
Pumpkin SupMCU Interface

Release 1.0.0

James Womack <james@pumpkininc.com>

Jun 08, 2021

CONTENTS:

1 *i2c* module API docs 1

1.1 *i2c.master* module API docs 1

2 *supmcu* module API docs 3

2.1 Data Types 3

2.2 Telemetry Parsing 6

2.3 SupMCU Discovery 6

2.4 SupMCU interface 8

3 Examples 9

3.1 Retrieving Telemetry Data 9

4 Indices and tables 11

Python Module Index 13

Index 15

I2C MODULE API DOCS

The `pumpkin_supmcu.i2c` module contains definitions for creating I2C Master devices, with a universal I2C Protocol. The `pumpkin_supmcu.i2c` module contains an implementations of the I2C Master for the following devices:

- The [I2C Driver Board](#) as `pumpkin_supmcu.i2cdriver.I2CDriverMaster`.
- Linux's `/dev/i2c-#` interface as `pumpkin_supmcu.linux.I2CLinuxMaster`
- (Coming soon) [Total Phase Aardvark Adaptor](#) as `I2CAardvarkMaster`.

Users can create (and possibly contribute) additional I2C Master implementations of other I2C Bus Adaptors.

1.1 `i2c.master` module API docs

The I2C Master [Protocol](#) is for creating I2C Bus Master implementations such as the `I2CDriverMaster`. Any implementation of the I2C Master can be used with functions/classes defined in the [supmcu](#) module.

class `pumpkin_supmcu.i2c.I2CBusSpeed(value)`

The possible I2C Bus speeds that are supported by the I2C Master Protocol.

Fast = 400

400 kHz bus speed

Standard = 100

100 kHz bus speed

class `pumpkin_supmcu.i2c.I2CMaster`

The [Protocol](#) to use for implementing a I2C Master device such as the `I2CDriver` or the `Aardvark`. Note the package

abstract property device_name: str

Returns the name of the I2C Master device.

Return type `str`

abstract property device_pullups: bool

Returns if the I2C pullups are ON or OFF.

Return type `bool`

abstract property device_speed: pumpkin_supmcu.i2c.master.I2CBusSpeed

Returns the I2C bus speed currently used.

Return type `I2CBusSpeed`

abstract get_bus_devices()

Gets the available I2C devices on the bus

Return type `Iterable[int]`

abstract read(*addr*, *amount*)

Reads *amount* bytes from the I2C address *addr*.

Return type `bytes`

abstract write(*addr*, *b*)

Writes the bytes *b* to the I2C address *addr*.

SUPMCU MODULE API DOCS

The `pumpkin_supmcu.supmcu` module contains the parsing, communication interface and module telemetry enumeration definitions.

2.1 Data Types

These data types are used throughout the `pumpkin_supmcu.supmcu` to type and structure the module definitions.

class `pumpkin_supmcu.supmcu.DataType(value)`

Different possible data types that can be returned from SupMCU Telemetry

Char = 2

A single *char* character

Double = 12

A *double* item.

Float = 11

A *float* item.

Hex16 = 14

A *uint16_t* item, displayed as a hexadecimal value.

Hex8 = 13

A *uint8_t* item, displayed as a hexadecimal value.

INT16 = 6

A *int16_t* item.

INT32 = 8

A *int32_t* item.

INT64 = 10

A *int64_t* item.

INT8 = 4

A *int8_t* item.

Str = 1

A null-terminated string

UINT16 = 5

A *uint16_t* item.

UINT32 = 7

A *uint32_t* item.

UINT64 = 9

A *uint64_t* item.

UINT8 = 3

A *uint8_t* item.

class `pumpkin_supmcu.supmcu.TelemetryType(value)`

Represents a module request for the SUP:TEL? items or MOD:TEL? items.

