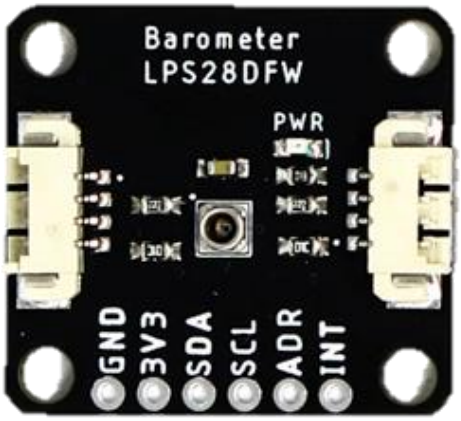
SmartElex Absolute Digital Barometer - LPS28DFW



The Absolute Digital Barometer - LPS28DFW offer a unique barometer breakout featuring the LPS28DFW from STMicroelectronics©. The LPS28DFW is an absolute barometer with a water-resistant package making it perfect for pressure measurement applications where the sensor is exposed to or even submerged in water[1].

Parameter	Min.	Typ.	Max.	Units	Notes
Supply Voltage	1.7	-	3.6	V	Breakouts run the sensor at 3.3V
Supply Current	-	1.7	-	µA	Average Selection (AVG)=4 and Output Data Rate (ODR)=1Hz.
	-	9.4	-		AVG=128 and ODR=1Hz.
	-	0.9	-		Sensor in power-down mode.
Operating Temperature Range	-40	-	+85	°C	
Operating Pressure Range					
Mode 1	260	-	1260	hPa	
Mode 2	260	-	4060		
Pressure Sensitivity					
Mode 1	-	4096	-	LSB/hPa	
Mode 2	-	2048	-		
Relative Pressure Accuracy					Test Conditions:
Mode 1	-	±0.015	-	hPa	Temp. = 25°C Press.=800~1100hPa
Mode 2	-	±1	-		Temp. = 25°C Press. = 2060~4060hPa

The sensor has two full-scale measurement ranges of 260 - 1260hPa and 260 - 4060hPa with an absolute pressure accuracy of 0.5hPa. The LPS28DFW is composed a piezoresistive pressure sensor with a metal lid and gel encasing to protect the sensing elements from water and other environmental hazards.

In this guide we'll cover the features and specifications of the LPS28DFW and other hardware present on these breakouts as well as the Arduino library to interact with the sensor.

1. Important! While the LPS28DFW is protected from water, the rest of the components on these breakouts are not protected by any conformal coating and can be damaged by exposure to liquids. Users who intend to use these breakouts in applications where the board may be exposed to water or other liquids should apply conformal coating to the board prior to use.

LPS28DFW Absolute Pressure Sensor

The LPS28DFW from STMicroelectronics is a digital output absolute pressure sensor with a gel-filled metal lid protecting the sensing element from moisture making it ideal for applications such as water depth measurements or other pressure-sensing projects in wet environments.

The LPS28DFW has two user-selectable measurement ranges (260 to 1260hPa and 260 to 4060hPa) with an absolute pressure accuracy of 0.5hPa and supports output data rates of 1 to 200Hz. The sensor supports communication over I^2C and MIPI $I3C^{SM}$ interfaces (though I3C communication is not covered in this guide or the Arduino Library). The table below outlines some of the parameters for the LPS28DFW. For a complete overview of the sensor, refer to the datasheet.

I^2C Interface

The standard size routes the I^2C interface to a pair of connectors as well as a 0.1"-spaced PTH header for users who prefer a traditional soldered connection. Both breakouts route the sensor's interrupt (INT) pin to a PTH pin.

Both boards set the LPS28DFW's I^2C address to **0x5C** by default. Adjust the ADR jumper to change to the alternate address (**0x5D**) or open it completely to toggle the address using the ADR PTH pin (Standard size only). More information on this jumper in the Solder Jumpers section below.

