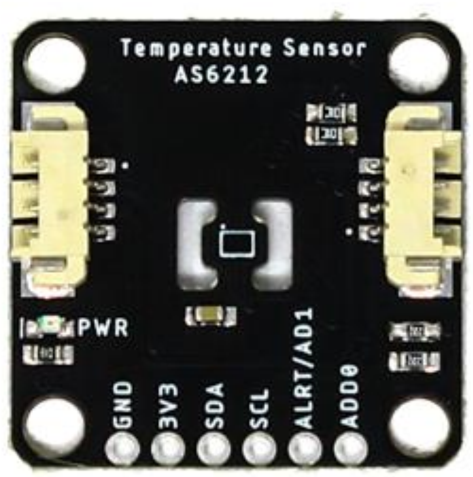
SmartElex Digital Temperature Sensor - AS6212

The AS6212 is a highly accurate and power efficient digital temperature sensor with a wide temperature sensing range (-40°C to 125°C) from ams AG. The AS6212 boasts a host of features including a configurable alert pin that can trigger when temperature data exceeds user-defined temperature thresholds. Read more on configuring the alert pin and temperature thresholds in the Arduino sections of this guide as well as in the AS6212 Datasheet. The AS6212 also features a sleep/low power mode that works in tandem with a Single Shot measurement mode to wake the device, retrieve stored temperature data and return it to sleep mode.



The AS6212 accepts a supply voltage between **1.7V** and **3.6** and typically consumes **6µA** during normal temperature conversions and **0.1µA** on standby. The AS6212 normally runs at **3.3V** and receives power either through the connectors or the dedicated **3.3V** and **GND** PTH pins.

The breakout design isolates the AS6212 from the rest of the PCB as much as possible to minimize ambient heat from interfering with temperature data. The table below outlines the temperature data accuracy across the AS6212's temperature measurement range:

Temperature Range	Temperature Accuracy
−10°C to 65°C	±0.2°C
−40°C to −10°C and 65°C to 85°C	±0.3°C
85°C to 125°C	±0.5°C

I^{2}C Interface

As the name of this breakout suggests, the board routes the AS6212's I^{2}C pins to a pair of connectors as well as a 0.1"-spaced PTH header for users who prefer a soldered connection. The AS6212 supports both fast (max 400kHz) and high-speed (max 3.4MHz) clock frequencies and has eight configurable I^{2}C addresses (default is **0x48**). Select the address by adjusting the labeled jumpers.

The Alert/AD1 and AD0 pins are also broken out to the same PTH header as the I^{2}C pins to interact with. The Alert pin can be enabled to act as an external hardware interrupt for an attached microcontroller.

Solder Jumpers

The Digital Temperature Sensor Breakout - AS6212 (has four solder jumpers labeled **LED**, **I2C**, **AD0** and **AD1**. The **LED** jumper connects the Power LED anode to **3.3V** via a **1kΩ** resistor. The jumper is CLOSED by default. Open the jumper to disable the Power LED and reduce the total current draw of the board. The **I2C** jumper ties the SDA and SCL lines to **3.3V** via a pair of **2.2kΩ** resistors and is CLOSED by default. Open the jumper to disable the pull-up resistors.

Note: Recommended practice suggests to only have a single pair of pull-up resistors on an I^{2}C bus to avoid creating too strong of a parallel resistance on the bus. If you disable the pull-ups on this breakout to use a separate pair, make sure the entire bus is operating at the appropriate logic level (in this case, **3.3V**) or the lines are properly shifted to avoid damaging this or other devices on the bus.

