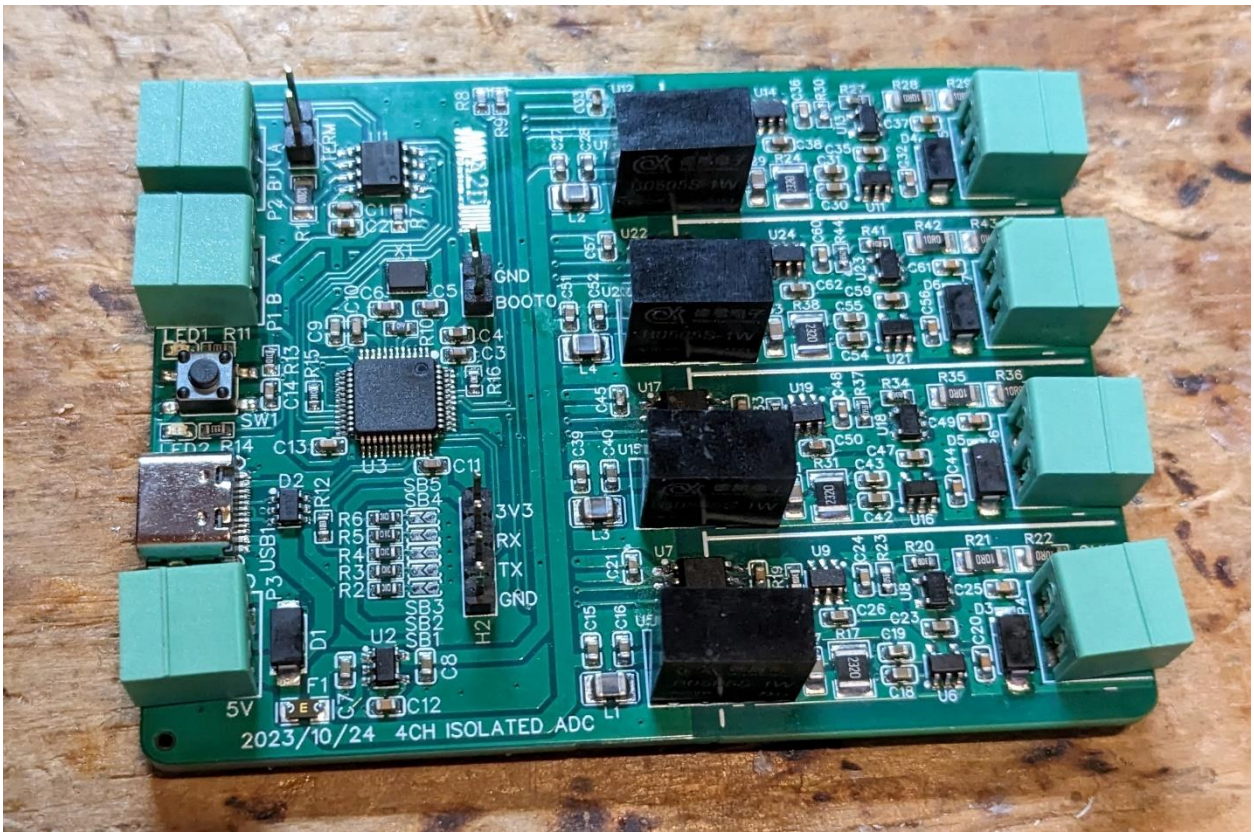
A2D Electronics

4CH Isolated ADC Spec Sheet



Description

4 galvanically isolated voltage inputs allows measurement of any voltage you want. 0-5V input range and 16-bits at 15 Samples Per Second is ideal for low voltage high precision acquisition. Includes a Programmable Gain Amplifier (PGA) for measuring even lower voltage signals.

Ideal for measuring a series stack of lithium-ion cells in a battery, or for removing the possibility of ground loops when measuring different sources. Up to 32 devices can be daisy-chained with RS485 communication layer for up to 128 analog inputs.

Specifications

Channels	4
Voltage Input Range	**0-5V**, 0-2.5V, 0-1.25V, 0-0.625V (depending on gain, default 0-5V)
Voltage Input Connector	3.81mm pitch 2-pin pluggable terminal block
ADC Bits	**16**, 14, 12 (configurable with sample rate, default 16)
ADC Sample Rate	**15**, 60, 240 Samples Per Second (configurable, default 15)
ADC Gain	**1**, 2, 4, 8 (configurable, default 1)
ADC Least-Significant-Bit Size	**176uV**, 88uV, 44uV, 22uV (depending on gain, default 176uV)
Power - Input Voltage	5V
Power - Input Source	3.81mm pitch 2-pin pluggable terminal block OR USB-C (not both at the same time)
Power - Input Current	250mA
Communication Interface	SCPI (through USB), RS485
RS485 Connector	3.81mm pitch 2-pin pluggable terminal block Up to 32 boards connected to a single USB interface

Included Parts

- 4CH Isolated ADC PCB
- 7x 3.81mm pitch 2-pin pluggable terminal block for RS485, channel inputs, and optional power input
- USB-A to USB-C cable

Communication

SCPI interface is available on the USB port. This device can interface with Python through PyVisa. I have created a python library for this device as well as a GUI for calibrating it automatically using another programmable multimeter and power supply.

