

Lab Entry – 2026-01-11

Metadata

- Date: 2026-01-11
- Project: Offgrid solar battery charger
- Board / Rev:
- Scope: HIL Test INA219 Breakout board

Objective

Verify that the INA219 Breakout Board works by seeing if it ACKS a request to write to it.

Setup

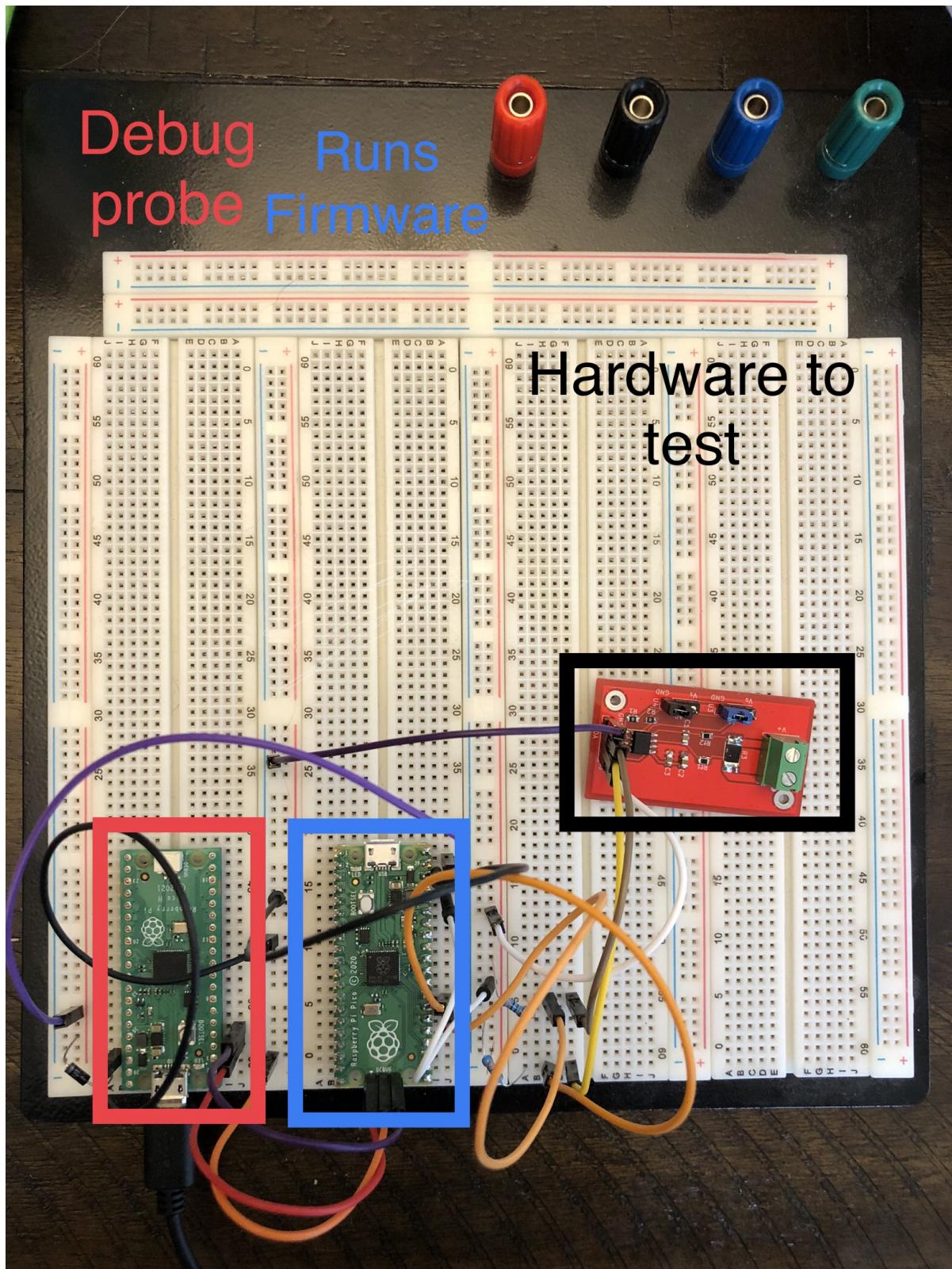


Figure 1: HIL Setup

We will use the i2c api provided by Raspberry pi to write to the hardware to be tested. The specific function used will be `i2c_write_blocking`, which returns the number of bytes written, or `PICO_ERROR_GENERIC` if address not acknowledged, no device present.

