ArduinoSMBus

1.0

Generated by Doxygen 1.10.0

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 ArduinoSMBus Class Reference

Public Member Functions

- ArduinoSMBus (uint8_t batteryAddress)

  Construct a new ArduinoSMBus:: ArduinoSMBus object.
- void setBatteryAddress (uint8_t batteryAddress)

  Set the battery's I2C address. Can be used to change the address after the object is created.
- uint16_t temperature ()

  Get the battery's temperature. Returns the battery temperature in 0.1 degrees Kelvin.
- uint16_t temperatureC ()

  Get the battery's temperature in Celsius. Returns the battery temperature in 0.1 degrees Celsius.
- uint16_t temperatureF ()

  Get the battery's temperature in Fahrenheit. Returns the battery temperature in 0.1 degrees Fahrenheit.
- uint16_t voltage ()

  Get the battery's voltage. Returns the sum of all cell voltages, in mV.
- uint16_t current ()

  Get the battery's current. Returns the battery measured current (from the coulomb counter) in mA.
- uint16_t averageCurrent ()

  Get the battery's average current. Returns the average current in a 1-minute rolling average, in mA.
- uint16_t maxError ()

  Get the battery's state of charge error. Returns the battery's margin of error when estimating SOC, in percent.
- uint16_t relativeStateOfCharge ()

  Get the battery's current relative charge. Returns the predicted remaining battery capacity as a percentage of full↩ ChargeCapacity()
- uint16_t absoluteStateOfCharge ()

  Get the battery's absolute charge. Returns the predicted remaining battery capacity as a percentage of designCapacity()
- uint16_t remainingCapacity ()

  Get the battery's capacity. Returns the predicted battery capacity when fully charged, in mAh. For some batteries, this may be in 10s of mWh, if the BatteryMode() register (0x03) is set to CAPM 1. See protocol documentation for details.
- uint16_t fullCapacity ()

  Get the battery's full capacity. Returns the predicted battery capacity when fully charged, in mAh. For some batteries, this may be in 10s of mWh, if the BatteryMode() register (0x03) is set to CAPM 1. See protocol documentation for details.

- uint16_t runTimeToEmpty ()

  Get the battery's time to empty. Returns the predicted time to empty, in minutes, based on current instantaneous discharge rate.
- uint16_t avgTimeToEmpty ()

  Get the battery's average time to empty. Returns the predicted time to empty, in minutes, based on 1-minute rolling average discharge rate.
- uint16_t avgTimeToFull ()

  Get the battery's time to full. Returns the predicted time to full charge, in minutes, based on 1-minute rolling average charge rate.
- uint16_t batteryStatus ()

  Get the Status from the battery. Returns the battery status register, which contains various alarm conditions and other status bits.
- uint16_t **chargingCurrent** ()
- uint16_t **chargingVoltage** ()
- bool statusOK ()

  Check if the battery status is OK. Check for any alarm conditions in the battery status register. These include bits 8, 9, 11, 12, 14, and 15. If any of these bits are set, the battery is not in error.
- bool isCharging ()

  Check if the battery is charging.
- bool isFullyCharged ()

  Check if the battery is fully charged.
- uint16_t cycleCount ()

  Get the battery's cycle count. Returns the number of discharge cycles the battery has experienced. A cycle is defined as an amount of discharge equal to the battery's design capacity.
- uint16_t designCapacity ()

  Get the battery's design capacity. Returns the theoretical maximum capacity of the battery, in mAh. For some batteries, this may be in 10 mWh, if the BatteryMode() register (0x03) is set to CAPM 1. See TI protocol documentation for details.
- uint16_t designVoltage ()

  Get the battery's design voltage. Returns the nominal voltage of the battery, in mV.
- uint16_t manufactureDate ()

  Get the battery's manufacture date. Returns the date the battery was manufactured, in the following format: Day + Month∗32 + (Year–1980)∗512.
- int manufactureYear ()

  Get the manufacture year from the manufacture date.
- uint16_t serialNumber ()

  Get the Serial Number from the battery.
- const char ∗ manufacturerName ()

  Get the Manufacturer Name from the battery.
- const char ∗ deviceName ()

  Get the Device Name from the battery.
- const char ∗ deviceChemistry ()

  Get the Device Chemistry from the battery.
- uint16_t stateOfHealth ()

  Get the State of Health from the battery. Returns the estimated health of the battery, as a percentage of design capacity This command is not supported by all batteries.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 ArduinoSMBus()

```
ArduinoSMBus::ArduinoSMBus (
            uint8_t batteryAddress )
```

Construct a new ArduinoSMBus:: ArduinoSMBus object.