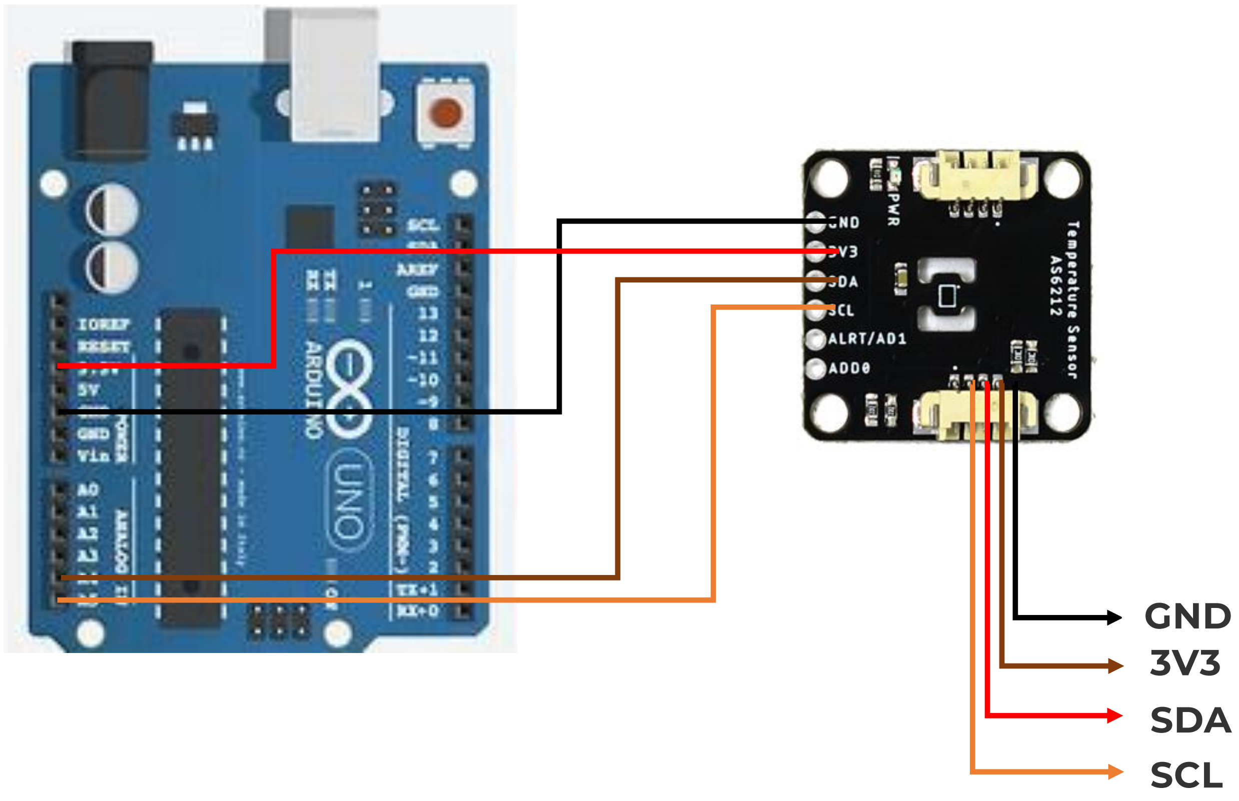
The **AD0** and **AD1** jumpers control the I^{2}C address as well as enabling/disabling the Alert pin. By default, these two-way jumpers connect the AD0 pin to **Ground** and the Alert/AD1 pin to **3.3V** to enable the Alert pin and set the I^{2}C address to **0x48**.

ALERT/AD1 Jumper Net	**AD0 Jumper Net**	**Alert Pin Functionality**	**I^{2}C Address**
VCC (PU)	GND	Enabled	0x48 (**Default**)
VCC (PU)	VCC	Enabled	0x49
VCC (PU)	SDA	Enabled	0x4A
VCC (PU)	SCL	Enabled	0x4B
SCL	GND	Disabled	0x44
SCL	VCC	Disabled	0x45
SCL	SDA	Disabled	0x46
SCL	SCL	Disabled	0x47
GND	GND	Disabled	0x48
GND	VCC	Disabled	0x49
GND	SDA	Disabled	0x4A
GND	SCL	Disabled	0x4B

Adjust these jumpers to change the address and/or disable the Alert pin. The table above outlines the various settings these jumpers can be set to.

Wiring & Test

You can easily wire this breakout to any microcontroller, we'll be using an Arduino. For another kind of microcontroller, just make sure it has I2C, then port the code.



Arduino	AS6212
SCL(A5)	SCL
SDA(A4)	SDA
3.3v	3V3
GND	GND

- Connect **Vin** to the power supply 3.3V.
- Connect **GND** to common power/data ground
- Connect the **SCL** pin to the I2C clock **SCL** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A5**, on a Mega it is also known as **digital 21** and on a Leonardo/Micro, **digital 3**

- Connect the **SDA** pin to the I2C data **SDA** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A4**, on a Mega it is also known as **digital 20** and on a Leonardo/Micro, **digital 2**

The **AS6212** has a default I2C address of **0x48**

AS6212 Arduino Library

The AS6212 Arduino Library helps users configure and pull temperature data from the AS6212. Install the library through the Arduino Library Installer by searching for **"** SparkFun **AS6212"**.

Library Functions

The list below outlines the functions included in the AS6212 Arduino Library along with short descriptions of what they do.

Class

Construct the `AS6212` object in the global class. The examples use `sensor` as the AS6212 breakout object.

