LTC2309 – A Brief Guide

Version 1.3

By Murtadha AlSaeedi

روابي القابضة
Rawabi Holding

Revision History

Date	Rev	Details
February 17, 2014	1.2	Release
March 4, 2014	1.3	Pin Functions and Interfacing: Updated with suggestions to decrease noise. Suggested setup is updated with a capacitor at pin 19. Application Information: Added device addresses in hex. Corrected info regarding the wait period if nACKed request.

Table of Contents

1. Description

The LTC2309 is a low power, low noise and 8 channel analog to digital converter, with an I2C compatible interface. The device operates from a single 5 V voltage source, and supports a sleep mode to reduce power dissipation during inactivity. It also includes a configurable 8 channel analog input multiplexer, a configurable address and a 2.5 V internal reference. Up to 9 LTC2309 devices can co-exist on one I2C bus. External capacitors are required for the device to operate properly.

2. Specifications

- **Resolution:** 12 bits, max voltage = 4.096 V
- **Analog Input Voltage:** (GND – 0.3 V) to (VDD + 0.3 V)
- **Digital Input Voltage:** (GND – 0.3 V) to (VDD + 0.3 V)
- **Digital Output Voltage:** (GND – 0.3 V) to (VDD + 0.3 V)
- **Noise:** SNR = 73.4 dB
- **Conversion Time:** Typically = 1.3 us, Maximum = 1.8 us
- **Input Ranges:** Unipolar or Bipolar input ranges
- **Channels:** Internal 8-channel multiplexer (8 different analog voltage sources can be connected and converted to DC)
- **Addresses:** 9 selectable addresses.
- **Operating Temperature:** -55 to 150 C
- **Digital output:** Two-Wire serial interface (I2C)
- **Supported I2C modes supported:** Standard-mode and Fast-mode.

3. Power Requirements

Symbol	**Parameter**	**MIN**	**TYP**	**MAX**	**UNITS**
VDD	Supply Voltage	4.75	5	5.25	V
IDD	Supply Current		2.3	3	mA
	Nap Mode		210	350	uA
	Sleep Mode		7	15	uA
PD	Power Dissipation		11.5	15	mW
	Nap Mode		1.05	1.75	mW
	Sleep Mode		35	75	uW

TABLE 1: POWER REQUIREMENTS

4. Pin Functions and Interfacing

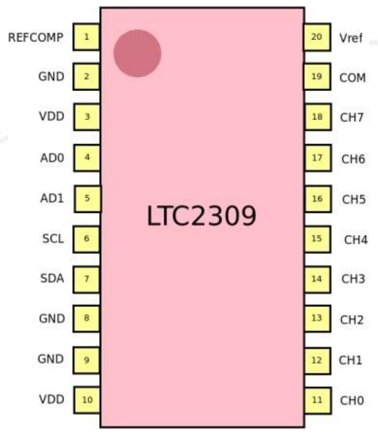

FIGURE 1: LTC2309 PINS

The LTC2309 has 20 pins. Pin 1 should be bypassed to ground with 10 uF and 0.1 uF ceramic capacitors in parallel. Pins 3 and 10 are used to supply voltage (5 V). According to the documentation of the device, V_{DD} should be bypassed to ground with a 10 uF ceramic capacitor in parallel with two 0.1 uF ceramic capacitors, each of the two should be as close as possible to one pin.

Pin 19 is the reference for all single-ended inputs, and must be free of noise (use a capacitor). It should be connected to ground for unipolar conversions, and midway between ground and REFCOMP for bipolar conversions. Pin 20 is the 2.5 V reference output, it should be bypassed to ground with a minimum 2.2 uF ceramic capacitor. This pin can be used to overdrive the internal reference by an external 2.5 V reference.

Pins 4 and 5 are used to select one of 9 possible addresses for the device by connecting them to ground, voltage source, or keeping them floating. Pins 7 and 8 are used to connect the device to the SCL line and the SDA line, respectively. Pins 11 to 18 are input ports for the analog voltages to be converted to digital, unused pins should be connected to ground to decrease noise. Pins 2, 8 and 9 should be connected to a solid ground.