Module = 1

Module telemetry items (e.g. BM:TEL? #)

SupMCU = 0

SupMCU telemetry item (e.g. SUP:TEL? #)

class `pumpkin_supmcu.supmcu.TelemetryDataItem(data_type, value, string_value)`

A single data item from a telemetry request.

data_type

The `pumpkin_supmcu.supmcu.DataType` stored in the *TelemetryDataItem*.

value

The value parsed from the telemetry response.

string_value

The value formatted as a string.

class `pumpkin_supmcu.supmcu.SupMCUHDR(ready, timestamp)`

The SupMCU Telemetry header with timestamp and is_ready information.

ready

Value of the *ready* flag in the SupMCU telemetry.

timestamp

The value of the timestamp in the SupMCU telemetry response.

class `pumpkin_supmcu.supmcu.SupMCUTelemetry(header, items)`

A SupMCU Telemetry request response. Consists of zero or more *TelemetryDataItems*.

header

The `pumpkin_supmcu.supmcu.SupMCUHDR` object for the telemetry response.

items

The list of `pumpkin_supmcu.supmcu.TelemetryDataItems` that make up the telemetry response.

class `pumpkin_supmcu.supmcu.SupMCUTelemetryDefinition(name, telemetry_length, idx, format, simulatable=False, defaults=None)`

A SupMCU Telemetry definition consists of the name, length and format of the returned data.

format

The format string returned from *SUP/MOD:TEL? #,FORMAT* request.

idx

The index number of the telemetry item

name

The name of the telemetry item as returned from *SUP/MOD:TEL? #,NAME* request.

telemetry_length

The length in bytes of the telemetry response from the SupMCU module.

class pumpkin_supmcu.supmcu.**SupMCUModuleDefinition**(*name, cmd_name, address, supmcu_telemetry, module_telemetry, commands=<factory>*)

A SupMCU Module definition consists of the name, address, cmd_name, SupMCU Telemetry, module telemetry, and SupMCU commands and module commands.

address

The I2C address of the module.

cmd_name

The name of the module as used in SCPI commands (e.g. Battery Module 2 is BM2).

module_telemetry

The module telemetry definitions for all the module specific telemetry. This is a dictionary keyed by telemetry index to telemetry definition.

name

The name of the SupMCU module.

supmcu_telemetry

The SupMCU telemetry definitions that are common across all SupMCU modules. This is a dictionary keyed by telemetry index to telemetry definition.

pumpkin_supmcu.supmcu.**sizeof_supmcu_type**(*t*)

Returns the size in bytes of a SupMCU Data type. Note the Str type returns 0 as it's size is unknown until parsed.

Parameters *t* (*DataType*) – The DataType *t*.

Return type *int*

Returns The size of the type in bytes, unless Str, then its zero.

pumpkin_supmcu.supmcu.**typeof_supmcu_fmt_char**(*fmt_char*)

Returns the underlying SupMCU Data type for a given *fmt_char*.

Parameters *fmt_char* (*str*) – The format character.

Return type *DataType*

Returns The DataType for the format character.

Raises **KeyError** – If no corresponding DataType is found for the format character.

pumpkin_supmcu.supmcu.**datatype_to_supmcu_fmt_char**(*data_type*)

Converts *data_type* to the corresponding SupMCU format character.

Parameters *data_type* (*DataType*) – The DataType to get the corresponding format character for.

Return type *str*

Returns The format character as used in SUP/MOD:TEL? #,FORMAT commands.

Raises **KeyError** – If no corresponding SupMCU format character is found for the *data_type*.

2.2 Telemetry Parsing

The telemetry parsing system for the `pumpkin_supmcu` package uses a set of `DataItemParser` to parse the various different data formats returned from the SupMCU modules. These are used by the `pumpkin_supmcu.parse_telemetry()` method in conjunction with the `pumpkin_supmcu.SupMCUTelemetryDefinition` in order to parse any set of telemetry from the SupMCU modules.

class `pumpkin_supmcu.supmcu.DataItemParser`

Parses a series of bytes for a given SupMCU *fmt_specifier* returning the corresponding python type.

abstract `parse(b)`

Parses the byte array for a `DataItem` and returns the left-over bytes after parsing the data item.

Parameters `b (bytes)` – The input bytes to parse the data item from

Return type `Tuple[TelemetryDataItem, bytes]`

Returns A tuple of the data item parsed and remaining bytes

`pumpkin_supmcu.supmcu.Parsers` Contains a mapping of SupMCU format character to appropriate parser for the format specifier.

