

# ArduinoSMBus

1.0

Generated by Doxygen 1.10.0



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 ArduinoSMBus Class Reference	5
3.1.1 Constructor & Destructor Documentation	7
3.1.1.1 ArduinoSMBus()	7
3.1.2 Member Function Documentation	7
3.1.2.1 absoluteStateOfCharge()	7
3.1.2.2 averageCurrent()	7
3.1.2.3 avgTimeToEmpty()	7
3.1.2.4 avgTimeToFull()	8
3.1.2.5 batteryMode()	8
3.1.2.6 batteryStatus()	8
3.1.2.7 chargingCurrent()	8
3.1.2.8 chargingVoltage()	9
3.1.2.9 current()	9
3.1.2.10 cycleCount()	9
3.1.2.11 designCapacity()	9
3.1.2.12 designVoltage()	10
3.1.2.13 deviceChemistry()	10
3.1.2.14 deviceName()	10
3.1.2.15 fullCapacity()	10
3.1.2.16 isCharging()	10
3.1.2.17 isFullyCharged()	11
3.1.2.18 manufactureDate()	11
3.1.2.19 manufacturerName()	11
3.1.2.20 manufactureYear()	11
3.1.2.21 maxError()	11
3.1.2.22 relativeStateOfCharge()	12
3.1.2.23 remainingCapacity()	12
3.1.2.24 remainingCapacityAlarm()	12
3.1.2.25 remainingTimeAlarm()	12
3.1.2.26 runTimeToEmpty()	13
3.1.2.27 serialNumber()	13
3.1.2.28 setBatteryAddress()	13
3.1.2.29 stateOfHealth()	13
3.1.2.30 statusOK()	14
3.1.2.31 temperature()	14
3.1.2.32 temperatureC()	14

---

3.1.2.33 temperatureF()	14
3.1.2.34 voltage()	15
3.2 BatteryMode Struct Reference	15
3.2.1 Detailed Description	15
3.2.2 Member Data Documentation	16
3.2.2.1 alarm_mode	16
3.2.2.2 capacity_mode	16
3.2.2.3 charge_controller_enabled	16
3.2.2.4 charger_mode	16
3.2.2.5 condition_flag	16
3.2.2.6 internal_charge_controller	16
3.2.2.7 primary_battery	17
3.2.2.8 primary_battery_support	17
<b>4 File Documentation</b>	<b>19</b>
4.1 C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/src/Arduino↔SMBus.cpp File Reference	19
4.1.1 Detailed Description	19
4.2 C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/src/Arduino↔SMBus.h File Reference	20
4.2.1 Detailed Description	21
4.3 ArduinoSMBus.h	21
4.4 C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/src/main.cpp File Reference	22
4.4.1 Detailed Description	23
<b>Index</b>	<b>25</b>

# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">ArduinoSMBus</a>	.....	5
<a href="#">BatteryMode</a>		
A struct to hold various battery mode flags	.....	15



## Chapter 2

# File Index

### 2.1 File List

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

C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/src/ <a href="#">ArduinoSMBus.cpp</a>	
Function definitions for the <a href="#">ArduinoSMBus</a> class . . . . .	19
C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/src/ <a href="#">ArduinoSMBus.h</a>	
Function declarations for the <a href="#">ArduinoSMBus</a> class . . . . .	20
C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/src/ <a href="#">main.cpp</a>	
Example arduino code to read battery data from an SMBus battery and print to serial output . .	22





# 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.

- 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](#) ()

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 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.

## 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

<i>batteryAddress</i>	
-----------------------	--

## 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()

```
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

### 3.1.2.7 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 isCharging()

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

Check if the battery is charging.

**Returns**

bool

### 3.1.2.17 isFullyCharged()

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

Check if the battery is fully charged.

Returns

bool

### 3.1.2.18 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

### 3.1.2.19 manufacturerName()

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

Get the Manufacturer Name from the battery.

Returns

const char\*

### 3.1.2.20 manufactureYear()

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

Get the manufacture year from the manufacture date.

Returns

int

### 3.1.2.21 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

### 3.1.2.22 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

### 3.1.2.23 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

### 3.1.2.24 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

### 3.1.2.25 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



#### 3.1.2.26 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

#### 3.1.2.27 serialNumber()

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

Get the Serial Number from the battery.

##### Returns

uint16\_t

#### 3.1.2.28 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

<i>batteryAddress</i>	
-----------------------	--

#### 3.1.2.29 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

### 3.1.2.30 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

### 3.1.2.31 temperature()

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

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

#### Returns

uint16\_t

### 3.1.2.32 temperatureC()

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

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

#### Returns

uint16\_t

### 3.1.2.33 temperatureF()

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

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

#### Returns

uint16\_t

### 3.1.2.34 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](#)

## 3.2 BatteryMode Struct Reference

A struct to hold various battery mode flags.

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

### Public Attributes

- bool [internal\\_charge\\_controller](#)  
*Indicates if the internal charge controller is supported.*
- bool [primary\\_battery\\_support](#)  
*Indicates if the primary battery support is supported.*
- bool [condition\\_flag](#)  
*Indicates the condition flag.*
- bool [charge\\_controller\\_enabled](#)  
*Indicates if the charge controller is enabled.*
- bool [primary\\_battery](#)  
*Indicates if the primary battery is enabled.*
- bool [alarm\\_mode](#)  
*Indicates the alarm mode.*
- bool [charger\\_mode](#)  
*Indicates the charger mode.*
- bool [capacity\\_mode](#)  
*Indicates the capacity mode.*

### 3.2.1 Detailed Description

A struct to hold various battery mode flags.

This struct holds various flags that represent the battery mode.

### 3.2.2 Member Data Documentation

#### 3.2.2.1 alarm\_mode

