3D::getX () const

Returns:

The X position

4.1.3.2 int Wingman3D::getMaxX () const

Returns:

The maximum output from X axis.

4.1.3.3 int Wingman3D::getY () const

Returns:

The Y position

4.1.3.4 int Wingman3D::getMaxY () const

Returns:

The maximum output from Y axis.

4.1.3.5 int Wingman3D::getTwist () const

Returns:

The twist handle position.

4.1.3.6 int Wingman3D::getMaxTwist () const

Returns:

The maximum output from twist axis.

4.1.3.7 int Wingman3D::getNumButtons () const

Returns:

The number of programmable buttons available

4.1.3.8 bool Wingman3D::isButtonPressed (int button) const

Returns:

true if button button is pressed, else false

4.1.3.9 int Wingman3D::getSliderValue () const

Returns:

The slider/throttle position. Range [0 - 255].

4.1.3.10 int Wingman3D::getMaxSliderValue () const

Returns:

The maximum output from slider.

4.1.3.11 int Wingman3D::getHatSwitchStatus () const

Returns:

A value indicating position of the hat switch. Range [1 - 8].

4.1.3.12 bool Wingman3D::isStatusOk () const

Returns:

true if no error, else false.

4.1.3.13 char* Wingman3D::getStatusMessage () const

Returns:

A string carrying current status of the device. This string also carries error messages when isStatusOk() returns false.

4.1.3.14 void Wingman3D::printDeviceInfo (int verbosity = 1) const

Print information about the device to stdout. Higher values of *verbosity* mean more detailed information.

4.1.3.15 wingman data t Wingman3D::getWingmanData () const [protected]

Returns:

The whole data structure for device with current joystick data

Returns:

The whole data structure for device with max limits for joystick data

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

 $\bullet \ \ Wingman 3D.hpp$

4.2 wingman data Struct Reference

A structure for holding wingman joystick data.

#include <Wingman3D.hpp>

Public Attributes

• bool statusOk

'true' if device status is OK.

• char statusMessage [80]

Char buffer to hold device status message.

• int x

X axis data.

• int y

Y axis data.

• int twist

Z axis data.

• int buttons

Button status data.

 \bullet int slider

 $Slider\ axis\ data.$

• int hatSwitch

 $Hat\ switch\ position\ data.$

4.2.1 Detailed Description

A structure for holding wingman joystick data.

4.2.2 Member Data Documentation

4.2.2.1 bool wingman data::statusOk

'true' if device status is OK.

4.2.2.2 char wingman data::statusMessage[80]

Char buffer to hold device status message.

4.2.2.3 int wingman data::x

X axis data.

4.2.2.4 int wingman data::y

Y axis data.

4.2.2.5 int wingman data::twist

Z axis data.

4.2.2.6 int wingman data::buttons

Button status data.

4.2.2.7 int wingman data::slider

Slider axis data.

4.2.2.8 int wingman data::hatSwitch

Hat switch position data.

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

• Wingman3D.hpp

4.3 wingman_device Struct Reference

A structure for wingman devices (for driver internal use). #include <Wingman3D.hpp>

Public Attributes

- wingman_data_t data device current data.
- wingman_data_t max_data max limits on data fields.
- wingman_report_t * report
- $\bullet \ \, hidd_device_instance_t * device_instance$

4.3.1 Detailed Description

A structure for wingman devices (for driver internal use).

4.3.2 Member Data Documentation

4.3.2.1 wingman_data_t wingman_device::data

4.3.2.2 wingman data t wingman device::max data

max limits on data fields.

device current data.

- 4.3.2.3 wingman report t* wingman device::report
- 4.3.2.4 hidd device instance t* wingman device::device instance

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

• Wingman3D.hpp

4.4 wingman report Struct Reference

A structure for device reports (for driver internal use). #include <Wingman3D.hpp>

Public Attributes

- hidd report instance * creport instance
- hidd report * creport
- $_uint16 * cbtnbuf$
- uint32 dev no

4.4.1 Detailed Description

A structure for device reports (for driver internal use).