`pumpkin_supmcu.supmcu.parse_telemetry(b, supmcu_telemetry_def)`

Parses the bytes `b` as SupMCU telemetry using the given `supmcu_telemetry_def` format string or definition.

Parameters

- `b (bytes)` – The telemetry bytes to parse.
- `supmcu_telemetry_def (Union[str, SupMCUTelemetryDefinition])` – The SupMCU format string or definition to use.

Return type `SupMCUTelemetry`

Returns A SupMCU parsed telemetry item.

`pumpkin_supmcu.supmcu.parse_header(b)`

Parses the SupMCU Telemetry header from the bytes and returns the `SupMCUHDR` object and the left-over bytes.

Parameters `b (bytes)` – The bytes to parse the SupMCU header from.

Return type `Tuple[SupMCUHDR, bytes]`

Returns A tuple of the `SupMCUHDR` object and the left-over bytes.

2.3 SupMCU Discovery

The discovery modules allow for automated discovery of all SupMCU and module telemetry definitions for a given I2C address. The list of telemetry items can be serialized after the fact and loaded again at a later time to avoid the lengthy discovery process.

`pumpkin_supmcu.supmcu.request_module_definition(i2c_master, address, module_cmd_name=None, module_name=None, response_delay=None)`

Requests all of the telemetry definitions from the module at I2C Address `address`, using `module_cmd_name` when requesting module telemetry definitions.

Parameters

- `i2c_master (Union[I2CMaster, SupMCUMaster])` – The I2C master to write/read the requests from.

- **address** (`int`) – The address of the module on the I2C bus.
- **module_cmd_name** (`Optional[str]`) – Optional, short name of the module as used in telemetry requests (e.g. BM for Battery Module).
- **module_name** (`Optional[str]`) – Optional name to give module, if None, then is set to *module_cmd_name*
- **response_delay** (`Optional[float]`) – The delay in seconds to wait between I2C read and I2C write.

Return type *SupMCUModuleDefinition*

Returns The module definition for the device at I2C Address *address*

```
pumpkin_supmcu.supmcu.request_telemetry_definition(i2c_master, address, module_cmd_name, idx,
                                                    response_delay=None,
                                                    is_simulatable_mod=False)
```

Requests the formatting, name and length information from the device at I2C address *address*, using the module short name *module_cmd_name* (e.g. BM for Battery Module), concatenating that with *idx* in a telemetry request s.t. cmd_to_send is *<module_cmd_name>:TEL? <idx>,NAME/FORMAT/LENGTH*

Parameters

- **i2c_master** (`Union[I2CMaster, SupMCUMaster]`) – The I2CMaster device to use.
- **address** (`int`) – The address of the device to request information from.
- **module_cmd_name** (`str`) – The module name used in the context of SCPI commands (e.g. DCPS for Desktop CubeSat Power Supply).
- **idx** (`int`) – The telemetry index to grab the information for.
- **response_delay** (`Optional[float]`) – The amount of time in seconds to wait between I2C Write and read. Can be None, or set from SupMCUMaster passed in as *i2c_master*.
- **is_simulatable_mod** (`bool`) – Whether or not the module is simulatable.

Return type *SupMCUTelemetryDefinition*

Returns The SupMCUTelemetryDefinition that represents the Telemetry data.

```
pumpkin_supmcu.supmcu.get_values(i2c_master, address, module_cmd_name, idx, fmt, response_delay=None)
```

Retrieves the current values of the telemetry object that is indicated by the provided index

Parameters

- **i2c_master** (`I2CMaster`) – The I2CMaster device to use.
- **address** (`int`) – The address of the device to request information from.
- **module_cmd_name** (`str`) – The module name used in the context of SCPI commands (e.g. DCPS for Desktop CubeSat Power Supply).
- **idx** (`int`) – The telemetry index to grab the information for.
- **response_delay** (`Optional[float]`) – The amount of time in seconds to wait between I2C Write and read. Can be None, or set from SupMCUMaster passed in as *i2c_master*.

