ArduinoSMBus

1.1

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 remainingCapacityAlarm ()

  Get the battery's remaining capacity alarm. Returns the battery's remaining capacity alarm threshold value, in mAh.
- uint16_t remainingTimeAlarm ()

  Get the battery's remaining time alarm. Returns the battery's remaining time alarm threshold value, in minutes.
- BatteryMode batteryMode ()

  Get the battery's mode.
- uint16_t temperature ()

  Get the battery's temperature. Returns the battery temperature in 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 that way 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 that way 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.

- BatteryStatus batteryStatus ()

  Get the battery's status.

- uint16_t chargingCurrent ()

  Get the battery's design charging current. Returns the desired design charging current of the battery, in mA.

- uint16_t chargingVoltage ()

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

- bool statusOK ()

  Check if the battery status is OK. Check for any alarm conditions in the battery status. These include over charge, termination charge, over temperature, termination discharge alarms. If any of these alarms are set, the battery is not OK.

- 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.

Public Attributes

- BatteryMode **battery_mode**

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 batteryMode()

```
BatteryMode ArduinoSMBus::batteryMode ( )
```

Get the battery's mode.

This method reads the battery's mode register, which contains various settings and status bits. It then creates a BatteryMode struct and sets its fields based on the bits in the mode.

Returns

BatteryMode A struct containing the following fields:

- internal_charge_controller: bit 0 of the mode register
- primary_battery_support: bit 1 of the mode register
- condition_flag: bit 7 of the mode register
- charge_controller_enabled: bit 8 of the mode register
- primary_battery: bit 9 of the mode register
- alarm_mode: bit 13 of the mode register
- charger_mode: bit 14 of the mode register
- capacity_mode: bit 15 of the mode register

3.1.2.6 batteryStatus()

```
BatteryStatus ArduinoSMBus::batteryStatus ( )
```

Get the battery's status.

This function reads the BatteryStatus register and returns a struct with its value. The BatteryStatus register indicates various alarm conditions and states of the battery. These include over charge, termination charge, over temperature, termination discharge, remaining capacity, remaining time, initialization, discharging, fully charged, and fully discharged states.

Returns

BatteryStatus A struct containing the status of each bit in the BatteryStatus register.

3.1.2.7 chargingCurrent()

```
uint16_t ArduinoSMBus::chargingCurrent ( )
```

Get the battery's design charging current. Returns the desired design charging current of the battery, in mA.

Returns

uint16_t

3.1.2.8 chargingVoltage()

```
uint16_t ArduinoSMBus::chargingVoltage ( )
```

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

Returns

uint16_t

3.1.2.9 current()

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

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

Returns

uint16_t

3.1.2.10 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

3.1.2.11 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

3.1.2.12 designVoltage()

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

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

Returns

uint16_t

3.1.2.13 deviceChemistry()

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

Get the Device Chemistry from the battery.

Returns

const char*

3.1.2.14 deviceName()

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

Get the Device Name from the battery.

Returns

const char*

3.1.2.15 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 that way See protocol documentation for details.

Returns

uint16_t

3.1.2.16 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

3.1.2.17 manufacturerName()

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

Get the Manufacturer Name from the battery.

Returns

const char*

3.1.2.18 manufactureYear()

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

Get the manufacture year from the manufacture date.

Returns

int

3.1.2.19 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

3.1.2.20 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

3.1.2.21 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 that way See protocol documentation for details.

Returns

uint16_t

3.1.2.22 remainingCapacityAlarm()

```
uint16_t ArduinoSMBus::remainingCapacityAlarm ( )
```

Get the battery's remaining capacity alarm. Returns the battery's remaining capacity alarm threshold value, in mAh.

Returns

uint16_t

3.1.2.23 remainingTimeAlarm()

```
uint16_t ArduinoSMBus::remainingTimeAlarm ( )
```

Get the battery's remaining time alarm. Returns the battery's remaining time alarm threshold value, in minutes.

Returns

uint16_t

3.1.2.24 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

3.1.2.25 serialNumber()

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

Get the Serial Number from the battery.

Returns

uint16_t

3.1.2.26 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	

3.1.2.27 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

3.1.2.28 statusOK()

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

Check if the battery status is OK. Check for any alarm conditions in the battery status. These include over charge, termination charge, over temperature, termination discharge alarms. If any of these alarms are set, the battery is not OK.

Returns

bool True if the battery status is OK, false otherwise.

3.1.2.29 temperature()

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

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

Returns

uint16_t

3.1.2.30 temperatureC()

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

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

Returns

uint16_t

3.1.2.31 temperatureF()

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

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

Returns

uint16_t

3.1.2.32 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:

- src/ArduinoSMBus.h
- src/ArduinoSMBus.cpp

3.2 BatteryMode Struct Reference

A struct to hold various battery mode flags.

```
#include <ArduinoSMBus.h>
```

Public Attributes

- bool internal_charge_controller
- bool primary_battery_support
- bool condition_flag
- bool charge_controller_enabled
- bool primary_battery
- bool alarm_mode
- bool charger_mode
- bool capacity_mode

3.2.1 Detailed Description

A struct to hold various battery mode flags.