Parameters

batteryAddress	

3.1.2 Member Function Documentation

3.1.2.1 absoluteStateOfCharge()

```
uint16_t ArduinoSMBus::absoluteStateOfCharge ( )
```

Get the battery's absolute charge. Returns the predicted remaining battery capacity as a percentage of designCapacity()

Returns

uint16_t

3.1.2.2 averageCurrent()

```
uint16_t ArduinoSMBus::averageCurrent ( )
```

Get the battery's average current. Returns the average current in a 1-minute rolling average, in mA.

Returns

uint16_t

3.1.2.3 avgTimeToEmpty()

```
uint16_t ArduinoSMBus::avgTimeToEmpty ( )
```

Get the battery's average time to empty. Returns the predicted time to empty, in minutes, based on 1-minute rolling average discharge rate.

Returns

uint16_t

3.1.2.4 avgTimeToFull()

```
uint16_t ArduinoSMBus::avgTimeToFull ( )
```

Get the battery's time to full. Returns the predicted time to full charge, in minutes, based on 1-minute rolling average charge rate.

Returns

uint16_t

3.1.2.5 batteryStatus()

```
uint16_t ArduinoSMBus::batteryStatus ( )
```

Get the Status from the battery. Returns the battery status register, which contains various alarm conditions and other status bits.

Returns

uint16_t

3.1.2.6 current()

```
uint16_t ArduinoSMBus::current ( )
```

Get the battery's current. Returns the battery measured current (from the coulomb counter) in mA.

Returns

uint16_t

3.1.2.7 cycleCount()

```
uint16_t ArduinoSMBus::cycleCount ( )
```

Get the battery's cycle count. Returns the number of discharge cycles the battery has experienced. A cycle is defined as an amount of discharge equal to the battery's design capacity.

Returns

uint16_t

3.1.2.8 designCapacity()

```
uint16_t ArduinoSMBus::designCapacity ( )
```

Get the battery's design capacity. Returns the theoretical maximum capacity of the battery, in mAh. For some batteries, this may be in 10 mWh, if the BatteryMode() register (0x03) is set to CAPM 1. See TI protocol documentation for details.

Returns

uint16_t

3.1.2.9 designVoltage()

```
uint16_t ArduinoSMBus::designVoltage ( )
```

Get the battery's design voltage. Returns the nominal voltage of the battery, in mV.

Returns

uint16_t

3.1.2.10 deviceChemistry()

```
const char * ArduinoSMBus::deviceChemistry ( )
```

Get the Device Chemistry from the battery.

Returns

const char*

3.1.2.11 deviceName()

```
const char * ArduinoSMBus::deviceName ( )
```

Get the Device Name from the battery.

Returns

const char*

3.1.2.12 fullCapacity()

```
uint16_t ArduinoSMBus::fullCapacity ( )
```

Get the battery's full capacity. Returns the predicted battery capacity when fully charged, in mAh. For some batteries, this may be in 10s of mWh, if the BatteryMode() register (0x03) is set to CAPM 1. See protocol documentation for details.

Returns

uint16_t

3.1.2.13 isCharging()

```
bool ArduinoSMBus::isCharging ( )
```

Check if the battery is charging.

Returns

bool

3.1.2.14 isFullyCharged()

```
bool ArduinoSMBus::isFullyCharged ( )
```

Check if the battery is fully charged.

Returns

bool

3.1.2.15 manufactureDate()

```
uint16_t ArduinoSMBus::manufactureDate ( )
```

Get the battery's manufacture date. Returns the date the battery was manufactured, in the following format: Day + Month∗32 + (Year–1980)∗512.