We set the addreses of the PCB to be `0x40` (`A0` and `A1` = `GND`).

If the INA219 PCB is working as expected, we should see `i2c_write_blocking` return a `0` (this is equivalent to ACKing a write request of `0` bytes).

```

1  /*
2   * Author: Josh Benner
3   * INA219 HIL test
4   *
5   * If ACK_Test returns 0 the HIL test is considered passed at this point.
6   * If ACK_Test return -1 then the request was NACKed and the test is failed.
7   *
8   * TODO: Create a more robust test, Come up with a simple circuit and calculate
9   *       expected values in each register. Verify results.
10  */
11 */
12 #include <stdio.h>
13 #include "pico/stl.h"
14 #include "hardware/i2c.h"
15
16 // I2C defines
17 // This example will use I2C0 on GPIO8 (SDA) and GPIO9 (SCL) running at 400KHz.
18 // Pins can be changed, see the GPIO function select table in the datasheet for information on GPIO assignments
19 #define I2C_PORT i2c0
20 #define I2C_SDA 4
21 #define I2C_SCL 5
22 #define CALIBRATION 0xFFFF
23 #define CURRENT_LSB
24
25 //PCB Deice
26 #define ADDRESS 0x40
27
28 typedef enum{
29     CONFIG_REG = 0x00,
30     SHUNT_VOL_REG,
31     BUS_VOL_REG,
32     POWER_REG,
33     CURRENT_REG,
34     CALIBRATION_REG
35 } INA219_REGS_E;
36
37
38
39
40
41
42 int main()
43 {
44     uint8_t datarx;
45     stdio_init_all();
46     int ACK_Test;;
47     // I2C Initialisation. Using it at 400Khz.
48     i2c_init(I2C_PORT, 400*1000);
49
50     gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
51     gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
52     gpio_pull_up(I2C_SDA);
53     gpio_pull_up(I2C_SCL);
54     // For more examples of I2C use see https://github.com/raspberrypi/pico-examples/tree/master/i2c
55     ACK_Test=i2c_write_blocking(I2C_PORT, ADDRESS, NULL, 0, false);
56
57     if(ACK_Test==0){
58         printf("Address: %u was ACKed\n",ADDRESS);
59     }else{
60         printf("Address: %u was NACKED",ADDRESS);
61     }
62 }

```

Figure 2: Firmware Running The Test

Observations

The screenshot shows the MicroPython Pico Editor interface. On the left is a sidebar with various icons for file operations, search, and project management. The main area displays the C code for `I2C_Test.c`. A yellow box highlights the line `ACK_Test = 0` in the `main()` function. The code itself is as follows:

```
C I2C_Test.c x
C I2C_Test.c > main()
5 // I2C defines
6 // Pins can be changed, see the GPIO FUNCTION SELECT TABLE in the datasheet for information on GPIO assignments
7 #define I2C_PORT i2c0
8 #define I2C_SDA 4
9 #define I2C_SCL 5
10
11 // PCB Device
12 #define ADDRESS 0x40
13
14
15
16
17 int main()
18 {
19     uint8_t datarx;
20     stdio_init_all();
21     int ACK_Test;;
22     // I2C Initialisation. Using it at 400Khz.
23     i2c_init(I2C_PORT, 400*1000);
24
25     gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
26     gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
27     gpio_pull_up(I2C_SDA);
28     gpio_pull_up(I2C_SCL);
29     // For more examples of I2C use see https://github.com/raspberrypi/pico-examples/tree/master/i2c
30     ACK_Test=i2c_write_blocking(I2C_PORT, ADDRESS, NULL, 0, false);
31
32     if(ACK_Test==0){
33         printf("Address: %u was ACKed\n",ADDRESS);
34     }else{
35         printf("Address: %u was NACKED",ADDRESS);
36     }
37 }
```

Figure 3: Test Result

Figure 3 is a snap shot of the debug process of running the firmware test. We have a breakpoint set right after `ACK_Test` has been assigned by `i2c_write_blocking` function. If we look at the yellow box at the top left corn, it shows that `ACK_Test` result is 0 and thus the INA219 PCB acknowledged the request to write 0 bytes to it. Therefore This pcb has passed this simple test.

Next Steps

Provide a more robust test. Use a 9 V battery and 100 ohm resistor. Calculate expected values in registers and then compare the expected result to the actual result. Consider error tolerances and if the results are within reason, the PCB will be considered functional.