- `AS6212 sensor;`

Device Setup and Settings

- `bool begin(uint8_t sensorAddress, TwoWire &wirePort);` - Initialize the AS6212 board at a specified address on a selected port. If left empty, default values are used for the address (**0x48**) and Wire port.
- `bool setDefaultSettings();` - Set the AS6212 CONFIG register to default/factory settings. This helps quickly return the sensor to default settings if anything in the CONFIG register has been changed as adjustments to the CONFIG register remain through power cycles.
- `uint8_t getAddress();` - Returns the device's I^{2}C address.
- `bool getAlertStatus();` - Returns the status of the Alert bit.
- `void setConsecutiveFaults(int faults);` - Sets the number of consecutive faults (temperature above `THigh`) to occur before the Alert pin state adjusts. Acceptable values are 1,2,3 and 4.
- `uint8_t getConsecutiveFaults();` - Returns the value set for `setConsecutiveFaults();`.

- `void setInterruptMode(bool mode);` - Set the AS6212 Alert pin to operate in Interrupt mode.
- `bool getInterruptMode();` - Read whether the Alert pin is set to operate in Interrupt mode.
- `void setConversionCycleTime(uint8_t cycleTime = AS6212_CONVERSION_CYCLE_TIME_250MS);` - Sets the time between temperature conversions in milliseconds. Acceptable entries are: 125MS, 250MS, 1000MS or 4000MS.
- `uint16_t getConversionCycleTime();` - Returns the value set for Conversion Cycle Time in milliseconds.
- `void setAlertPolarity(bool polarity);` - Set the polarity of the Alert pin output to go either `HIGH` or `LOW` when triggered. Default is `LOW`/`0`.
- `bool getAlertPolarity();` - Returns the value set for the Alert polarity bit. `0` for active LOW or `1` for active HIGH.
- `void sleepModeOn();` - Put the AS6212 into Sleep Mode. The device must be in Sleep Mode for Single Shot measurements to be made.
- `void sleepModeOff();` - Clears the Sleep Mode bit in the config register and after reseting the SM bit to `0` the device returns to continuous conversion mode.
- `bool getSleepMode();` - Returns the Sleep Mode bit status as a boolean.
- `void triggerSingleShotConversion();` - Tell the AS6212 to perform a Single Shot temperature conversion.
- `bool getSingleShotStatus();` - Returns the Single Shot mode bit status as a boolean. `0` for no conversion ongoing/conversion finished. `1` for start Single Shot conversion/conversion ongoing.
- `void setConfig(uint16_t targetState);` - Legacy function for users who wish to interact directly with the CONFIG register. Refer to section 6.2 in the AS6212 Datasheet for a detailed description of this regsiter and the adjustable bits in it.
- `uint16_t readConfig();` - Returns the settings in the CONFIG register as an unsigned integer.

Temperature Data

- `float readTempC();` - Returns the recorded temperature in degrees Celsius.
- `float getTLowC();` - Returns the temperature value set for `setTLowC`.
- `bool setTLowC(int16_t lowLimit);` - Sets the temperature in °C for low temperature threshold. Used for the alert pin temperature limits.
- `float getTHighC();` - Returns the temperature value set for `setTHighC`.

- bool setTHighC(int16_t highLimit); - Sets the temperature in °C for the high temperature threshold. Used for the alert pin temperature limits.
- float readTempF(); - Returns the recorded temperature in degrees Fahrenheit.
- float getTLowF(); - Returns the temperature value set for setTLowF.
- bool setTLowF(int16_t lowLimit); - Sets the temperature in °F for low temperature threshold. Used for the alert pin temperature limits.
- float getTHighF(); - Returns the temperature value set for setTHighF.
- bool setTHighF(int16_t highLimit); - ets the temperature in °F for high temperature threshold. Used for the alert pin temperature limits.

Arduino Examples

The SparkFun AS6212 Arduino Library includes ten examples to showcase the various capabilities and settings of the AS6212. In this section we'll take an in-depth look at most of those examples and highlight pertinent bits of code where necessary. The examples build on each other so it may help to go through them in sequential order.

Prior to uploading the examples, let's take a quick look at the setup function used in all examples:

```
COPY CODEif (sensor.begin() == false)
{
   Serial.println("AS6212 Qwiic failed to respond. Please check wiring and possibly the I
2C address. Freezing...");
   while (1);
};
```

After uploading any of the examples, open the Arduino serial monitor with the baud set to **115200**. If the AS6212 fails to initialize on the bus, the code freezes and prints out "AS6212 Qwiic failed to respond. Please check wiring and possibly the I2C address. Freezing...".