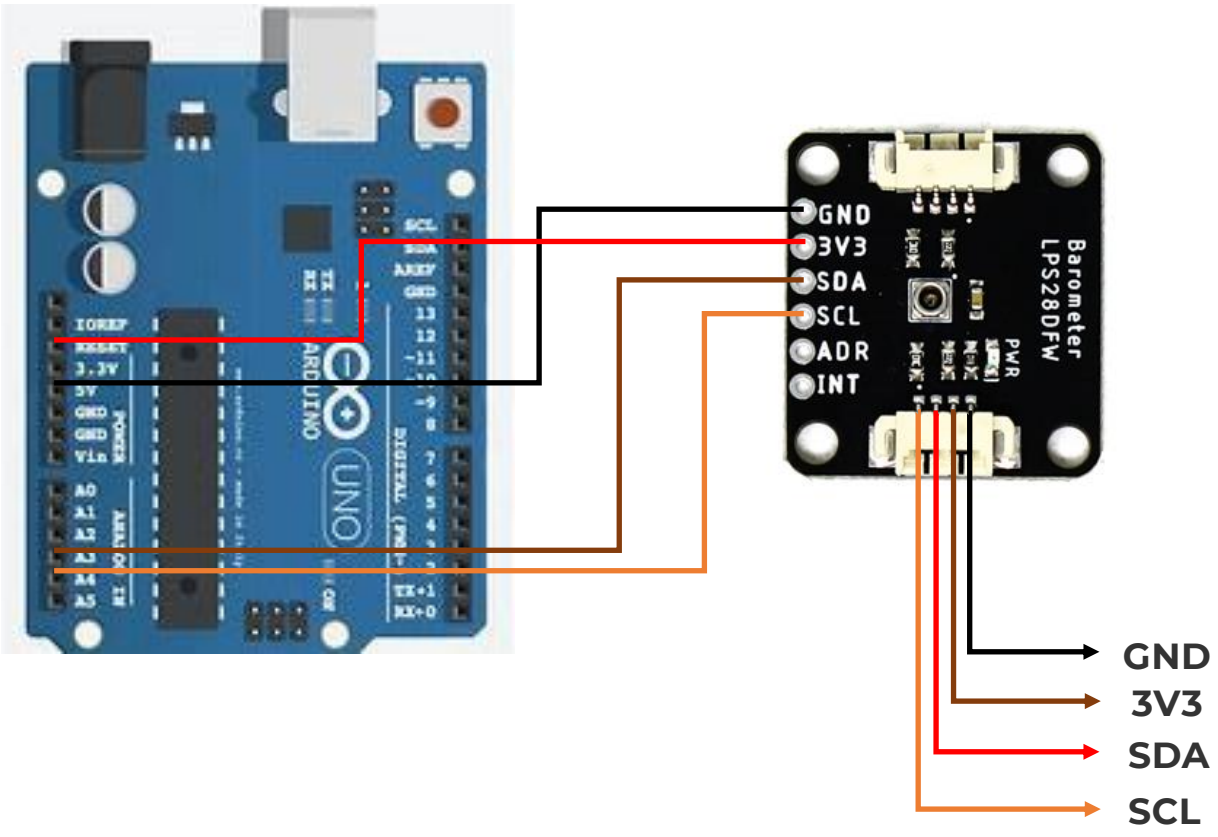
Solder Jumpers

Both LPS28DFW breakouts have three solder jumpers labeled: **PWR**, **I2C**, and **ADR**. The table below outlines the jumpers' label, default state, function, and any notes about their behavior.

Label	Default State	Function	Notes
PWR	CLOSED	Completes the Power LED circuit.	Open to disable the Power LED.
I2C	CLOSED	Pulls the SDA/SCL lines to VCC (3.3V) through a pair of 10kΩ resistors.	Open to disable pull-up resistors on the I²C lines.
ADR	SEE NOTE	Sets the I²C address of the LPS28DFW.	I²C address is 0x5C by default. Sever the trace connecting the center pad to the pad labeled 0x5C and connect it to the pad labeled 0x5D to change the address.

Wiring

Connecting the LPS28DFW to Arduino:



Arduino	LPS28DFW
SCL(A5)	SCL
SDA(A4)	SDA
3.3v	3V3
GND	GND

Conformal Coating for Waterproofing

While the LPS28DFW's gel filled cap protects the sensing element from liquid and other environmental effects, the breakouts do not have any coating to protect the other components from damage due to exposure to liquids. A protective coating is required for applications that expose the board(s) to liquid.

LPS28DFW Arduino Library

Library Functions

The list below outlines and describes the functions available in the SparkFun LPS28DFW Arduino Library. For detailed information on the parameters and use of all functions, refer to the .cpp file in the library.

Device Initialization and Configuration

- 'int32_t begin(uint8_t address = LPS28DFW_I2C_ADDRESS_DEFAULT, TwoWire& wirePort = Wire);' - Begin communication with the sensor over I^2 at the defined address and on the defined port. If no error occur, perform a soft reset and initialize the sensor.
- 'int32_t init();' - Enables the BDU and IF_ADD_INC bits in the control registers.
- 'int32_t boot();' - Enables the BOOT bit in the control registers
- 'int32_t reset();' - Resets the sensor.
- 'int32_t setModeConfig(lps28dfw_md_t* mode);' - Configures the operation mode settings for the sensor including range and ODR.
- 'int32_t getModeConfig(lps28dfw_md_t* mode);' - Returns the operation mode settings.
- 'int32_t getStatus(lps28dfw_stat_t* status);' - Returns the sensor status bits such as data ready, overrun, etc.