4.4.2 Member Data Documentation

- 4.4.2.1 struct hidd report instance* wingman report::creport instance
- 4.4.2.2 struct hidd report* wingman report::creport
- 4.4.2.3 uint16* wingman report::cbtnbuf
- 4.4.2.4 _uint32 wingman_report::dev_no

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

• Wingman3D.hpp

16	$\mathbf{Logitech}$	Wingman	3D	Class	Driver	for	QNX	Class	Documentation

Logitech Wingman 3D Class Driver for QNX File Documentation

5.1 Wingman3D.hpp File Reference

```
#include <sys/hiddi.h>
```

Classes

- struct wingman_data
 - $A\ structure\ for\ holding\ wingman\ joystick\ data.$
- struct wingman report
 - A structure for device reports (for driver internal use).
- struct wingman device
 - A structure for wingman devices (for driver internal use).
- class Wingman3D

The class Wingman3D provides an interface to the following Logitech (Vendor ID: 0x046D) USB HID compliant joysticks:

Typedefs

- typedef wingman_data wingman_data_t
- $\bullet \ \ typedef \ wingman_report \ wingman_report_t$
- typedef wingman device wingman device t

5.1.1 Typedef Documentation

- 5.1.1.1 typedef struct wingman data wingman data t
- 5.1.1.2 typedef struct wingman_report wingman_report_t
- 5.1.1.3 typedef struct wingman device wingman device t

Index

\sim Wingman3D	$\mathrm{hat}\mathrm{Switch}$
Wingman3D, 9	wingman_data, 13
buttons	is Button Pressed
wingman_data, 13	Wingman3D, 10
	isStatusOk
cbtnbuf	Wingman3D, 10
$wingman_report, 15$	G ,
creport	\max_data
$wingman_report, 15$	$wingman_device, 14$
creport_instance	
wingman_report, 15	$\operatorname{printDeviceInfo}$
1-4-	$ m Wingman 3D, \ 10$
data	
wingman_device, 14	report
dev_no	${ m wingman_device, 14}$
wingman_report, 15	1: 1
device_instance	slider
wingman_device, 14	wingman_data, 13
getHatSwitchStatus	statusMessage
Wingman3D, 10	wingman_data, 12
getMaxSliderValue	statusOk
Wingman3D, 10	wingman_data, 12
getMaxTwist	twist
Wingman3D, 9	
getMaxX	$wingman_{data}$, 13
Wingman3D, 9	Wingman3D, 7
getMaxY	~Wingman3D, 9
Wingman3D, 9	getHatSwitchStatus, 10
getNumButtons	getMaxSliderValue, 10
Wingman3D, 9	getMaxTwist, 9
getSliderValue	getMaxX, 9
Wingman3D, 10	getMaxY, 9
getStatusMessage	getNumButtons, 9
Wingman3D, 10	getSliderValue, 10
get Twist	getStatusMessage, 10
Wingman3D, 9	getTwist, 9
get WingmanData	getWingmanData, 10
Wingman3D, 10	getWingmanMaxData, 10
get WingmanMaxData	get X, 9
Wingman3D, 10	getY, 9
get X	isButtonPressed, 10
Wingman3D, 9	isStatusOk, 10
getY	printDeviceInfo, 10
Wingman3D 9	Wingman3D 9

20 INDEX

```
Wingman3D.hpp, 17
    wingman_data_t, 18
    wingman_device_t, 18
    wingman_report_t, 18
wingman_data, 12
    buttons, 13
    hatSwitch, 13
    slider, 13
    statusMessage, 12
    statusOk, 12
    twist, 13
    x, 12
    y, 13
wingman\_data\_t
    Wingman3D.hpp, 18
wingman_device, 14
    data, 14
    device instance, 14
    \max data, 14
    report, 14
wingman\_device\_t
    Wingman3D.hpp, 18
wingman_report, 15
    cbtnbuf, 15
    creport, 15
    {\tt creport\_instance},\, {\color{red} 15}
    dev_no, 15
wingman\_report\_t
    Wingman3D.hpp, 18
х
    wingman_data, 12
y
    wingman_data, 13
```