Return type `List[TelemetryDataItem]`

Returns A list of *TelemetryDataItem*

2.4 SupMCU interface

The SupMCU interface is the responsibility of the `SupMCUMaster` class. This provides the interface to request telemetry and write commands that are registered with the `SupMCUMaster`. Note that the telemetry definitions need to be discovered via the `request_telemetry_definition()` for single telemetry items or `request_module_definition()` for whole modules.

class `pumpkin_supmcu.supmcu.SupMCUMaster(i2c_master, supmcu_modules=None, request_delay=0.1)`

An interface into communicating to SupMCU modules via I2CMaster object.

Variables `i2c_master` – The underling I2CMaster device used to communicate with the I2C bus.

property `request_delay: float`

The amount of delay in seconds that is made between a telemetry write and read request.

Return type `float`

Returns The amount in fractional seconds between the TEL? write and read I2C transactions.

request_telemetry(*module, tel_type, idx*)

Requests the telemetry of *tel_type* at index *idx* from the *module*. *module* can be a I2C address, command name or the name of a module contained in *supmcu_modules*.

This will write the I2C request to the I2C Master, then wait *self.request_delay* seconds before reading the telemetry back from the I2C Master.

Parameters

- **module** (`Union[int, str]`) – The I2C address/command_name or name of the module to request telemetry from.
- **tel_type** (`TelemetryType`) – The type of telemetry being requested.
- **idx** (`int`) – The telemetry index being requested.

Return type `SupMCUTelemetry`

Returns The backing `SupMCUTelemetry` object from the telemetry request.

send_command(*module, cmd*)

Sends the SCPI command *cmd* to the *module* given. The module must be in the list of the registered modules *supmcu_modules*.

Parameters

- **module** (`Union[int, str]`) – The I2C, cmd_name or name of the module to send the command to.
- **cmd** (`str`) – The command to send to the module.

property `supmcu_modules:`

`Iterable[pumpkin_supmcu.supmcu.types.SupMCUModuleDefinition]`

Returns the list of `SupMCUModuleDefinitions` known by the `SupMCUMaster`.

Return type `Iterable[SupMCUModuleDefinition]`

EXAMPLES

Warning: These examples assume that you have already installed pumpkin-supmcpu with `pip3 install pumpkin-supmcpu` on python 3.6 or above.

Note: Just want the code? See [this folder](#)

3.1 Retrieving Telemetry Data

This example will walk through initializing a *I2CMaster* and using it to get the current values of all the telemetry of each module connected to the I2C bus.

```
from pumpkin_supmcpu.i2c import I2CDriverMaster, I2CLinuxMaster
from pumpkin_supmcpu.supmcpu import request_module_definition, get_values

PORT = 'COM4'

i2c_master = I2CDriverMaster(PORT)
```

This code imports everything that is needed from pumpkin-supmcpu and initializes an I2C master. PORT's value depends on what kind of I2C master is being initialized. For the *I2CDriverMaster*, PORT is what serial port the *I2CDriver* is, and can be a COM port on windows or a `/dev/ttyUSB*` filename on linux. For the *I2CLinuxMaster*, PORT is the number of the linux `/dev/i2c-*` interface. For instance, if the module was connected to `/dev/i2c-2` then PORT should be set to 2. Different I2C masters can be used interchangeably, after their initialization.

```
addresses = i2c_master.get_bus_devices()

module_definitions = []
for address in addresses:
    module_definitions.append((address, request_module_definition(i2c_master, address)))
```

This block gets the addresses of all active devices on the I2C bus, and then uses *request_module_definition* to save a *SupMCUModuleDefinition* for each device.

```
for addr, mod_def in module_definitions:
    for idx, telem in mod_def.supmcpu_telemetry.items():
        if telem.simulatable:
```

(continues on next page)

(continued from previous page)

```
items = get_values(i2c_master, addr, mod_def.cmd_name, idx, telem.format)
for item in items:
    print(item.string_value, end=' ')
print()
else:
    print("Not simulatable, might be a garbage value")
```

This code iterates through each telemetry object in each module definition and uses *get_values* to print its values if it's simulatable. When the telemetry isn't simulatable, *get_values* will return garbage data on QSMs or STMs.

