

Lab Entry – 2026-01-14

Metadata

- Date: 2026-01-14
- Project: Off Grid Solar Battery Charger
- Board / Rev:
- Scope: HIL test INA219. write to config register, read bus voltage reg and current reg.

Objective

Verify that the INA219 operates as expected. I found a few issues with my previous HIL test, so I made a more robust test.

Verify that we can calibrate the INA219 by writing to the calibration register. Verify that we can read the current register and it is what we expected. Verify that we can read the bus voltage and verify that it is what we expected.

Setup

Setup

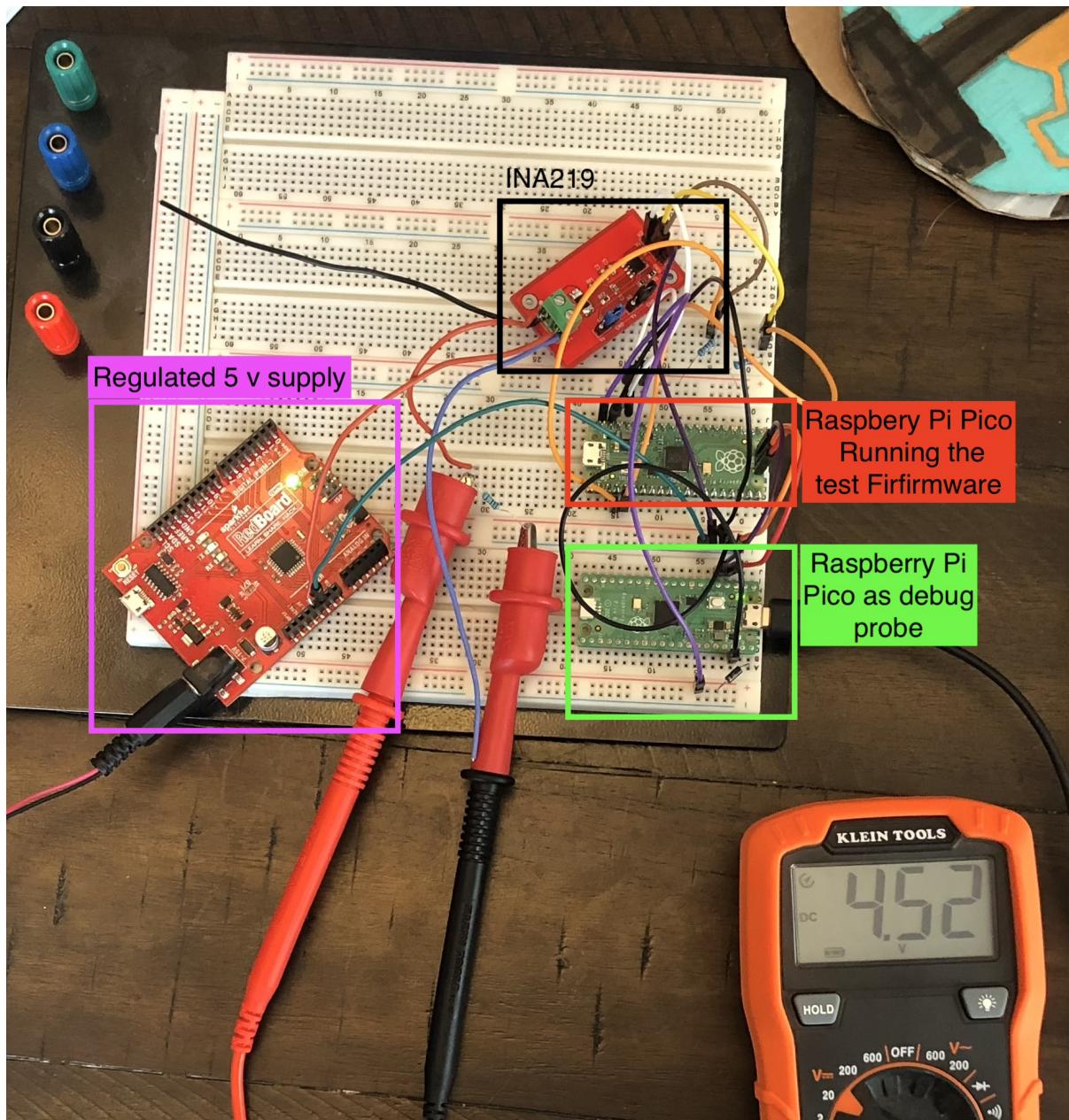


Figure 1: HIL Setup for Bus Voltage Verification

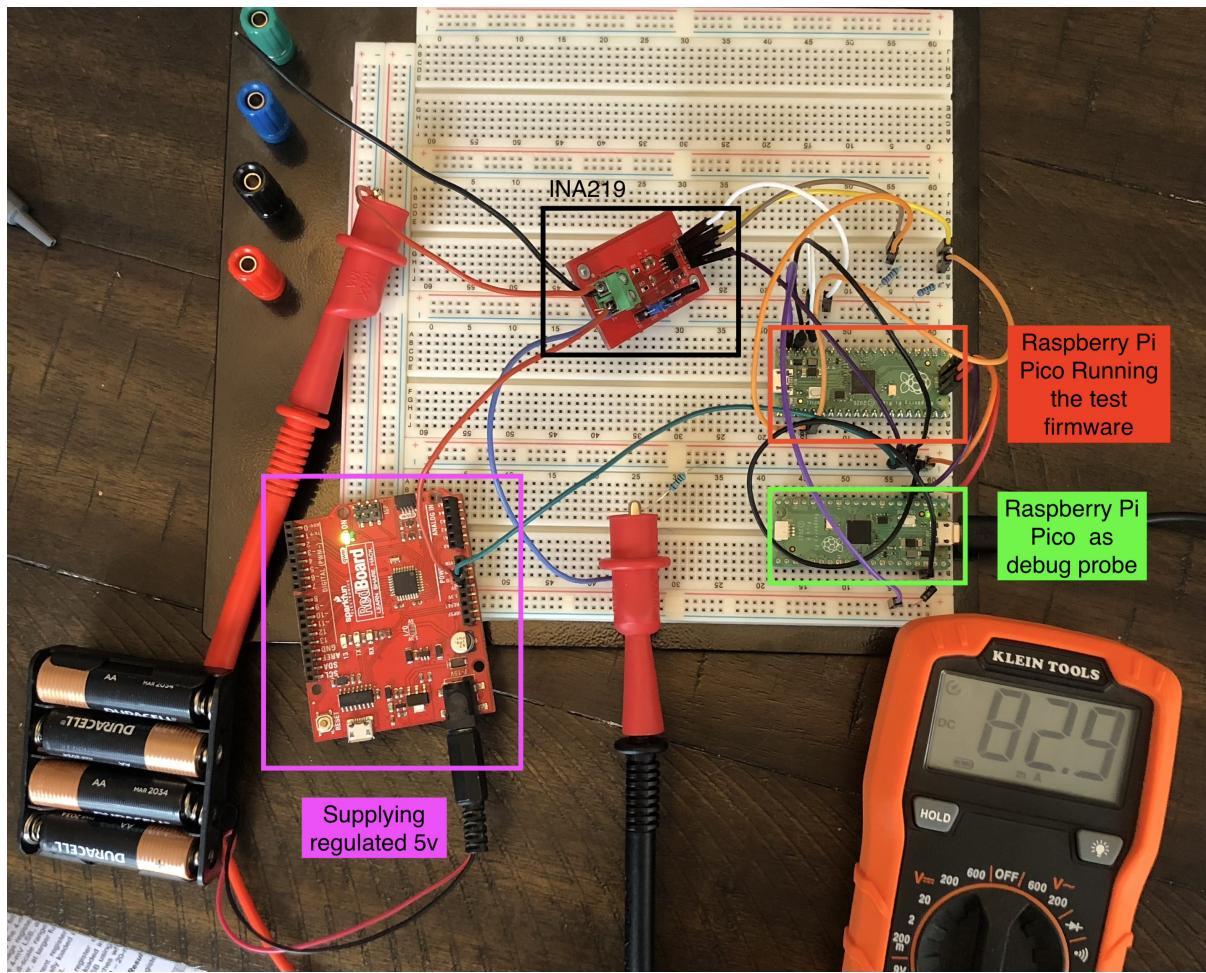


Figure 2: HIL Setup for Current measurement Verification

```

RUN AND DEBUG Pico Deb... ⚙ ... C I2C_Test.c {} launch.json C i2c.h

VARIABLES
  Local
    > reg = [3]
      r = <optimized out>
  Global
  Static: /Users/joshbenner/Projects/Code/Embedded
  Registers

WATCH
  > Datarx_Current = [2]
    Datarx_BusVoltage = [2]
      0 = 0x23
      1 = 0xfa

CALL STACK
BREAKPOINTS
CORTEX LIVE WATCH
XPERIPHERALS
  ADC @ 0x4004c000
  BUSCTRL @ 0x40030000
  CLOCKS @ 0x40008000
  DMA @ 0x50000000
  I2C0 @ 0x40044000
  I2C1 @ 0x40048000
  IO_BANK0 @ 0x40014000
  IO_QSPI @ 0x40018000
  PADS_BANK0 @ 0x40016000
PERIPHERALS

REGISTERS
  There is no data provider registered that can provide view data.

I2C_Test.c
1  #include <stdio.h>
2  #include "pico/stlib.h"
3  #include "hardware/i2c.h"
4
5
6  #define I2C_PORT i2c1
7  #define INA219_ADDR 0x40