```
bool BatteryMode::alarm_mode
```

Indicates the alarm mode.

True - enable alarmWarning broadcasts to host. False - disable alarmWarning broadcasts to host.

#### 3.2.2.2 capacity\_mode

```
bool BatteryMode::capacity_mode
```

Indicates the capacity mode.

True - report in mA or mAh. False - report in 10mW or 10mWh.

#### 3.2.2.3 charge\_controller\_enabled

```
bool BatteryMode::charge_controller_enabled
```

Indicates if the charge controller is enabled.

This flag is true if the charge controller is enabled, and false otherwise.

#### 3.2.2.4 charger\_mode

```
bool BatteryMode::charger_mode
```

Indicates the charger mode.

True - enable chargingCurrent and chargingVoltage broadcasts to host. False - disable chargingCurrent and chargingVoltage broadcasts to host.

#### 3.2.2.5 condition\_flag

```
bool BatteryMode::condition_flag
```

Indicates the 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
```

Indicates if the internal charge controller is supported.

This flag is true if the internal charge controller is supported, and false otherwise.

### 3.2.2.7 primary\_battery

```
bool BatteryMode::primary_battery
```

Indicates if the primary battery is enabled.

This flag is true if the primary battery is enabled, and false otherwise.

### 3.2.2.8 primary\_battery\_support

```
bool BatteryMode::primary_battery_support
```

Indicates if the primary battery support is supported.

This flag is true if the primary battery support is supported, and false otherwise.

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

- C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/src/[ArduinoSMBus.h](#)



# 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](mailto: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

- struct [BatteryMode](#)  
*A struct to hold various battery mode 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 ( [cleee@unitedconsulting.com](mailto:cleee@unitedconsulting.com) )

#### Version

1.0

#### Date

2024-02-29

#### Copyright

Copyright (c) 2024

## 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
00054 struct BatteryMode {
00060     bool internal_charge_controller;
00061
00067     bool primary_battery_support;
00068
00074     bool condition_flag;
00075
00081     bool charge_controller_enabled;
```