Sensor Data

- 'int32_t getSensorData();' - Get pressure data from the sensor.

Interrupt Control and Feature Selection

- 'int32_t setInterruptMode(lps28dfw_int_mode_t* intMode);' - Configures the interrupt pin to be either HIGH/LOW and LATCHED/PULSED.
- 'int32_t enableInterrupts(lps28dfw_pin_int_route_t* intRoute);' - Enables the data ready and FIFO interrupt conditions.
- 'int32_t getInterruptStatus(lps28dfw_all_sources_t* status);' - Returns the status of the interrupt flags.

FIFO Buffer Control

- 'int32_t setFIFOConfig(lps28dfw_fifo_md_t* fifoConfig);' - Sets the FIFO configuration parameters.
- 'int32_t getFIFOConfig(lps28dfw_fifo_md_t* fifoConfig);' - Returns settigs of FIFO buffer.
- 'int32_t getFIFOLength(uint8_t* numData);' - Returns the number of samples stored in the FIFO buffer (up to 128).
- 'int32_t getFIFOData(lps28dfw_fifo_data_t* data, uint8_t numData);' - Gets pressure data from the FIFO buffer.
- 'int32_t flushFIFO();' - Clear all data from the FIFO buffer.

Reference Mode Control

- 'int32_t setReferenceMode(lps28dfw_ref_md_t* mode);' - Sets the sensor to operate in reference mode. When called it stores the latest pressure data as a reference pressure. The reference pressure can be used with Threshold Mode to trigger interrupts.
- 'int32_t setThresholdMode(lps28dfw_int_th_md_t* mode);' - Configures the sensor to trigger interrupts when the pressure measured exceeds a threshold relative to the defined reference pressure.
- 'int32_t getReferencePressure(int16_t* pressRaw);' - Returns the value stored for the reference pressure.

Arduino Example

Now let's take a closer look at a few of the examples included in the LPS28DFW Arduino Library.

Example - Basic Readings

The first example covers the basics of polling the LPS28DFW for pressure and temperature data over I^2C. Open the example by navigating to **File > Examples > SparkFun LPS28DFW Arduino Library > Example1_BasicReadings**. Select your Board and Port and click the Upload button. Once upload completes, open the serial monitor with the baud set to **115200** and watch pressure and temperature data print out.

```
#include <Wire.h>

#include "SparkFun_LPS28DFW_Arduino_Library.h"

// Create a new sensor object

LPS28DFW pressureSensor;

// I2C address selection

uint8_t i2cAddress = LPS28DFW_I2C_ADDRESS_DEFAULT; // 0x5C

//uint8_t i2cAddress = LPS28DFW_I2C_ADDRESS_SECONDARY; // 0x5D

void setup()

{

  // Start serial

  Serial.begin(115200);

  Serial.println("LPS28DFW Example 1 - Basic Readings!");



  // Initialize the I2C library

  Wire.begin();

  // Check if sensor is connected and initialize

  // Address is optional (defaults to 0x5C)

  while(pressureSensor.begin(i2cAddress) != LPS28DFW_OK)

  {

    // Not connected, inform user
```

```
      Serial.println("Error: LPS28DFW not connected, check wiring and I2C address!");


      // Wait a bit to see if connection is established
      delay(1000);
    }
    Serial.println("LPS28DFW connected!");
}
void loop()
{
    // Get measurements from the sensor. This must be called before accessing
    // the pressure data, otherwise it will never update
    pressureSensor.getSensorData();


    // Print temperature and pressure
    Serial.print("Temperature (C): ");
    Serial.print(pressureSensor.data.heat.deg_c);
    Serial.print("\t\t");
    Serial.print("Pressure (hPa): ");
    Serial.println(pressureSensor.data.pressure.hpa);


    // Only print every second
    delay(1000);
}
```

The code assumes the sensor uses the default I^2C address so if you have adjusted the ADR jumper to switch to the alternate address, comment/uncomment the line with the correct value listed

The example attempts to initialize the sensor with default settings in I^2C at the specified address. If it cannot initialize properly, the code prints out an error in over serial

If you see this error, double check the sensor is connected properly and set to the correct I^2C address and reset the development board or re-upload the code.

The main loop gets temperature and pressure data measurements from the sensor every second

Try moving the sensor up and down to see the pressure data change.