Returns

uint16_t

3.1.2.16 manufacturerName()

```
const char * ArduinoSMBus::manufacturerName ( )
```

Get the Manufacturer Name from the battery.

Returns

const char∗

3.1.2.17 manufactureYear()

```
int ArduinoSMBus::manufactureYear ( )
```

Get the manufacture year from the manufacture date.

Returns

int

3.1.2.18 maxError()

```
uint16_t ArduinoSMBus::maxError ( )
```

Get the battery's state of charge error. Returns the battery's margin of error when estimating SOC, in percent.

Returns

uint16_t

3.1.2.19 relativeStateOfCharge()

```
uint16_t ArduinoSMBus::relativeStateOfCharge ( )
```

Get the battery's current relative charge. Returns the predicted remaining battery capacity as a percentage of fullChargeCapacity()

Returns

uint16_t

3.1.2.20 remainingCapacity()

```
uint16_t ArduinoSMBus::remainingCapacity ( )
```

Get the battery's capacity. Returns the predicted battery capacity when fully charged, in mAh. For some batteries, this may be in 10s of mWh, if the BatteryMode() register (0x03) is set to CAPM 1. See protocol documentation for details.

Returns

uint16_t

3.1.2.21 runTimeToEmpty()

```
uint16_t ArduinoSMBus::runTimeToEmpty ( )
```

Get the battery's time to empty. Returns the predicted time to empty, in minutes, based on current instantaneous discharge rate.

Returns

uint16_t

3.1.2.22 serialNumber()

```
uint16_t ArduinoSMBus::serialNumber ( )
```

Get the Serial Number from the battery.

Returns

uint16_t

3.1.2.23 setBatteryAddress()

```
void ArduinoSMBus::setBatteryAddress (
            uint8_t batteryAddress )
```

Set the battery's I2C address. Can be used to change the address after the object is created.

Parameters

batteryAddress	

3.1.2.24 stateOfHealth()

```
uint16_t ArduinoSMBus::stateOfHealth ( )
```

Get the State of Health from the battery. Returns the estimated health of the battery, as a percentage of design capacity This command is not supported by all batteries.

Returns

uint16_t

3.1.2.25 statusOK()

```
bool ArduinoSMBus::statusOK ( )
```

Check if the battery status is OK. Check for any alarm conditions in the battery status register. These include bits 8, 9, 11, 12, 14, and 15. If any of these bits are set, the battery is not in error.

Returns

bool

3.1.2.26 temperature()

```
uint16_t ArduinoSMBus::temperature ( )
```

Get the battery's temperature. Returns the battery temperature in 0.1 degrees Kelvin.

Returns

uint16_t

3.1.2.27 temperatureC()

```
uint16_t ArduinoSMBus::temperatureC ( )
```

Get the battery's temperature in Celsius. Returns the battery temperature in 0.1 degrees Celsius.

Returns

uint16_t

3.1.2.28 temperatureF()

```
uint16_t ArduinoSMBus::temperatureF ( )
```

Get the battery's temperature in Fahrenheit. Returns the battery temperature in 0.1 degrees Fahrenheit.

Returns

uint16_t

3.1.2.29 voltage()

```
uint16_t ArduinoSMBus::voltage ( )
```

Get the battery's voltage. Returns the sum of all cell voltages, in mV.

Returns

uint16_t

The documentation for this class was generated from the following files:

- C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/src/ArduinoSMBus.h
- C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/src/ArduinoSMBus.cpp

Chapter 4

File Documentation

4.1 C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/src/ArduinoSMBus.cpp File Reference

Function definitions for the ArduinoSMBus class.

```
#include "ArduinoSMBus.h"
```

4.1.1 Detailed Description

Function definitions for the ArduinoSMBus class.

