Overview

The ALS31313 is a linear hall effect sensor,

Objective

Program the I2C address to be 0x41

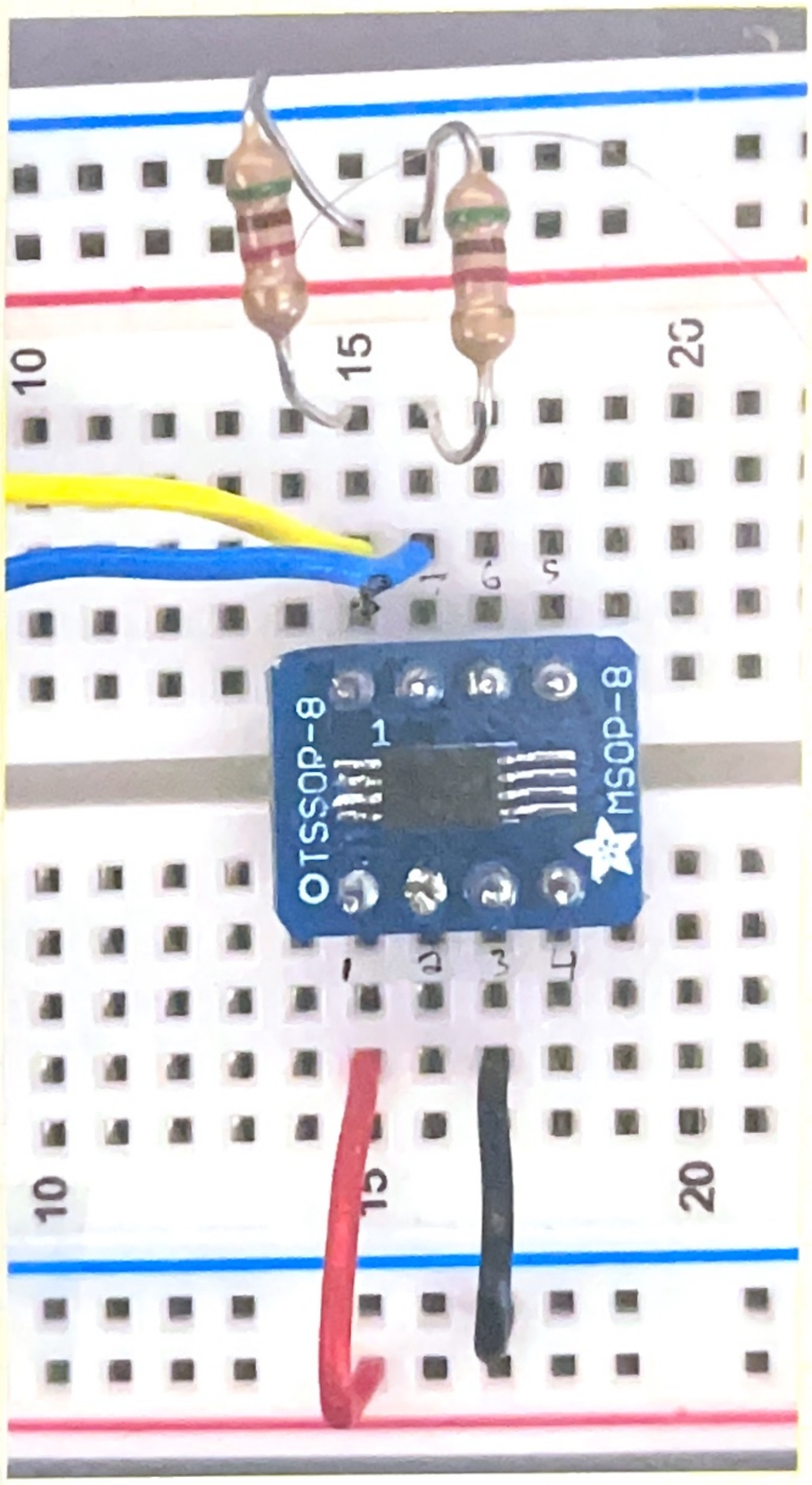
Pinout and Circuit

PINOUT DIAGRAM AND TERMINAL LIST TABLE

Package LE, 8-Pin TSSOP Pinout Diagram

Terminal List Table

Number	Name	Function
1	VCC	Power supply input. Bypass VCC to GND with a 0.1 µF capacitor.
2	ADR0	I²C Address Select 0. Connect a resistive divider to ADR0 to select the device address. See Application Information section on addressing for more information.
3	GND	Ground signal terminal.
4	$\overline{INT}$	Interrupt output. See Application Information section on interrupt function for more information.
5	NC	Not internally connected. Connect to GND.
6	ADR1	I²C Address Select 1. Connect a resistive divider to ADR1 to select the device's address. See Application Information section on addressing for more information.
7	SDA	I²C serial data input/output. Open-drain.
8	SCL	I²C serial clock input



Programming the ALS31313

6/19/2023
Wesley Cash

I^2C Addressing Options

I²C Addressing

Table 19 outlines the different addresses available to the ALS31300. In the special case where AD0 and AD1 are both tied to VCC, the device will respond to the slave address stored in register 0x02 (bits 10:16). From the factory, this is set to 7b000000, with the bit following the address indicating a read or write per the I²C specification. Note: Different values for the three MSBs of the address bits (A6, A5, and A4) are available for factory programming if a conflict with other units occurs in the application design.