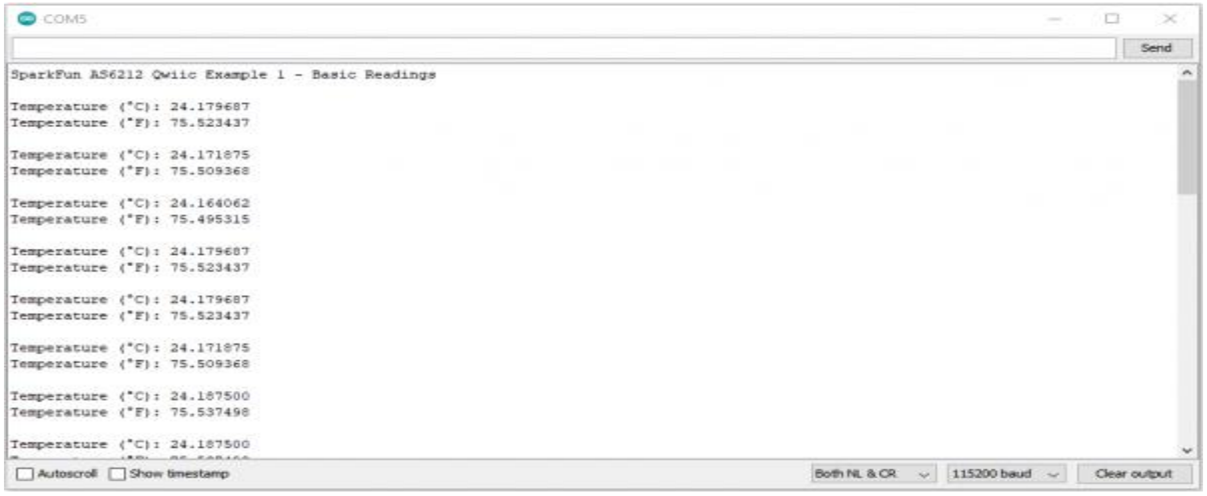
If you see this message, check the connection between the breakout and controller and make sure the AS6212 is either set to the default address or the sensor.begin(); function is adjusted to the correct address. Refer to Example 2 - Different I2C Address for a quick demonstration of initializing the AS6212 on an alternate address.
Each example also includes a quick call of the setDefaultSettings(); function to return the AS6212 to default settings as adjustments to the CONFIG register (e.g. conversion cycle time, interrupt mode, alert pin polarity) remain through power cycles.

Example Code

```
#include "SparkFun_AS6212_Qwiic.h" //Click here to get the library: http://librarymanager/All#SparkFun_AS6212

#include <Wire.h>


AS6212 sensor;


//Initialize temperature variables as floats

float tempC;

float tempF;


void setup(){


  Serial.begin(115200);


  Serial.println("SparkFun AS6212 Qwiic Example 1 - Basic Readings");


  Wire.begin();


  // Check to see if AS6212 Qwiic is present on the bus

  // Note, here we are calling begin() with no arguments = defaults (address:0x48, I2C-port:Wire)

  if (sensor.begin() == false)

  {

    Serial.println("AS6212 Qwiic failed to respond. Please check wiring and possibly the I2C address. Freezing...");

    while (1);

  };
}

void loop(){

  tempC = sensor.readTempC();

  tempF = sensor.readTempF();



  Serial.println();

  Serial.print("Temperature (°C): ");

  Serial.print(tempC, 6);          //Reads out 6 characters of the temperature float

  Serial.print("\tTemperature (°F): ");

  Serial.println(tempF, 6);          //Reads out 6 characters of the temperature float



  delay(1000);

}
```

Example 1 - Basic Readings

The first example in the library demonstrates how to initialize the AS6212 on the I^{2}C bus and retrieve temperature data from the sensor in both °C and °F. Open the example by navigating to **File > Examples > SparkFun AS6212 Arduino Library > Example_01_BasicReadings**. Select the appropriate Board and Port and click upload. Assuming the upload was successful, open the serial monitor with the baud set to **115200**.



```
SparkFun AS6212 Qwiic Example 1 - Basic Readings

Temperature (°C): 24.179687
Temperature (°F): 75.523437

Temperature (°C): 24.171875
Temperature (°F): 75.509368

Temperature (°C): 24.164062
Temperature (°F): 75.495315

Temperature (°C): 24.179687
Temperature (°F): 75.523437

Temperature (°C): 24.179687
Temperature (°F): 75.523437

Temperature (°C): 24.171875
Temperature (°F): 75.509368

Temperature (°C): 24.187500
Temperature (°F): 75.537498

Temperature (°C): 24.187500
```

After initializing, the code prints out temperature data recorded by the AS6212 in both °C and °F every second. Try breathing on the sensor or gently press your finger to it and watch the temperature data change.