8  #define I2C_SDA 2
9  #define I2C_SCL 3
10
11
12  uint8_t Datarx_Current[2];
13  uint8_t Datarx_BusVoltage[2];
14
15
16
17  int main() {
18      stdio_init_all();
19      i2c_init(I2C_PORT, 100000);
20      gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
21      gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
22      gpio_pull_up(4);
23      gpio_pull_up(5);
24
25      uint8_t reg[3] = {
26          0x05, //write to the calibration reg
27          0x0E, // calibration MSB
28          0x50 // calibration LSB
29      };
29
30      //update calibration register
31      int r = i2c_write_blocking(I2C_PORT, INA219_ADDR, &reg[0], 3, false);
32
33      // if successfully updated calibration reg get the measured current
34      if(r ==3){
35          uint8_t Current_Reg_Addr = 0x04;
36          r = i2c_write_blocking(I2C_PORT, INA219_ADDR, &Current_Reg_Addr, 1, true);
37
38          if(r==1){//successfully updated pointer to current reg read word
39              r = i2c_read_blocking(I2C_PORT, INA219_ADDR, &Datarx_Current[0], 2, false);
40          }
41
42
43          r=0; //reset r
44          reg [0] = 0x02; //update pointer to bus voltage register
45          r = i2c_write_blocking(I2C_PORT, INA219_ADDR, &reg[0], 1, true);
46
47          if (r == 1){// if pointer successfully updated read word
48              r = i2c_read_blocking(I2C_PORT, INA219_ADDR, &Datarx_BusVoltage[0], 2, false);
49          }else{
50              printf("INA219 did NOT ACK at 0x40\n");
51          }
52          while (1) tight_loop_contents();
53      }
54  }

