AdafruitINA260 Library Documentation

Release 1.0

Bryan Siepert

Oct 25, 2021

Contents

CircuitPython driver for the TI INA260 current and power sensor

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython
- Bus Device
- Register

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle.

1.1 Installing from PyPI

Note: This library is not available on PyPI yet. Install documentation is included as a standard element. Stay tuned for PyPI availability! If the library is not planned for PyPI, remove the entire 'Installing from PyPI' section. On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally from PyPI. To install for current user:

```
pip3 install adafruit-circuitpython-ina260
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-ina260
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-ina260
```

CHAPTER 2

Usage Example

```
import time
import board
import adafruit_ina260

i2c = board.I2C()
ina260 = adafruit_ina260.INA260(i2c)
while True:
    print("Current: %.2f Voltage: %.2f Power:%.2f"
        %(ina260.current, ina260.voltage, ina260.power))
    time.sleep(1)
```

CHAPTER 3

Documentation

API documentation for this library can be found on Read the Docs.

CHAPTER 4

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 5

Documentation

For information on building library documentation, please check out this guide.

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/ina260_simpletest.py

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_ina260

i2c = board.I2C()
ina260 = adafruit_ina260.INA260(i2c)
while True:
    print(
        "Current: %.2f mA Voltage: %.2f V Power:%.2f mW"
        % (ina260.current, ina260.voltage, ina260.power)
    )
    time.sleep(1)
```

6.2 adafruit_ina260

CircuitPython driver for the TI INA260 current and power sensor * Author(s): Bryan Siepert Implementation Notes ———————— **Hardware:** * INA260 Breakout **Software and Dependencies:** * Adafruit CircuitPython firmware for the supported boards: * https://github.com/adafruit/circuitpython/releases * Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice * Adafruit's Register library: https://github.com/adafruit/Adafruit_CircuitPython_Register

class adafruit_ina260.**AveragingCount**
Options for `averaging_count`

AveragingCount	Number of measurements to average
AveragingCount.COUNT_1	1 (Default)
AveragingCount.COUNT_4	4
AveragingCount.COUNT_16	16
AveragingCount.COUNT_64	64
AveragingCount.COUNT_128	128
AveragingCount.COUNT_256	256
AveragingCount.COUNT_512	512
AveragingCount.COUNT_1024	1024

static get_averaging_count(*avg_count*)
Retrieve the number of measurements giving value read from register

class adafruit_ina260.**ConversionTime**
Options for `current_conversion_time` or `voltage_conversion_time`

ConversionTime	Time
ConversionTime.TIME_140_us	140 us
ConversionTime.TIME_204_us	204 us
ConversionTime.TIME_332_us	332 us
ConversionTime.TIME_558_us	588 us
ConversionTime.TIME_1_1_ms	1.1 ms (Default)
ConversionTime.TIME_2_116_ms	2.116 ms
ConversionTime.TIME_4_156_ms	4.156 ms
ConversionTime.TIME_8_244_ms	8.244 ms

static get_seconds(*time_enum*)
Retrieve the time in seconds giving value read from register

class adafruit_ina260.**INA260**(*i2c_bus*, *address=64*)
Driver for the INA260 power and current sensor.

Parameters

- **i2c_bus** (*I2C*) – The I2C bus the INA260 is connected to.
- **address** – The I2C device address for the sensor. Default is `0x40`.

alert_function_flag
While only one Alert Function can be monitored at the ALERT pin at time, the Conversion Ready can also be enabled to assert the ALERT pin. Reading the Alert Function Flag following an alert allows the user to determine if the Alert Function was the source of the Alert.

When the Alert Latch Enable bit is set to Latch mode, the Alert Function Flag bit clears only when the Mask/Enable Register is read. When the Alert Latch Enable bit is set to Transparent mode, the Alert Function Flag bit is cleared following the next conversion that does not result in an Alert condition.

alert_latch_enable
Configures the latching feature of the ALERT pin and Alert Flag Bits.

alert_limit
The Alert Limit Register contains the value used to compare to the register selected in the Mask/Enable Register to determine if a limit has been exceeded. The format for this register will match the format of the register that is selected for comparison.

alert_polarity_bit
Active-high open collector when True, Active-low open collector when false (default).

averaging_count
The window size of the rolling average used in continuous mode

bus_voltage_over_voltage
Setting this bit high configures the ALERT pin to be asserted if the bus voltage measurement following a conversion exceeds the value programmed in the Alert Limit Register.

bus_voltage_under_voltage
Setting this bit high configures the ALERT pin to be asserted if the bus voltage measurement following a conversion drops below the value programmed in the Alert Limit Register.

conversion_ready
Setting this bit high configures the ALERT pin to be asserted when the Conversion Ready Flag, Bit 3, is asserted indicating that the device is ready for the next conversion.

current
The current (between V+ and V-) in mA

current_conversion_time
The conversion time taken for the current measurement

mask_enable
The Mask/Enable Register selects the function that is enabled to control the ALERT pin as well as how that pin functions. If multiple functions are enabled, the highest significant bit position Alert Function (D15-D11) takes priority and responds to the Alert Limit Register.

math_overflow_flag
This bit is set to 1 if an arithmetic operation resulted in an overflow error.

mode
The mode that the INA260 is operating in. Must be one of `Mode.CONTINUOUS`, `Mode.TRIGGERED`, or `Mode.SHUTDOWN`

overcurrent_limit
Setting this bit high configures the ALERT pin to be asserted if the current measurement following a conversion exceeds the value programmed in the Alert Limit Register.

power
The power being delivered to the load in mW

power_over_limit
Setting this bit high configures the ALERT pin to be asserted if the Power calculation made following a bus voltage measurement exceeds the value programmed in the Alert Limit Register.

reset_bit
Setting this bit t 1 generates a system reset. Reset all registers to default values.

revision_id
Device revision identification bits

under_current_limit
Setting this bit high configures the ALERT pin to be asserted if the current measurement following a conversion drops below the value programmed in the Alert Limit Register.

voltage
The bus voltage in V

voltage_conversion_time
The conversion time taken for the bus voltage measurement

class adafruit_ina260.**Mode**

Modes avaible to be set

Mode	Description
`Mode.CONTINUOUS`	Default: The sensor will continuously measure the bus voltage and shunt voltage across the shunt resistor to calculate `power` and `current`
`Mode.TRIGGERED`	The sensor will immediately begin measuring and calculating current, bus voltage, and power. Re-set this mode to initiate another measurement
`Mode.SHUTDOWN`	Shutdown the sensor, reducing the quiescent current and turning off current into the device inputs. Set another mode to re-enable

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index