Table 19: I²C Slave Address Decoding

Voltage on AD1, V_{A1} (× V_{CC})	Voltage on AD0, V_{A0} (× V_{CC})	4-Bit Code from ADR1 and ADR0 Voltages				Slave Address Bits							Slave Address
		E3	E2	E1	E0	A6	A5	A4	A3	A2	A1	A0	
0	0	0	0	0	0	1	1	0	0	0	0	0	96
	0.33	0	0	0	1	1	1	0	0	0	0	1	97
	0.67	0	0	1	0	1	1	0	0	0	1	0	98
	1	0	0	1	1	1	1	0	0	0	1	1	99
0.33	0	0	1	0	0	1	1	0	0	1	0	0	100
	0.33	0	1	0	1	1	1	0	0	1	0	1	101
	0.67	0	1	1	0	1	1	0	0	1	1	0	102
	1	0	1	1	1	1	1	0	0	1	1	1	103
0.67	0	1	0	0	0	1	1	0	1	0	0	0	104
	0.33	1	0	0	1	1	1	0	1	0	0	1	105
	0.67	1	0	1	0	1	1	0	1	0	1	0	106
	1	1	0	1	1	1	1	0	1	0	1	1	107
1	0	1	1	0	0	1	1	0	1	1	0	0	108
	0.33	1	1	0	1	1	1	0	1	1	0	1	109
	0.67	1	1	1	0	1	1	0	1	1	1	0	110
	1	1	1	1	1	x	x	x	x	x	x	x	Set to 0 at the factory

There are a few ways to set the address on the ALS31313 chip.

The first way is based on the voltage on the ADDR 0 and ADDR 1 pins. In our case we left them open or "0" voltage. In this case our address will be 0b 0 110 0000 or 0x 60.

The other method involves using a programed address from the memory of the chip. To use this number, set ADDR0 and ADDR 1 to high.

MEMORY MAP

The memory map below lists the locations of accessible registers on the ALS31313. See the following sections on EEPROM and Primary Registers for detailed information.

Reserved	Read Only	Read/Write Volatile	Read/Write EEPROM	Read/Write 1 to Clear	Clear on Read

Table 1: Memory Map

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02	RESERVED								BW Select			Hall Mode		I²C CRC Enable	Disable Slave ADC	Slave Address							I²C Threshold	Channel Z Enable	Channel Y Enable	Channel X Enable	INT Latch Enable	Customer EE				
0x03	RESERVED								Signed INT Enable	INT Mode	INT EEPROM Status	INT EEPROM Enable	Z INT Enable	Y INT Enable	X INT Enable	Z INT Threshold					Y INT Threshold						X INT Threshold						
0x0D	RESERVED								Customer EEPROM																							
0x0E	RESERVED								Customer EEPROM																							
0x0F	RESERVED								Customer EEPROM																							
0x27	RESERVED																								Low Power Counter		I²C Loop Mode				Sleep	
0x28	X_Axis_MSBs								Y_Axis_MSBs							Z_Axis_MSBs								New Data	INT	Temperature MSBs						
0x29	RESERVED											INT Write	X_Axis_LSBs			Y_Axis_LSBs					Z_Axis_LSBs			Hall Status		Temperature LSBs						
Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Notice that the slave address is in register 0x02 and occupies bits 16:10. Note that this is split between two bytes.

31:24 23:16 15:8 7:0
~~Byte~~ MSB LSB

00000000 0000000**0** 0000000**0** 00000000
16th bit 10th bit

Programming the ALS31313 I2C Address

06/19/2023 Wesley Ocola

EEPROM

The following EEPROM addresses are customer accessible and may be read at any time, with or without entering the customer access code. Customer Access mode must be enabled to write to any of these registers.

Reserved	Read Only	Read/Write Volatile	Read/Write EEPROM	Write 1 to Clear	Clear on Read