The *pumpkin_supmcu* package has the following functionality:

- Discover all telemetry items on a module, and put into telemetry definitions.
- Parse any telemetry item on a module given a module telemetry definition.
- Provide a universal I2C interface so we can integrate multiple I2C Masters.
- Request telemetry from any SupMCU module via I2C interface.
- Write commands to SupMCU modules via I2C interface.

The *pumpkin_supmcu.i2c* package provides *I2CMaster* implementations for the *I2CDriver Board* or (*support coming soon*) *Total Phase Aardvark Adaptor*.

Users are encouraged to contribute more implementations of other I2CMaster devices as *pumpkin_supmcu* provides a *I2CMaster Protocol* to implement (see [PEP 544](#))

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

- `pumpkin_supmcu.i2c`, 1
- `pumpkin_supmcu.i2c.master`, 1
- `pumpkin_supmcu.supmcu`, 3
- `pumpkin_supmcu.supmcu.discovery`, 6
- `pumpkin_supmcu.supmcu.i2c`, 8
- `pumpkin_supmcu.supmcu.parsing`, 6
- `pumpkin_supmcu.supmcu.types`, 3

A

`address` (*pumpkin_supmcu.supmcu.SupMCUModuleDefinition* attribute), 5

C

`Char` (*pumpkin_supmcu.supmcu.DataType* attribute), 3
`cmd_name` (*pumpkin_supmcu.supmcu.SupMCUModuleDefinition* attribute), 5

D

`data_type` (*pumpkin_supmcu.supmcu.TelemetryDataItem* attribute), 4
`DataItemParser` (class in *pumpkin_supmcu.supmcu*), 6
`DataType` (class in *pumpkin_supmcu.supmcu*), 3
`datatype_to_supmcu_fmt_char()` (in module *pumpkin_supmcu.supmcu*), 5
`device_name` (*pumpkin_supmcu.i2c.I2CMaster* property), 1
`device_pullups` (*pumpkin_supmcu.i2c.I2CMaster* property), 1
`device_speed` (*pumpkin_supmcu.i2c.I2CMaster* property), 1
`Double` (*pumpkin_supmcu.supmcu.DataType* attribute), 3

F

`Fast` (*pumpkin_supmcu.i2c.I2CBusSpeed* attribute), 1
`Float` (*pumpkin_supmcu.supmcu.DataType* attribute), 3
`format` (*pumpkin_supmcu.supmcu.SupMCUTelemetryDefinition* attribute), 4

G

`get_bus_devices()` (*pumpkin_supmcu.i2c.I2CMaster* method), 1
`get_values()` (in module *pumpkin_supmcu.supmcu*), 7

H

`header` (*pumpkin_supmcu.supmcu.SupMCUTelemetry* attribute), 4
`Hex16` (*pumpkin_supmcu.supmcu.DataType* attribute), 3
`Hex8` (*pumpkin_supmcu.supmcu.DataType* attribute), 3

I

`I2CBusSpeed` (class in *pumpkin_supmcu.i2c*), 1
`I2CMaster` (class in *pumpkin_supmcu.i2c*), 1
`idx` (*pumpkin_supmcu.supmcu.SupMCUTelemetryDefinition* attribute), 4
`INT16` (*pumpkin_supmcu.supmcu.DataType* attribute), 3
`INT32` (*pumpkin_supmcu.supmcu.DataType* attribute), 3
`INT64` (*pumpkin_supmcu.supmcu.DataType* attribute), 3
`INT8` (*pumpkin_supmcu.supmcu.DataType* attribute), 3
`items` (*pumpkin_supmcu.supmcu.SupMCUTelemetry* attribute), 4

M

module
 pumpkin_supmcu.i2c, 1
 pumpkin_supmcu.i2c.master, 1
 pumpkin_supmcu.supmcu, 3
 pumpkin_supmcu.supmcu.discovery, 6
 pumpkin_supmcu.supmcu.i2c, 8
 pumpkin_supmcu.supmcu.parsing, 6
 pumpkin_supmcu.supmcu.types, 3
`Module` (*pumpkin_supmcu.supmcu.TelemetryType* attribute), 4
`module_telemetry` (*pumpkin_supmcu.supmcu.SupMCUModuleDefinition* attribute), 5