<u>NOTE:</u> *The original document of the LTC2309 says that the pins 5 and 10, both should be connected to V_{DD}. However, our experiments on the package we have, showed that only pin 5 should be connected to V_{DD}, otherwise the ADC will not function properly.*

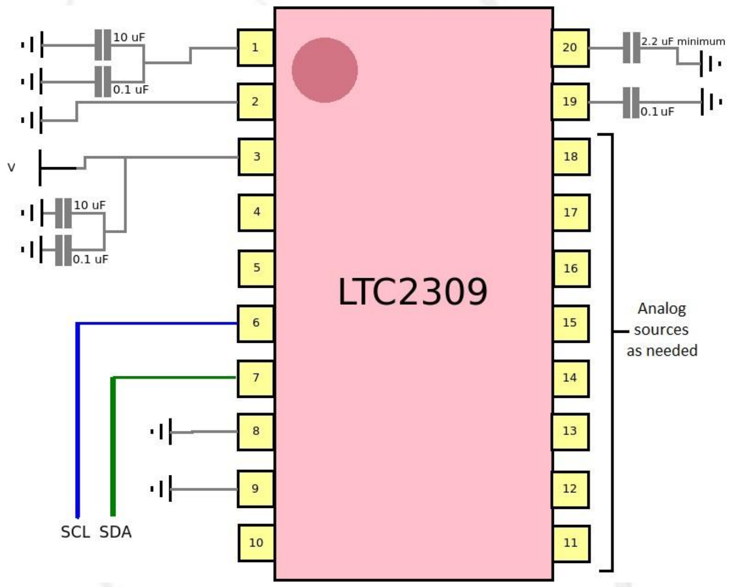

FIGURE 2: SUGGESTED SETUP

5. Application Information

The LTC2309 communicates through an I2C compatible interface. Conversions are initiated after successfully addressing the device and issuing a STOP condition. The device will not acknowledge (i.e. will send a nACK signal) any operation until the conversion is done. *Due to this, a wait period equal to the maximum conversion time (see the specifications) for the device, should be considered by the engineer after any not acknowledged request.* The LTC2309 has two registers, a 12-bit read only data register that stores converted values, this converted value is called an ADC code, and a 6-bit write only register to store the configuration. Any read operation to the LTC2309 will return the value stored in the data register, and any write operation will overwrite the configuration register. The device has eight channels to receive analog signals. A built-in multiplexer is used to choose one out (two in bipolar mode) of the eight channels, to convert its analog voltage to a digital value. The 12-bit converted value is sent through the I2C bus with four trailing zeroes.

5.1. Registers Description

5.1.1. Conversion Result Register

The conversion result register is a 12-bit read only register that stores the most recent conversion. The LTC2309 converts the analog voltage to what is called an ADC code. This code can be converted by the controller to actual voltage using an equation if needed. The minimum value of the ADC code is 0, and the maximum value is 4096.

5.1.2. Configuration Register

The configuration register is a 6-bit write only register that stores the operational configuration of the device. The value of the register on power up is zero, configuration may be needed on every power up. The purpose of each bit is described in table 3.

Bit Number	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	S/D	O/S	S1	S0	UNI	SLP

TABLE 2: CONFIGURATION REGISTER BITS

Bit(s)	**0**	**1**
S/D	DIFFEREBTIAL	SINGLE-ENDED
O/S	SIGN	ODD
S1, S2	Channel select	
UNI	BIPOLAR	UNIPOLAR
SLP	NAP MODE	SLEEP MODE

TABLE 3: CONFIGURATION BITS DESCRIPTION

5.2. Mechanism

5.2.1. Address Assignment

The pins AD1 and AD2 are used to configure the address of the device. Each pin can be connected to voltage source, ground or kept floating. Table X shows the address of the device for every state of the address pins. Also, the LTC2309 has a **global address (1101011)** which can be used to synchronize multiple LTC2309s.

AD1	AD2	Address
0	0	0001000 (0x8)
0	FLOAT	0001001 (0x9)
0	1	0001010 (0xA)
FLOAT	1	0001011 (0xB)
FLOAT	FLOAT	0011000 (0x18)
FLOAT	0	0011001 (0x19)
1	0	0011010 (0x1A)
1	FLOAT	0011011 (0x1B)
1	1	0101000 (0x28)

TABLE 4: ADDRESS ASSIGNMENT

5.2.2. Configuration

The operational configuration of the device can be changed by overwriting the configuration register. The configuration register is 6-bit wide. To overwrite the configuration register, you should follow the I2C protocol. You first issue a START condition, then issue a write operation to the device. After that, if the write operation is acknowledged, you send a byte of data, where the six most significant bits are the configuration bits, and the two least significant bits are just zeroes. Finally, after the device acknowledges receiving the data, a stop condition should be issued to end the session.