```

00082
00088     bool primary_battery;
00089
00096     bool alarm_mode;
00097
00104     bool charger_mode;
00105
00112     bool capacity_mode;
00113 };
00114
00115 class ArduinoSMBus {
00116 public:
00117
00118     BatteryMode battery_mode;
00119
00120     ArduinoSMBus(uint8_t batteryAddress);
00121     void setBatteryAddress(uint8_t batteryAddress);
00122
00123     uint16_t remainingCapacityAlarm();
00124     uint16_t remainingTimeAlarm();
00125     BatteryMode batteryMode();
00126     uint16_t temperature();
00127     uint16_t temperatureC();
00128     uint16_t temperatureF();
00129     uint16_t voltage();
00130     uint16_t current();
00131     uint16_t averageCurrent();
00132     uint16_t maxError();
00133     uint16_t relativeStateOfCharge();
00134     uint16_t absoluteStateOfCharge();
00135     uint16_t remainingCapacity();
00136     uint16_t fullCapacity();
00137     uint16_t runTimeToEmpty();
00138     uint16_t avgTimeToEmpty();
00139     uint16_t avgTimeToFull();
00140     uint16_t batteryStatus();
00141     uint16_t chargingCurrent();
00142     uint16_t chargingVoltage();
00143     bool statusOK();
00144     bool isCharging();
00145     bool isFullyCharged();
00146     uint16_t cycleCount();
00147     uint16_t designCapacity();
00148     uint16_t designVoltage();
00149     uint16_t manufactureDate();
00150     int manufactureYear();
00151     uint16_t serialNumber();
00152     const char* manufacturerName();
00153     const char* deviceName();
00154     const char* deviceChemistry();
00155     uint16_t stateOfHealth();
00156
00157 private:
00158     uint8_t _batteryAddress;
00159     uint16_t readRegister(uint8_t reg);
00160     void readBlock(uint8_t reg, uint8_t* data, uint8_t len);
00161 };
00162
00163
00164 #endif

```

## 4.4 C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 - ArduinoSMBus/ArduinoSMBus/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](mailto:clee@unitedconsulting.com) )

#### Version

1.0

#### Date

2024-02-29

#### Copyright

Copyright (c) 2024



# Index

- absoluteStateOfCharge
  - ArduinoSMBus, [7](#)
- alarm\_mode
  - BatteryMode, [16](#)
- ArduinoSMBus, [5](#)
  - absoluteStateOfCharge, [7](#)
  - ArduinoSMBus, [7](#)
  - averageCurrent, [7](#)
  - avgTimeToEmpty, [7](#)
  - avgTimeToFull, [7](#)
  - batteryMode, [8](#)
  - batteryStatus, [8](#)
  - chargingCurrent, [8](#)
  - chargingVoltage, [8](#)
  - current, [9](#)
  - cycleCount, [9](#)
  - designCapacity, [9](#)
  - designVoltage, [9](#)
  - deviceChemistry, [10](#)
  - deviceName, [10](#)
  - fullCapacity, [10](#)
  - isCharging, [10](#)
  - isFullyCharged, [10](#)
  - manufactureDate, [11](#)
  - manufacturerName, [11](#)
  - manufactureYear, [11](#)
  - maxError, [11](#)
  - relativeStateOfCharge, [11](#)
  - remainingCapacity, [12](#)
  - remainingCapacityAlarm, [12](#)
  - remainingTimeAlarm, [12](#)
  - runTimeToEmpty, [12](#)
  - serialNumber, [13](#)
  - setBatteryAddress, [13](#)
  - stateOfHealth, [13](#)
  - statusOK, [13](#)
  - temperature, [14](#)
  - temperatureC, [14](#)
  - temperatureF, [14](#)
  - voltage, [14](#)
- averageCurrent
  - ArduinoSMBus, [7](#)
- avgTimeToEmpty
  - ArduinoSMBus, [7](#)
- avgTimeToFull
  - ArduinoSMBus, [7](#)
- BatteryMode, [15](#)
  - alarm\_mode, [16](#)
  - capacity\_mode, [16](#)
  - charge\_controller\_enabled, [16](#)
  - charger\_mode, [16](#)
  - condition\_flag, [16](#)
  - internal\_charge\_controller, [16](#)
  - primary\_battery, [16](#)
  - primary\_battery\_support, [17](#)
- batteryMode
  - ArduinoSMBus, [8](#)
- batteryStatus
  - ArduinoSMBus, [8](#)
- C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 -  
ArduinoSMBus/ArduinoSMBus/src/ArduinoSMBus.cpp, [19](#)
- C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 -  
ArduinoSMBus/ArduinoSMBus/src/ArduinoSMBus.h, [20, 21](#)
- C:/Users/Chris Lee/Sync/Personal Projects/p2024-005 -  
ArduinoSMBus/ArduinoSMBus/src/main.cpp, [22](#)
- capacity\_mode
  - BatteryMode, [16](#)
- charge\_controller\_enabled
  - BatteryMode, [16](#)
- charger\_mode
  - BatteryMode, [16](#)
- chargingCurrent
  - ArduinoSMBus, [8](#)
- chargingVoltage
  - ArduinoSMBus, [8](#)
- condition\_flag
  - BatteryMode, [16](#)
- current
  - ArduinoSMBus, [9](#)
- cycleCount
  - ArduinoSMBus, [9](#)
- designCapacity
  - ArduinoSMBus, [9](#)
- designVoltage
  - ArduinoSMBus, [9](#)
- deviceChemistry
  - ArduinoSMBus, [10](#)
- deviceName
  - ArduinoSMBus, [10](#)
- fullCapacity
  - ArduinoSMBus, [10](#)
- internal\_charge\_controller

- BatteryMode, [16](#)
- isCharging
  - ArduinoSMBus, [10](#)
- isFullyCharged
  - ArduinoSMBus, [10](#)
- manufactureDate
  - ArduinoSMBus, [11](#)
- manufacturerName
  - ArduinoSMBus, [11](#)
- manufactureYear
  - ArduinoSMBus, [11](#)
- maxError
  - ArduinoSMBus, [11](#)
- primary\_battery
  - BatteryMode, [16](#)
- primary\_battery\_support
  - BatteryMode, [17](#)
- relativeStateOfCharge
  - ArduinoSMBus, [11](#)
- remainingCapacity
  - ArduinoSMBus, [12](#)
- remainingCapacityAlarm
  - ArduinoSMBus, [12](#)
- remainingTimeAlarm
  - ArduinoSMBus, [12](#)
- runTimeToEmpty
  - ArduinoSMBus, [12](#)
- serialNumber
  - ArduinoSMBus, [13](#)
- setBatteryAddress
  - ArduinoSMBus, [13](#)
- stateOfHealth
  - ArduinoSMBus, [13](#)
- statusOK
  - ArduinoSMBus, [13](#)
- temperature
  - ArduinoSMBus, [14](#)
- temperatureC
  - ArduinoSMBus, [14](#)
- temperatureF
  - ArduinoSMBus, [14](#)
- voltage
  - ArduinoSMBus, [14](#)