RS485 communication interface is available (though not currently supported in the firmware). It can be used to chain up to 32 boards together with a single USB interface, allowing a total of 128 isolated voltage measurements. The first board gets powered by USB, and subsequent boards get powered by the external 5V supply. The GND of all the boards (including the one powered by USB) must be connected as the RS485 interface is not isolated.

Firmware is available on Github. You are free to modify firmware and upload your own. The simplest way is to upload over the USB interface using the Arduino IDE, but for more advanced users there are serial header pins and a BOOT0 jumper available.

SCPI Commands

The device will show up as a COM port on the computer. You can use PyVisa to connect to it using the PySerial backend. Improper commands or parameters will not have any negative effects, they will simply be ignored by the board and will do nothing unless type conversion from String to float or String to Integer fails.

<ch> parameter

For all the commands below, the <ch> parameter is an integer in the range of 1 to 4 to select which channel. 0 is also supported on some commands and is used to select all the channels. For queries with ch 0 parameter, the query will return the results of all 4 channels separated by a comma.

PyVisa Settings

The following connection settings should be used for the PyVisa connection:

read_termination	'/r/n'
write_termination	'/n'
baud_rate	115200
query_delay	0.001

Once connected, you can send commands and read responses from the instrument using PyVisa's *query*, *write*, and *query_ascii* commands.

*IDN?

Returns the instrument identification: Manufacturer, Model, Serial Number, Firmware Version

Syntax	*IDN?
Response Example	A2D Electronics,4CH Isolated ADC,1234,V1.0.0
PyVisa Example	instrument_idn = isnt.query('*IDN?')

*RST

Resets the instrument. Turns off the LED, sets the active calibration values to the values stored in flash, and sets the ADCs to a GAIN of 1, Data Rate of 15 samples per second for 16-bit readings, and continuous conversion mode.

Syntax	*RST
PyVisa Example	isnt.write('*RST')

MEAS:VOLT <ch>?

Returns the voltage at the input for the specified channels. The active calibration values are used to convert the voltage measured at the ADC to the voltage at the input.

Syntax	MEAS:VOLT <ch>?
<ch>	Integer from 0-4
Response Example	1.2345
PyVisa Example	ch1_voltage = float(isnt.query('MEAS:VOLT 1?'))

MEAS:VOLT:ADC <ch>?

Returns the voltage at the ADC for the specified channels.

Syntax	MEAS:VOLT:ADC <ch>?
<ch>	Integer from 0-4
Response Example	1.2345
PyVisa Example	ch1_adc_voltage = float(isnt.query('MEAS:VOLT:ADC 1?'))

INSTR:LED <state>

Turns the LED on or off according to *state*.

Syntax	INSTR:LED <state>
<state>	Integer from 0-1. 0 turns the LED off. 1 turns the LED on.
PyVisa Example	inst.write('INSTR:LED 1')

CAL:RESET <ch>

Sets the active calibration values for the specified channels to the default values.

Syntax	CAL:RESET <ch>
<ch>	Integer from 0-4
PyVisa Example	inst.write('CAL:RESET 1')

CAL:SAVE <ch>

Saves the active calibration values for the specified channels to flash. Upon reset (*RST) and power-up, the calibration values stored in flash are set as the active calibration values.

Syntax	CAL:SAVE <ch>
<ch>	Integer from 0-4
PyVisa Example	inst.write('CAL:SAVE 1')

CAL:VOLT <m1,a1,m2,a2,ch>

Generates new calibration values for the selected channel and sets them as the active calibration values. Calibration is done with a linear calibration with offset and gain values.

Calibration points should be chosen relative to the range you want it to be accurate in. If you want to measure in a range of 2.5 to 4.2V for battery testing, then choose your calibration points to be near the two edges of that range.

Syntax	CAL:VOLT <m1,a1,m2,a2,ch>
<m1>	Float. The measured voltage at the ADC (MEAS:VOLT:ADC <ch>?) when applying the 1st calibration point to the input.
<a1>	Float. The actual voltage applied to the input when measurement <m1> was taken. This could be from a calibrated DMM or a known voltage reference.

