energymon

Release 0.1.2

Connor Imes

Feb 07, 2024

CONTENTS

This project provides Python bindings to energymon libraries.

CHAPTER

ONE

INSTALLATION

The package is available on PyPI:

```
pip install energymon
```

and on Conda Forge:

```
conda install energymon
```

CHAPTER

TWO

GETTING STARTED

The energymon libraries should be installed to the system and on the library search path (e.g., `LD_LIBRARY_PATH` on Linux/POSIX systems or `DYLD_LIBRARY_PATH` on macOS systems).

The simplest and most Pythonic approach is to use the `EnergyMon` context manager. For example, to measure the energy consumption of a task using the `energymon-default` library:

```
from energymon.context import EnergyMon

with EnergyMon() as em:
    print('Energy source:', em.get_source())
    uj_start = em.get_uj()
    # do some non-trivial work (must take longer than the energy source's update interval)
    uj_end = em.get_uj()
print('Energy (uJ):', uj_end - uj_start)
```

Specify the `lib` and `func_get` parameters when instantiating `EnergyMon` to use other energymon libraries.

Direct C struct bindings are also available using the `energymon` ctypes class. See the README in the project source for a detailed example.

CHAPTER

THREE

API REFERENCE

3.1 energymon

3.1.1 energymon package

Bindings for the native interface defined in `energymon.h`

class energymon.**energymon**

Bases: `Structure`

Binding to `energymon` struct.

fexclusive

Structure/Union member

ffinish

Structure/Union member

finit

Structure/Union member

finterval

Structure/Union member

fprecision

Structure/Union member

fread

Structure/Union member

fsource

Structure/Union member

state

Structure/Union member

Submodules

energymon.context module

Context management for using an `energymon`.

`class energymon.context.EnergyMon(lib: str | CDLL = 'energymon-default', func_get: str = 'energymon_get_default')`

Bases: `object`

A wrapper class and context manager for an `energymon`.

As a context manager, it is both reentrant and reusable.

Methods raise exceptions if the underlying native functions return an error.

`__init__(lib: str | CDLL = 'energymon-default', func_get: str = 'energymon_get_default')`

Create a new instance.

Parameters

- **lib** (*Union[str, ctypes.CDLL]*) – The library name (a `str`) or the library handler (a `ctypes.CDLL`).
- **func_get** (*str, optional*) – The native "getter" function name to use.

`finish() → None`

Finish the underlying `energymon`. If not already initialized, this is a no-op.

Only call this method if not using the pattern: `with EnergyMon(...) as context:`.

`get_interval_us() → int`

Get the refresh interval in microseconds.

Returns
The refresh interval in microseconds. This value should be greater than 0. If there is no minimum interval, returns 1.

Return type
int

`get_precision_uj() → int`

Get the best possible possible read precision in microjoules.

Returns
The best possible possible read precision in microjoules. If `0 < precision <= 1`, returns 1. If the precision is unknown, returns 0.

Return type
int

`get_source() → str`

Get a human-readable description of the energy monitoring source.

Initialization is not required to use this method.

Returns
A human-readable description of the energy monitoring source.

Return type
str

get_uj() → int

Get the total energy in microjoules.

Returns
The total energy in microjoules.

Return type
int

init() → None

Initialize the underlying energymon.

Only call this method if not using the pattern: `with EnergyMon(...) as context:`.

property initialized: bool

True if the energymon is initialized, False otherwise.

Type
bool

is_exclusive() → bool

Get whether the implementation requires exclusive access.

Initialization is not required to use this method.

Returns
True if the implementation requires exclusive access, False otherwise.

Return type
bool

energymon.util module

Utilities for using an energymon.

See documentation in the native energymon.h header file for additional details.

energymon.util.**finish**(*em:* energymon) → None

Finish the energymon (clean up private state).

Parameters
em (energymon) – The energymon must be initialized.

Raises
OSError – If the underlying function returns an error. This may occur under normal conditions, but is unlikely.

energymon.util.**get_energymon**(*lib*, *func_get: str = 'energymon_get_default'*) → *energymon*

Create an energymon and "get" it (populate its function pointers), but do not initialize it.

Parameters

- **lib** (*ctypes library*) – An energymon library loaded by ctypes (e.g., a CDLL).
- **func_get** (*str*) – The library function name used to populate the energymon struct.

Returns
An uninitialized energymon instance.

Return type
energymon

Raises

- **AttributeError** – If the getter function is not found.
- **OSError** – If the underlying function returns an error. This is allowed by the API but should not occur under normal conditions.

Notes

Assumes a standard "get" function prototype: `int (get) (energymon*)`. All known instances use this prototype, but the energymon API doesn't actually define this.

energymon.util.**get_interval_us**(*em:* energymon) → int

Get the refresh interval in microseconds.

Parameters
em (energymon) – The energymon must be initialized.

Returns
The refresh interval in microseconds. This value should be greater than 0. If there is no minimum interval, returns 1.

Return type
int

Raises
OSError – If the underlying function returns an error. This is allowed by the API but should not occur under normal conditions.

energymon.util.**get_precision_uj**(*em:* energymon) → int

Get the best possible possible read precision in microjoules.

For implementations that read from power sensors, this is a function of the precision of the power readings and the refresh interval.

Parameters
em (energymon) – The energymon must be initialized.

Returns
The best possible possible read precision in microjoules. If `0 < precision <= 1`, returns 1. If the precision is unknown, returns 0.

Return type
int

Raises
OSError – If the underlying function returns an error. This is allowed by the API but should not occur under normal conditions.

energymon.util.**get_source**(*em:* energymon, *maxlen=256*, *encoding='UTF-8'*, *errors='strict'*) → str

Get a human-readable description of the energy monitoring source.

Parameters
em (energymon) – The energymon doesn't need to be initialized.

Returns
A human-readable description of the energy monitoring source.

Return type
str

Raises
OSError – If the underlying function returns an error. This is allowed by the API but should not occur under normal conditions.

energymon.util.**get_uj**(*em:* energymon) → int

Get the total energy in microjoules.

Parameters
em (energymon) – The energymon must be initialized.

Returns
The total energy in microjoules.

Return type
int

Raises
OSError – If the underlying function returns an error. This may occur under normal conditions for some underlying sensors.

energymon.util.**init**(*em:* energymon) → None

Initialize the energymon (initialize private state).

Parameters
em (energymon) – The energymon must not be initialized.

Raises
OSError – If the underlying function returns an error. This may occur under normal conditions, e.g., if the sensors are not present or cannot be initialized.

energymon.util.**is_exclusive**(*em:* energymon) → bool

Get whether the implementation requires exclusive access.

In such cases it may be beneficial to run in a separate process and expose energy data over shared memory (or other means) so multiple applications can use the data source simultaneously.

Parameters
em (energymon) – The energymon doesn't need to be initialized.

Returns
True if the implementation requires exclusive access, False otherwise.

Return type
bool

energymon.util.**load_energymon_library**(*name: str = 'energymon-default'*)

Load an energymon library by name (no leading 'lib' or trailing extensions or version number).

Recommend setting LD_LIBRARY_PATH on Linux/POSIX, DYLD_LIBRARY_PATH on OSX, and PATH on Windows.

Parameters
name (*str, optional*) – The library name to search for.

Returns
The loaded shared library.

Return type
A ctypes library, e.g., a CDLL

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

INDEX

Symbols

E

F

G

I

L

M

S