```

Figure 3: Software for running the Voltage Bus HIL test

```

RUN AND DEBUG Pico Deb... ⚡ ... C I2C_Test.c {} launch.json C i2c.h
VARIABLES Local > reg = [3] r = 0x2
Global Static: /Users/joshbenner/Projects/Code/Embedded/Registers
WATCH Datarx = [2]
0 = 0x0
1 = 0x56
CALL STACK
BREAKPOINTS I2C_Test.c (42) I2C_Test.c (47)
CORTEX LIVE WATCH Hint: Use & Enable "liveWatch" in your l...
XPERIPHERALS ADC @ 0x4004c000
BUSCTRL @ 0x40030000
CLOCKS @ 0x40008000
DMA @ 0x50000000
I2C0 @ 0x40044000
I2C1 @ 0x40048000
IO_BANK0 @ 0x40014000
IO_CSPL @ 0x40018000
PERIPHERALS
REGISTERS There is no data provider registered that can provide view data.
I2C_Test.c main()
1 #include <stdio.h>
2 #include "pico/stl.h"
3 #include "hardware/i2c.h"
4
5
6 #define I2C_PORT i2c1
7 #define INA219_ADDR 0x40
8 #define I2C_SDA 2
9 #define I2C_SCL 3
10
11 uint8_t Datarx[2];
12
13
14
15
16 int main() {
17     stdio_init_all();
18     i2c_init(I2C_PORT, 100000);
19     gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
20     gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
21     gpio_pull_up(4);
22     gpio_pull_up(5);
23
24     uint8_t reg[3] = {
25         0x05, //write to the calibration reg
26         0x0E, // calibration MSB
27         0x50 // calibration LSB
28     };
29     //update calibration register
30     int r = i2c_write_blocking(I2C_PORT, INA219_ADDR, &reg[0], 3, false);
31
32     // if successfully updated calibration reg get the measure current
33     if(r == 3){
34         uint8_t Current_Reg_Addr = 0x04;
35         r = i2c_write_blocking(I2C_PORT, INA219_ADDR, &Current_Reg_Addr, 1, true);
36
37         if(r == 1){ //successfully updated pointer to current reg read word
38             r = i2c_read_blocking(I2C_PORT, INA219_ADDR, &Datarx[0], 2, false);
39         }
40     }
41
42     if (r == 1){
43         printf("INA219 ACKed at 0x40\n");
44     }else{
45         printf("INA219 did NOT ACK at 0x40\n");
46
47     while (1) tight_loop_contents();
48     return 1;
49 }

```

Figure 4: Software for running the Current measurement HIL test

Note: I set the initial calibration register to 0xFFE. I then measured the actual current and adjust the calibration register using the following equation from the INA219 data sheet.

Corrected_Full_Scale_Cal = trunc((Cal*MeasuredShuntCurrent)/INA219_Current) which ended up being = 0x0E50

Measurements & Observations

If we look at the multimeter in Figure 1, we can see that the measured bus voltage is 4.52 V. If we look at Figure 3, we see that the Datarx_BusVoltage contains 0x23FA.

According to the INA219 data sheet to convert Vbus into raw reading, we perform a shift left by 3 bits and multiply by the LSB (4mV). Performing these steps below:

$$(0x23FA \gg 3) * 4 = 0x11FC = 4,604 \text{ mV.}$$

Looking at Figure 2 we can see that the multimeter is reading 82.9 mA of current in the system. If we look at Figure 4, we can see that Datarx buffer contains 0x0056 which is 86 mA.

The multimeter and the INA219 are slightly off. This could be due to not recalibrating before each test.

Conclusions / Next Steps

The INA219 is functional and has passed the HIL test.

The next step is to repeat the process for the second board.