N

`name` (*pumpkin_supmcu.supmcu.SupMCUModuleDefinition* attribute), 5
`name` (*pumpkin_supmcu.supmcu.SupMCUTelemetryDefinition* attribute), 4

P

`parse()` (*pumpkin_supmcu.supmcu.DataItemParser* method), 6
`parse_header()` (in module *pumpkin_supmcu.supmcu*), 6
`parse_telemetry()` (in module *pumpkin_supmcu.supmcu*), 6
pumpkin_supmcu.i2c
 module, 1

pumpkin_supmcu.i2c.master
 module, 1
 pumpkin_supmcu.supmcu
 module, 3
 pumpkin_supmcu.supmcu.discovery
 module, 6
 pumpkin_supmcu.supmcu.i2c
 module, 8
 pumpkin_supmcu.supmcu.Parsers (in module pump-
 kin_supmcu.supmcu.parsing), 6
 pumpkin_supmcu.supmcu.parsing
 module, 6
 pumpkin_supmcu.supmcu.types
 module, 3

R

read() (pumpkin_supmcu.i2c.I2CMaster method), 2
 ready (pumpkin_supmcu.supmcu.SupMCUHDR at-
 tribute), 4
 request_delay (pump-
 kin_supmcu.supmcu.SupMCUMaster prop-
 erty), 8
 request_module_definition() (in module pump-
 kin_supmcu.supmcu), 6
 request_telemetry() (pump-
 kin_supmcu.supmcu.SupMCUMaster method),
 8
 request_telemetry_definition() (in module pump-
 kin_supmcu.supmcu), 7

S

send_command() (pump-
 kin_supmcu.supmcu.SupMCUMaster method),
 8
 sizeof_supmcu_type() (in module pump-
 kin_supmcu.supmcu), 5
 Standard (pumpkin_supmcu.i2c.I2CBusSpeed at-
 tribute), 1
 Str (pumpkin_supmcu.supmcu.DataType attribute), 3
 string_value (pump-
 kin_supmcu.supmcu.TelemetryDataItem
 attribute), 4
 SupMCU (pumpkin_supmcu.supmcu.TelemetryType at-
 tribute), 4
 supmcu_modules (pump-
 kin_supmcu.supmcu.SupMCUMaster prop-
 erty), 8
 supmcu_telemetry (pump-
 kin_supmcu.supmcu.SupMCUModuleDefinition
 attribute), 5
 SupMCUHDR (class in pumpkin_supmcu.supmcu), 4
 SupMCUMaster (class in pumpkin_supmcu.supmcu), 8
 SupMCUModuleDefinition (class in pump-
 kin_supmcu.supmcu), 4

SupMCUTelemetry (class in pumpkin_supmcu.supmcu),
 4
 SupMCUTelemetryDefinition (class in pump-
 kin_supmcu.supmcu), 4

T

telemetry_length (pump-
 kin_supmcu.supmcu.SupMCUTelemetryDefinition
 attribute), 4
 TelemetryDataItem (class in pump-
 kin_supmcu.supmcu), 4
 TelemetryType (class in pumpkin_supmcu.supmcu), 4
 timestamp (pumpkin_supmcu.supmcu.SupMCUHDR at-
 tribute), 4
 typeof_supmcu_fmt_char() (in module pump-
 kin_supmcu.supmcu), 5

U

UINT16 (pumpkin_supmcu.supmcu.DataType attribute), 3
 UINT32 (pumpkin_supmcu.supmcu.DataType attribute), 3
 UINT64 (pumpkin_supmcu.supmcu.DataType attribute), 3
 UINT8 (pumpkin_supmcu.supmcu.DataType attribute), 4

V

value (pumpkin_supmcu.supmcu.TelemetryDataItem at-
 tribute), 4

W

write() (pumpkin_supmcu.i2c.I2CMaster method), 2