3.2.2 Member Data Documentation

3.2.2.1 alarm_mode

```
bool BatteryMode::alarm_mode
```

True to enable alarmWarning broadcasts to host, false to disable.

3.2.2.2 capacity_mode

```
bool BatteryMode::capacity_mode
```

True to report in mA or mAh, false to report in 10mW or 10mWh units.

3.2.2.3 charge_controller_enabled

```
bool BatteryMode::charge_controller_enabled
```

True if the charge controller is enabled, false otherwise.

3.2.2.4 charger_mode

```
bool BatteryMode::charger_mode
```

True to enable chargingCurrent and chargingVoltage broadcasts to host, false to disable.

3.2.2.5 condition_flag

```
bool BatteryMode::condition_flag
```

False if condition is ok, true if battery conditioning cycle is needed.

3.2.2.6 internal_charge_controller

```
bool BatteryMode::internal_charge_controller
```

True if the internal charge controller is supported, false otherwise.

3.2.2.7 primary_battery

```
bool BatteryMode::primary_battery
```

True if the primary battery is enabled, false otherwise.

3.2.2.8 primary_battery_support

```
bool BatteryMode::primary_battery_support
```

True if the primary battery support is supported, false otherwise.

The documentation for this struct was generated from the following file:

- src/ArduinoSMBus.h

3.3 BatteryStatus Struct Reference

A struct to hold various battery status flags.

```
#include <ArduinoSMBus.h>
```

Public Attributes

- bool over_charged_alarm
- bool term_charge_alarm
- bool over_temp_alarm
- bool term_discharge_alarm
- bool rem_capacity_alarm
- bool rem_time_alarm
- bool initialized
- bool discharging
- bool fully_charged
- bool fully_discharged

3.3.1 Detailed Description

A struct to hold various battery status flags.

This struct holds various flags that represent the battery status.

3.3.2 Member Data Documentation

3.3.2.1 discharging

```
bool BatteryStatus::discharging
```

True if the battery is discharging, false otherwise. Corresponds to bit 6 of the BatteryStatus register.

3.3.2.2 fully_charged

```
bool BatteryStatus::fully_charged
```

True if the battery is fully charged, false otherwise. Corresponds to bit 5 of the BatteryStatus register.

3.3.2.3 fully_discharged

```
bool BatteryStatus::fully_discharged
```

True if the battery is fully discharged, false otherwise. Corresponds to bit 4 of the BatteryStatus register.

3.3.2.4 initialized

```
bool BatteryStatus::initialized
```

True if the battery is initialized, false otherwise. Corresponds to bit 7 of the BatteryStatus register.

3.3.2.5 over_charged_alarm

```
bool BatteryStatus::over_charged_alarm
```

True if the battery is overcharged, false otherwise. Corresponds to bit 15 of the BatteryStatus register.

3.3.2.6 over_temp_alarm

```
bool BatteryStatus::over_temp_alarm
```

True if the battery temperature is over the limit, false otherwise. Corresponds to bit 12 of the BatteryStatus register.

3.3.2.7 rem_capacity_alarm

```
bool BatteryStatus::rem_capacity_alarm
```

True if the remaining capacity alarm is set, false otherwise. Corresponds to bit 9 of the BatteryStatus register.

3.3.2.8 rem_time_alarm

```
bool BatteryStatus::rem_time_alarm
```

True if the remaining time alarm is set, false otherwise. Corresponds to bit 8 of the BatteryStatus register.

3.3.2.9 term_charge_alarm

```
bool BatteryStatus::term_charge_alarm
```

True if the termination charge alarm is set, false otherwise. Corresponds to bit 14 of the BatteryStatus register.

3.3.2.10 term_discharge_alarm

```
bool BatteryStatus::term_discharge_alarm
```

True if the termination discharge alarm is set, false otherwise. Corresponds to bit 11 of the BatteryStatus register.

The documentation for this struct was generated from the following file:

- src/ArduinoSMBus.h

Chapter 4

File Documentation

4.1 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

1.1

Date

2024-03-06

Copyright

Copyright (c) 2024

4.2 src/ArduinoSMBus.h File Reference

Function declarations for the ArduinoSMBus class.