Table 2: EEPROM 0x02

Address	Bits	Default	Name	Description
0x02	31:24	0	Reserved	Reserved
	23:21	0	BW Select	Used to control the sample rate of the device. Resolution can be traded for faster samples. 0 = Slowest sample rate, highest resolution 7 = Fastest sample rate, lowest resolution See *Bandwidth Selection* section.
	20:19	See Selection Guide	Hall Mode	Controls the operation mode of the Hall plates. 0 = Single-Ended Hall Mode 1 = Differential Hall Mode 2 = Common Hall Mode 3 = Alternating Hall mode (switches between Differential and Common Hall Modes for each conversion per enabled axis. See *Hall Modes* section.
	18	0	I²C CRC Enable	I²C Cyclic Redundancy Check (CRC) output byte enabled. Enable CRC for applications that require high data integrity. 0 = Disabled 1 = Enabled See *CRC* section
	17	0	Disable Slave ADC	Disable the external slave address pins. When set, the EEPROM setting in Slave Address is used to determine the slave address. See *I²C Addressing* section.
	16:10	0	Slave Address	Used to set the slave address for the device when either Disable Slave ADC is set, or the voltages on the slave address pins are set to V_{CC}. See *I²C Addressing* section.
	9	1	I²C Threshold	Enables 1.8 V or 3 V compatible I²C. 0 = 3 V compatible mode (Increases threshold for logic input high level) 1 = 1.8 V compatible mode
	8	1	Channel Z Enable	Enables the Z channel. Disable for faster update rate if this axis is not needed.
	7	1	Channel Y Enable	Enables the Y channel. Disable for faster update rate if this axis is not needed.
	6	1	Channel X Enable	Enables the X channel. Disable for faster update rate if this axis is not needed.
	5	0	$\overline{INT}$ Latch Enable	Enables volatile latching of the $\overline{INT}$ signal. When set, if an interrupt event occurs, the $\overline{INT}$ status bit and $\overline{INT}$ output will both remain latched even after the event goes away. See *Interrupt* section.
	4:0	0	Customer EEPROM	Customer non-volatile EEPROM. Can be used to store any customer information. Does not affect device operation.

Notice that to write to the 0x02, we have to use the Customer Access Mode...

Customer Write Access

An access code must be sent to the device prior to writing any of the volatile registers or EEPROM in the ALS31313. If customer access mode is not enabled, then no writes to the device are allowed. The only exception to this rule is the sleep register, which can be written regardless of the access mode. Furthermore, any register or EEPROM location can be read at any time regardless of the access mode.

To enter customer access mode, an access command must be sent via the I²C interface. The command consists of a serial write operation with the address and data values shown in Table 16. Once the customer access mode is entered, it is not possible to change access modes without power-cycling the device. After power up, there is no time limit to when the access code may be entered.

Table 16: Customer Access Code

Access Mode	Address	Data
Customer Access	0x35	0x2C413534

To enable customer access mode, we have to send a some data to a specific register...

MSB 2C
41
35
LSB 34

We can use the following function to set the device to customer access mode.

```
32 void enableWrite()
33 {
34   // Enable the Customer Access mode from the data sheet
35
36   // Data from the data sheet
37   // uint32_t data = 0x2C413534;
38
39   Wire.beginTransmission(devAddr);
40   Wire.write(0x35);       // Customer Access Register
41   Wire.write(0x2C);       // Customer Access Code
42   Wire.write(0x41);
43   Wire.write(0x35);
44   Wire.write(0x34);
45   Wire.endTransmission();
46 }
47
```

Programming the ALS31313 I2C Address

06/19/2023
Wesley Coder

To change the address, we need to set bits 16:10 of Register 0x02.

If we read the register befor changing it contains the following:

0x 00 00 03 C0

To change it to 0x41 we need to set it to

0x 00 01 07 C0

0000 0000 0000 0001 0000 0111 1100 0000

16th bit, 10th bit

To do this, we can use the following method:

```
void changeAddr()
{
  uint32_t data = 0b00000000000000010000011111000000; // 31-24, 23-16, 15-8, 7-0

  Serial.println("Attempting to send...");
  Serial.println(data, HEX);

  Wire.beginTransmission(devAddr); // Tell the chip we are writing to it
  Wire.write(0x02);                // Tell the chip the cusster access address

  Wire.write(data>>24 & 0xFF);
  Wire.write(data>>16 & 0xFF);
  Wire.write(data>>8 & 0xFF);
  Wire.write(data & 0xFF);

  Wire.endTransmission();
}
```

0x 00 01 07 C0

Programming the AL531313 I2C Address

```ino
#include <Wire.h>

const byte devAddr = 0x60; // From data sheet when ADDR0 and ADDR1 are low
                           // Or use an I2C scanner


void setup()
{
  Wire.begin();         // join i2c bus
  Serial.begin(9600);  // start serial for output
}

void readAddr(byte regAddr)
{

  Wire.beginTransmission(devAddr);     // Communicate with this I2C addr
  Wire.write(regAddr);                 // Go to this register on the chip
  Wire.endTransmission();

  Wire.requestFrom(devAddr, 4);        // Request 4 Bytes from the device

  while (Wire.available()) {          // slave may send less than requested
    byte c = Wire.read();             // receive a byte
    Serial.print(c, HEX);             // print the byte
    Serial.print(", ");
    Serial.print(c, BIN);
    Serial.print(", ");
    Serial.print(c);
    Serial.println();
  }
}
```

```ino
void loop()
{
  // See what 0x02 holds
  readAddr(0x02);
  Serial.println();

  // Turn on customer access mode
  enableWrite();
  Serial.println();

  // Change the 0x02 register
  changeAddr();

  // See what 0x02 holds now...
  readAddr(0x02);
  Serial.println();

  // Sometimes the first read has all 0's
  // So we read it again
  delay(1000);
  readAddr(0x02);
  Serial.println();

  // End of program
  while(1);
}
```

Programming the ALS31313 I2C Address

06/19/2023
Wesley Cook

After running the code on the previous page, you should be able to pull ADDR0 and ADDR1 high. Then you should be able to communicate with it using I2C at 0x41.