Author

Christopher Lee (`clee@unitedconsulting.com`)

Version

1.0

Date

2024-02-29

Copyright

Copyright (c) 2024

4.2 C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/src/ArduinoSMBus.h File Reference

Function declarations for the ArduinoSMBus class.

```
#include <Arduino.h>
#include <Wire.h>
```

Classes

- class ArduinoSMBus

Macros

- #define **TEMPERATURE** 0x08
- #define **VOLTAGE** 0x09
- #define **CURRENT** 0x0a
- #define **AVERAGE_CURRENT** 0x0b
- #define **MAX_ERROR** 0x0c
- #define **REL_STATE_OF_CHARGE** 0x0d
- #define **ABS_STATE_OF_CHARGE** 0x0e
- #define **REM_CAPACITY** 0x0f
- #define **FULL_CAPACITY** 0x10
- #define **RUN_TIME_TO_EMPTY** 0x11
- #define **AVG_TIME_TO_EMPTY** 0x12
- #define **AVG_TIME_TO_FULL** 0x13
- #define **BATTERY_STATUS** 0x16
- #define **CHARGING_CURRENT** 0x14
- #define **CHARGING_VOLTAGE** 0x15
- #define **CYCLE_COUNT** 0x17
- #define **DESIGN_CAPACITY** 0x18
- #define **DESIGN_VOLTAGE** 0x19
- #define **MANUFACTURE_DATE** 0x1b
- #define **SERIAL_NUMBER** 0x1c
- #define **MANUFACTURER_NAME** 0x20
- #define **DEVICE_NAME** 0x21
- #define **DEVICE_CHEMISTRY** 0x22
- #define **STATE_OF_HEALTH** 0x4f

4.2.1 Detailed Description

Function declarations for the ArduinoSMBus class.

Author

Christopher Lee (`clee@unitedconsulting.com`)

Version

1.0

Date

2024-02-29

Copyright

Copyright (c) 2024

4.3 ArduinoSMBus.h

Go to the documentation of this file.

```

#ifndef ArduinoSMBus_h
#define ArduinoSMBus_h

#include <Arduino.h>
#include <Wire.h>

//Usable Commands
#define TEMPERATURE 0x08
#define VOLTAGE 0x09
#define CURRENT 0x0a
#define AVERAGE_CURRENT 0x0b
#define MAX_ERROR 0x0c
#define REL_STATE_OF_CHARGE 0x0d
#define ABS_STATE_OF_CHARGE 0x0e
#define REM_CAPACITY 0x0f
#define FULL_CAPACITY 0x10
#define RUN_TIME_TO_EMPTY 0x11
#define AVG_TIME_TO_EMPTY 0x12
#define AVG_TIME_TO_FULL 0x13
#define BATTERY_STATUS 0x16
#define CHARGING_CURRENT 0x14
#define CHARGING_VOLTAGE 0x15
#define CYCLE_COUNT 0x17
#define DESIGN_CAPACITY 0x18
#define DESIGN_VOLTAGE 0x19
#define MANUFACTURE_DATE 0x1b
#define SERIAL_NUMBER 0x1c
#define MANUFACTURER_NAME 0x20
#define DEVICE_NAME 0x21
#define DEVICE_CHEMISTRY 0x22
#define STATE_OF_HEALTH 0x4f

class ArduinoSMBus {
public:
  ArduinoSMBus(uint8_t batteryAddress);
  void setBatteryAddress(uint8_t batteryAddress);

  uint16_t temperature();
  uint16_t temperatureC();
  uint16_t temperatureF();
  uint16_t voltage();
  uint16_t current();
  uint16_t averageCurrent();
  uint16_t maxError();
  uint16_t relativeStateOfCharge();
  uint16_t absoluteStateOfCharge();
  uint16_t remainingCapacity();
  uint16_t fullCapacity();
  uint16_t runTimeToEmpty();
  uint16_t avgTimeToEmpty();
  uint16_t avgTimeToFull();
  uint16_t batteryStatus();
  uint16_t chargingCurrent();
  uint16_t chargingVoltage();
  bool statusOK();
  bool isCharging();
  bool isFullyCharged();
  uint16_t cycleCount();
  uint16_t designCapacity();
  uint16_t designVoltage();
  uint16_t manufactureDate();
  int manufactureYear();
  uint16_t serialNumber();
  const char* manufacturerName();
  const char* deviceName();
  const char* deviceChemistry();
  uint16_t stateOfHealth();

private:
  uint8_t _batteryAddress;
  uint16_t readRegister(uint8_t reg);
  void readBlock(uint8_t reg, uint8_t* data, uint8_t len);
};

#endif
```

Index