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

Classes

- struct BatteryMode

  A struct to hold various battery mode flags.
- struct BatteryStatus

  A struct to hold various battery status flags.
- class ArduinoSMBus

Macros

- #define **MANUFACTURER_ACCESS** 0x00
- #define **REMAINING_CAPACITY_ALARM** 0x01
- #define **REMAINING_TIME_ALARM** 0x02
- #define **BATTERY_MODE** 0x03
- #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

1.1

Date

2024-03-06

Copyright

4.3 ArduinoSMBus.h

Go to the documentation of this file.

```
00001
00012 #ifndef ArduinoSMBus_h
00013 #define ArduinoSMBus_h
00014
00015 #include <Arduino.h>
00016 #include <Wire.h>
00017
00018  //Usable Commands
00019 #define MANUFACTURER_ACCESS 0x00
00020 #define REMAINING_CAPACITY_ALARM 0x01
00021 #define REMAINING_TIME_ALARM 0x02
00022 #define BATTERY_MODE 0x03
00023 #define TEMPERATURE 0x08
00024 #define VOLTAGE 0x09
00025 #define CURRENT 0x0a
00026 #define AVERAGE_CURRENT 0x0b
00027 #define MAX_ERROR 0x0c
00028 #define REL_STATE_OF_CHARGE 0x0d
00029 #define ABS_STATE_OF_CHARGE 0x0e
00030 #define REM_CAPACITY 0x0f
00031 #define FULL_CAPACITY 0x10
00032 #define RUN_TIME_TO_EMPTY 0x11
00033 #define AVG_TIME_TO_EMPTY 0x12
00034 #define AVG_TIME_TO_FULL 0x13
00035 #define BATTERY_STATUS 0x16
00036 #define CHARGING_CURRENT 0x14
00037 #define CHARGING_VOLTAGE 0x15
00038 #define CYCLE_COUNT 0x17
00039 #define DESIGN_CAPACITY 0x18
00040 #define DESIGN_VOLTAGE 0x19
00041 #define MANUFACTURE_DATE 0x1b
00042 #define SERIAL_NUMBER 0x1c
00043 #define MANUFACTURER_NAME 0x20
00044 #define DEVICE_NAME 0x21
00045 #define DEVICE_CHEMISTRY 0x22
00046 #define STATE_OF_HEALTH 0x4f
00047
00052 struct BatteryMode {
00053   bool internal_charge_controller;
00054   bool primary_battery_support;
00055   bool condition_flag;
00056   bool charge_controller_enabled;
00057   bool primary_battery;
00058   bool alarm_mode;
00059   bool charger_mode;
00060   bool capacity_mode;
00061 };
00062
00069 struct BatteryStatus {
00070   bool over_charged_alarm;
00071   bool term_charge_alarm;
00072   bool over_temp_alarm;
00073   bool term_discharge_alarm;
00074   bool rem_capacity_alarm;
00075   bool rem_time_alarm;
00076   bool initialized;
00077   bool discharging;
00078   bool fully_charged;
00079   bool fully_discharged;
00080 };
00081
00082 class ArduinoSMBus {
00083 public:
00084
00085
00086   BatteryMode battery_mode;
00087
00088   ArduinoSMBus(uint8_t batteryAddress);
00089   void setBatteryAddress(uint8_t batteryAddress);
00090
00091   uint16_t remainingCapacityAlarm();
00092   uint16_t remainingTimeAlarm();
00093   BatteryMode batteryMode();
00094   uint16_t temperature();
00095   uint16_t temperatureC();
00096   uint16_t temperatureF();
00097   uint16_t voltage();
00098   uint16_t current();
00099   uint16_t averageCurrent();
00100   uint16_t maxError();
00101   uint16_t relativeStateOfCharge();
00102   uint16_t absoluteStateOfCharge();
```

```
00103   uint16_t remainingCapacity();
00104   uint16_t fullCapacity();
00105   uint16_t runTimeToEmpty();
00106   uint16_t avgTimeToEmpty();
00107   uint16_t avgTimeToFull();
00108   BatteryStatus batteryStatus();
00109   uint16_t chargingCurrent();
00110   uint16_t chargingVoltage();
00111   bool statusOK();
00112   uint16_t cycleCount();
00113   uint16_t designCapacity();
00114   uint16_t designVoltage();
00115   uint16_t manufactureDate();
00116   int manufactureYear();
00117   uint16_t serialNumber();
00118   const char* manufacturerName();
00119   const char* deviceName();
00120   const char* deviceChemistry();
00121   uint16_t stateOfHealth();
00122
00123 private:
00124   uint8_t _batteryAddress;
00125   uint16_t readRegister(uint8_t reg);
00126   void readBlock(uint8_t reg, uint8_t* data, uint8_t len);
00127 };
00128
00129 #endif
```

4.4 src/main.cpp File Reference

Example arduino code to read battery data from an SMBus battery and print to serial output.

```
#include <Arduino.h>
#include "ArduinoSMBus.h"
```

Functions

- void **setup** ()
- void **loop** ()

Variables

- ArduinoSMBus **battery** (0x0B)

4.4.1 Detailed Description

Example arduino code to read battery data from an SMBus battery and print to serial output.

Author

Christopher Lee (clee@unitedconsulting.com)

Version

1.1

Date

2024-03-06

Copyright

Index