WiiChuck

Nintendo Wii NunChuck Arduino and chipKit library

Manual

Introduction:

This library has been made to easily interface and use the Nintendo Wii NunChuck with an Arduino or chipKit.

This library will default to I^2C Fast Mode (400 KHz) when using the hardware I^2C interface.

The library has not been tested in combination with the Wire library and I have no idea if they can share pins. Do not send me any questions about this. If you experience problems with pin-sharing you can move the NunChuck SDA and SCL pins to any available pins on your development board. This library will in this case fall back to a software-based, TWI-/I^2C-like protocol which *will* require exclusive access to the pins used.

If you are using a chipKit Uno32 or uC32 and you want to use the hardware I^2C interface you must remember to set the JP6 and JP8 jumpers to the I^2C position (closest to the analog pins).

You can always find the latest version of the library at **http://www.RinkyDinkElectronics.com/**

For version information, please refer to **version.txt**.

Functions:

WiiChuck(SDA, SCL);

The main class constructor.

Parameters: SDA: Pin connected to the SDA-pin of the Nintendo Wii NunChuck
SCL: Pin connected to the SCL-pin of the Nintendo Wii NunChuck

Usage: `WiiChuck myChuck(SDA, SCL); // Start an instance of the WiiChuck class using the hardware I²C int.`

Notes: You can connect the NunChuck to any available pin but if you use any other than hardware I²C pin the library will fall back to a software-based, TWI-like protocol which will require exclusive access to the pins used, and you will also have to use appropriate, external pull-up resistors on the data and clock signals. External pull-up resistors are *always* needed on chipKit boards.

begin();

Initialize the WiiChuck for use.

Parameters: None

Usage: `myChuck.begin(); // Initialize the WiiChuck object`

readData();

Read the data from the WiiChuck.
Must be called before reading any of the joystick, accelerometer or button values.

Parameters: None

Usage: `myChuck.readData(); // Read the data from the WiiChuck`

getJoyX();

Get the current X position of the joystick.

Parameters:	None
Returns:	(int) -100 to 100: The position of the joystick in % from the center
Usage:	`int joyX = myChuck.getJoyX(); // Get the current X position of the joystick`
Notes:	Negative values indicate that the joystick is to the left of the center while positive values indicate that the joystick is to the right.

getJoyY();

Get the current Y position of the joystick.

Parameters:	None
Returns:	(int) -100 to 100: The position of the joystick in % from the center
Usage:	`int joyY = myChuck.getJoyY(); // Get the current Y position of the joystick`
Notes:	Negative values indicate that the joystick is below the center while positive values indicate that the joystick is above center.

getRollAngle();

Get the roll angle of the NunChuck.

Parameters: None

Returns: (int) -179 to 180: Roll angle in degrees

Usage: `int roll = myChuck.getRollAngle(); // Get the current roll angle of the NunChuck`

Notes: Negative values indicate that the NunChuck is rolled to the left while positive values indicate that the NunChuck is rolled to the right.

getPitchAngle();

Get the pitch angle of the NunChuck.

Parameters: None

Returns: (int) -179 to 180: Pitch angle in degrees

Usage: `int pitch = myChuck.getPitchAngle(); // Get the current pitch angle of the NunChuck`

Notes: Negative values indicate that the NunChuck is pitched backwards (towards the lead) while positive values indicate that the NunChuck is pitched forwards.

getAccelX();

Get the raw accelerometer data for the X axis adjusted for offset.

Parameters: None

Returns: (int) -511 to 512

Usage: `int accelX = myChuck.getAccelX(); // Get the X axis data from the accelerometer`

Notes: The range is theoretical. Normal maximum values are usually a fair amount lower.

getAccelY();

Get the raw accelerometer data for the Y axis adjusted for offset.

Parameters: None

Returns: (int) -511 to 512

Usage: `int accelY = myChuck.getAccelY(); // Get the Y axis data from the accelerometer`

Notes: The range is theoretical. Normal maximum values are usually a fair amount lower.

getAccelZ();

Get the raw accelerometer data for the Z axis adjusted for offset.

Parameters: None

Returns: (int) -511 to 512

Usage: `int accelZ = myChuck.getAccelZ(); // Get the Z axis data from the accelerometer`

Notes: The range is theoretical. Normal maximum values are usually a fair amount lower.

checkButtonC();	
Check if the C button is depressed.	
Parameters:	None
Returns:	(boolean) **TRUE** if the button is depressed, otherwise **FALSE**
Usage:	`Boolean myChuck.checkButtonC(); // Check if the C button is depressed`

checkButtonZ();	
Check if the Z button is depressed.	
Parameters:	None
Returns:	(boolean) **TRUE** if the button is depressed, otherwise **FALSE**
Usage:	`Boolean myChuck.checkButtonZ(); // Check if the Z button is depressed`