5.2.3. Channel Configuration

Table 5 explains the channel configuration briefly.

S/D	O/S	S1	S0	CH0	CH1	CH2	CH3	CH4	CH5	CH6	CH7	COM
0	0	0	0	+	-							
0	0	0	1			+	-					
0	0	1	0					+	-			
0	0	1	1							+	-	
0	1	0	0	-	+							
0	1	0	1			-	+					
0	1	1	0					-	+			
0	1	1	1							-	+	
1	0	0	0	+								-
1	0	0	1			+						-
1	0	1	0					+				-
1	0	1	1							+		-
1	1	0	0		+							-
1	1	0	1				+					-
1	1	1	0						+			-
1	1	1	1								+	-

TABLE 5: CHANNEL CONFIGURATION

5.2.4. Reading Conversion Result

The converted voltage is stored in the conversion result register (data register), reading it is straight forward. You first issue a START condition, then issue a read operation to the device. After that, if the read operation is acknowledged, you read two bytes of data, where the 12 most significant bits are the conversion bits, and the 4 least significant bits are just zeroes. Finally, after the device acknowledges receiving the data, a stop condition should be issued to end the session.

6. LTC2309 Propeller Example

```
/** Company: Rawabi United Safety Services
     Engineer: Murtadha Al-Saeedi


     Description: This code is written for the LTC2309. The code first overwrite the      the
configuration register (Single ended, sign, ch0, unipolar, nap mode).       After that it just
performs continuous voltage reading, an equation      and constants from the official website of the
ADC are used to convert the adc_code to real voltage.
**/

#include "simpletools.h" // Include simple tools
#include "i2c.h" // Include i2c library

unsigned int stack[(160 + (50 * 4)) / 4];
static float LTC2309_lsb = 1.0002442E-03;
//!< Ideal LSB voltage for a perfect part
static int16_t LTC2309_offset_code = 0;
//!< Ideal offset for a perfect part
uint16_t adc_code; float sum = 0;
uint16_t firstByte = 0;
uint16_t secondByte = 0;
float numOfSamples = 100.0;


int main()
{
  i2c* bus;
  pause(1000); //to make sure we can see the readings printed on the terminal screen
  bus = i2c_open (bus, 2, 1, 0); //establishing i2c lines on port 2 (SCL) and port1 (SDA)
  i2c_start(bus); //sending the start bit to the bus
  if(i2c_writeByte(bus, 0x30)==0) //sending the address of the LTC2309(0001_111) and the write bit
  (0)    printf("ack\n");
  if(i2c_writeByte(bus, 0x88)==0) //sending the configuration (0101_1000)
  printf("ack2\n");
  i2c_stop(bus); //sending the stop bit to the bus
  while(1)
  {
     for(int i = 0; i < numOfSamples; i++) //collecting "numOfSamples" samples, then averaging
     {
        i2c_start(bus); //seding the start bit to the bus
        i2c_writeByte(bus, 0x31); //sending the address of the LTC2309 (0001_111) and the write bit
        (1) firstByte = i2c_readByte(bus, 0); //getting the first byte and acknowledging receiving it
        secondByte = i2c_readByte(bus, 0); //getting the second byte, and acknowledging receiving it
        firstByte <<= 4; //shifting to the left by 4 to make space for the remaining 4 bits of the
        2nd byte secondByte >>= 4; //shifting to the right by 4, to correct the alignment of bits
        i2c_stop(bus); //sending the stop bit to the bus
        adc_code = ( firstByte | secondByte); //ORing the two bytes to get the adc_code
        sum = sum + (float)(adc_code+LTC2309_offset_code)*LTC2309_lsb; //just for accumulation and
        converting
        //the adc_code to real voltage
     }
     printf("Voltage: %f V\n", sum/numOfSamples); //printing the averaged voltage
     printf("\1"); //clearing the screen
     sum = 0; //emptying the accumulator
     pause(300); //just slowing down
  }
}
```