bendlabs

ADS Driver Porting Guide

ADS DRIVER

Porting Guide

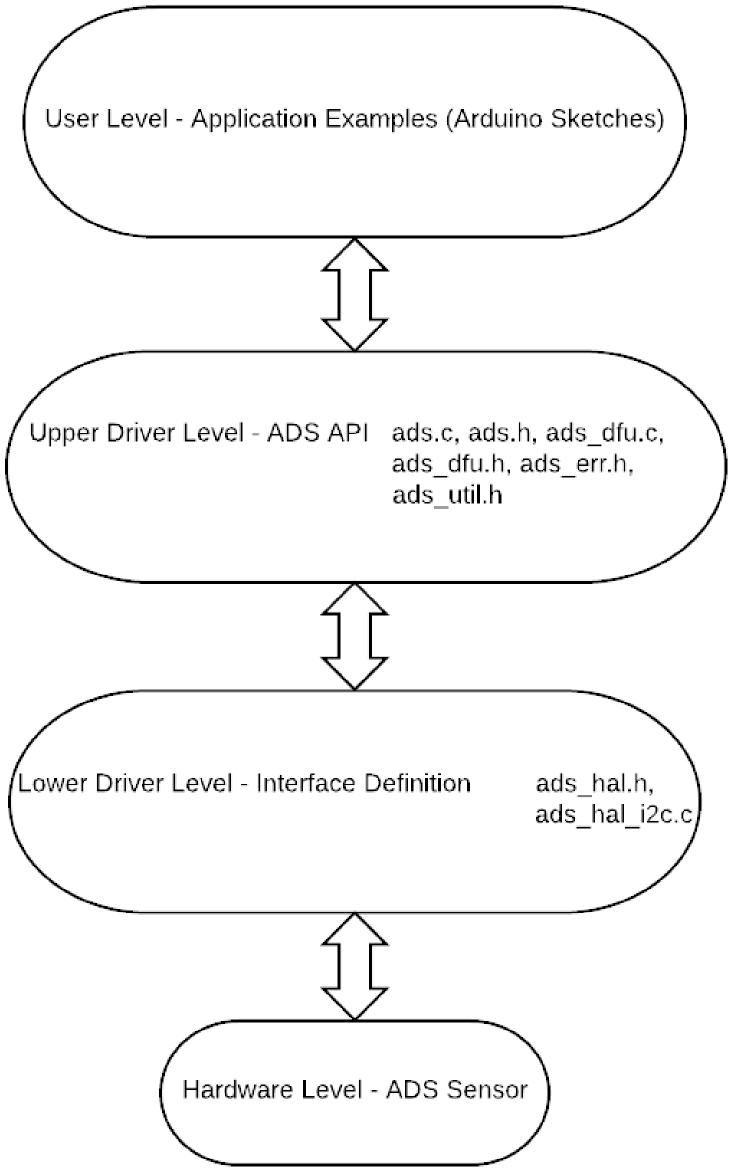
Table of Contents

Introduction

This guide is intended to assist users of the Bend Labs angular displacement sensors (ads sensors) to interface the ads_driver with their MCU.

This Guide describes how to port the ads driver to the users MCU. The source code for the portable driver can be found at https://github.com/bendlabs/one_axis_ads.

The chart below depicts the levels of the driver necessary for porting the driver.


User Level - Application Examples (Arduino Sketches)
Upper Driver Level - ADS API
ads.c, ads.h, ads_dfu.c, ads_dfu.h, ads_err.h, ads_util.h
Lower Driver Level - Interface Definition
ads_hal.h, ads_hal_i2c.c
Hardware Level - ADS Sensor

Driver Contents

File	Description	Comments
ads.c	ADS sensor API	It is not recommended to change the contents of this file.
ads_hal_i2c.c	Platform-specific functions	It is necessary to implement the functions ads_hal_interrupt, ads_hal_pin_int_init, ads_hal_gpio_pin_write, ads_hal_delay, ads_hal_pin_int_enable, ads_hal_write_buffer, ads_hal_read_buffer, ads_hal_reset, and ads_hal_i2c_init based on your own platform.
ads_dfu.c	API to update the firmware on the ADS sensor	It is not recommended to change the contents of this file.
ads.h	ADS sensor API	It is not recommended to change the contents of this file.
ads_hal.h	Platform-specific functions	
ads_dfu.h	API to update the firmware on the ADS sensor	It is not recommended to change the contents of this file.
ads_fw.h	ADS firmware for dfu (device firmware update)	It is not recommended to change the contents of this file.
ads_err.h	Enumeration of API error codes	It is not recommended to change the contents of this file.
ads_util.h	Command and ID definitions	It is not recommended to change the contents of this file.

Porting Instructions

The reference examples provided were coded for the Arduino Development Environment.

To port the driver to a new platform, modify the contents of ads_hal_i2c.c as follows:

Remove the following lines that are Arduino specific:
#include “Arduino.h”
 #include “Wire.h”

Re-implement the low-level hardware specific functions based on the new platform. Keep the alignment of the input parameters and the return format of the following functions with their current declaration.
ads_hal_interrupt()
ads_hal_pin_int_init()
ads_hal_gpio_pin_write()
ads_hal_delay()
ads_hal_pin_int_enable()
ads_hal_write_buffer()
ads_hal_read_buffer()
ads_hal_reset()
ads_hal_i2c_init()

Hardware Specific Functions

Function	Inputs	Return	Comments
ads_hal_interrupt			Interrupt service routine. Handles data ready interrupt from falling edge ads interrupt line (nDRDY). Reads data from ADS sensor and returns data through callback to ads.c.
ads_hal_pin_int_init			Attach falling interrupt to pin number assigned to ADS_INTERRUPT_PIN. Set ads_hal_interrupt as the interrupt service routine.
ads_hal_gpio_pin_write	u8 pin, u8 val		Writes digital value (val) to gpio number (pin).
ads_hal_delay	u16 ms		Delays processor for number of milliseconds equal to ms.

ads_hal_pin_int_enable	bool enable		Enables/disables the interrupt attached to ADS_INTERRUPT_PIN.
ads_hal_write_buffer	u8* buffer, u8 len	ADS_OK, ADS_ERR	Writes len number of bytes contained in buffer to the I2C address contained in the local variable _address. Returns ADS_OK if the write operation is successful, and ADS_ERR if the operation failed.
ads_hal_read_buffer	u8* buffer, u8 len	ADS_OK, ADS_ERR	Reads len number of bytes into buffer from the I2C address contained in the local variable _address. Returns ADS_OK if the read operation is successful, and ADS_ERR if the operation failed.
ads_hal_reset			Configures ADS_RESET_PIN as an output, drives reset low, waits 10 ms, brings ADS_RESET_PIN high, then configures ADS_RESET_PIN as an input with pullup enable.
ads_hal_i2c_init			Configures I2C bus used by the ADS sensor as a master with 400kHz or lower clock frequency. Enable slave clock stretching if available.