<m2>	Float. The measured voltage at the ADC (MEAS:VOLT:ADC <ch>?) when applying the 2nd calibration point to the input.
<a2>	Float. The actual voltage applied to the input when measurement <m2> was taken. This could be from a calibrated DMM or a known voltage reference.
<ch>	Integer from 1-4
PyVisa Example	inst.write('CAL:VOLT 1.02,2.34,1.98,4.34,1)

CAL <ch>?

Returns the gain and offset calibration for the requested channels. Return is <offset>,<gain>.

Syntax	CAL <ch>?
Response Example	0.004,2.81562
PyVisa Example	cal_values = [float(val) for val in inst.query_ascii_values('CAL 1?')]

Firmware and Arduino IDE Settings

Firmware for this board is under the following GitHub repository:
https://github.com/mbA2D/A2D_4CH_Isolated_ADC

The STM32 board files will need to be installed in the Arduino IDE under the "Additional Boards Manager URLs" in the File->Preferences menu. Instructions are available here:
https://github.com/stm32duino/Arduino_Core_STM32/wiki/Getting-Started

The following settings should be used in the Arduino IDE to upload any modifications to the firmware:

Board	Generic STM32F1 series
Debug symbols and core logs	None
Optimize	Smallest (-Os default)
Board part number	BluePill F103C8
C Runtime Library	Newlib Nano (default)
Upload method	Maple DFU Bootloader 2.0
USB support (if available)	CDC (generic 'Serial' supersede U(S)ART)
U(S)ART support	Enabled (generic 'Serial')
USB speed (if available)	Low/Full Speed

Hardware and Schematic

The full schematic can be found here: https://a2delectronics.ca/wp-content/uploads/2023/11/A2D-Electronics-4CH-Isolated-ADC-Schematic.pdf

Programming

This board was designed with a similar schematic to a standard 'Blue Pill' board while fixing common issues (i.e. USB D+ resistor among others). This should make it easy for others to understand the schematic and to develop custom firmware for it should they want to. The H1 header connects BOOT0 to 3V3 (not GND as mistakenly labelled in the silkscreen).

ESD Protection

A small capacitor, a TVS diode, and current limiting resistors are applied to each channel input for ESD protection. There will be some leakage current into this TVS diode from the input, and some leakage into the buffer opamp non-inverting terminal.

Power Input and Voltage Measurement Range Limits

An isolated, unregulated, DC-DC converter is used to power each channel's input buffer. A 232Ohm resistor is placed on the output of each isolated DC-DC converter to ensure the minimum load current spec is met (this is a large part of the 250mA input current required for this device). The maximum voltage that the buffer can output is its supply voltage, which comes from the DC-DC converter, which comes from the input power source. If your input power source is not 5V, then the ADC will not be able to measure up to 5V. If powering from the USB-C port, the power source can be as low as 4.5V, so only up to 4.5V can be measured reliably on the channel inputs. Up to 5.5V can be safely applied to the additional power input header but should never be applied at the same time as the USB-C input voltage.

External power should be used with the RS485 communication interface, or the power wire of the USB cable should be cut.

ADCs and Calibration

The ADCs used are the MCP3425. 1 on each channel, with the 'A0' version for channel 1, 'A1' for channel 2, and so on. Each device has a separate fixed I2C address, and they are all on the same I2C bus (SCL1/SDA1). These ADCs feature self-calibration of offset and gain for each conversion. The additional calibration provided by this board is used to correct for errors in the voltage divider tolerance, opamp buffer input current and offset voltage, and ESD protection reverse leakage effects. There is an RC low-pass filter on the input of each ADC with a corner frequency of 2.3kHz. This is roughly 10x the highest sampling rate of 240Hz in 12-bit mode for the ADCs, so there will be no significant attenuation of the signal in the relevant sampling range.

RS485 Interface

The solder bridges on the board go to GPIO pins of the STM32 microcontroller and could be used to set an address for RS485 communication. 5 solder bridges can be set for up to 32 devices. TX3/RX3 of the micro are used for the RS485 transceiver. The RS485 bus operates differentially with a 3V3 power supply. A 120 Ohm termination resistor should be added to both ends of the bus. Putting a jumper over H3 connects a termination resistor on the board. The interface is not isolated, so all boards connected with RS485 must have the same GND reference on their power input.

V1.0 Hardware Errors

Please note the following mistakes on the V1.0 hardware (identified by no version label in the silkscreen):

1. The pin labeled GND in silkscreen on the BOOT0 header pin is actually 3V3