

# Infineon Arduino Library Documentation

**Author:** Infineon Technologies AG

**Date:** August 13, 2018

# Contents

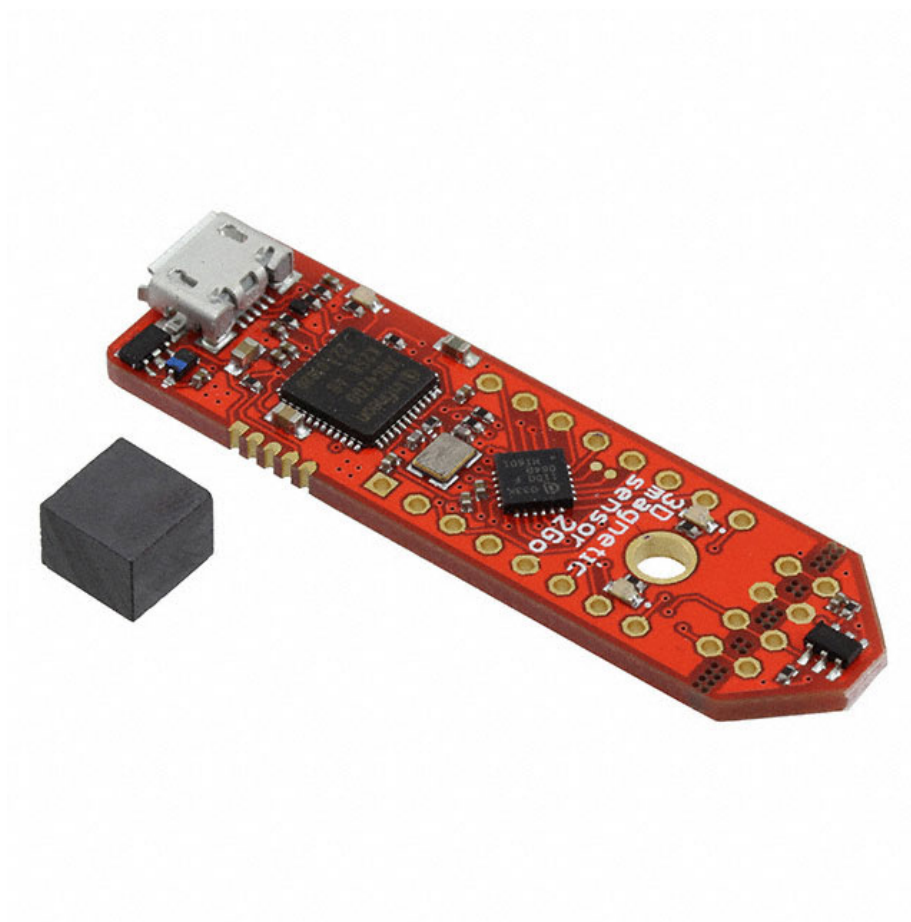
<b>1</b>	<b>Arduino Library for Infineon's Magnetic 3D Sensor TLE493D-W2B6</b>	<b>4</b>
1.1	Introduction	4
1.2	Main Features	4
1.2.1	Interrupts	4
1.2.2	Collision Avoidance and Clock Stretching	5
1.2.3	Wake Up Mode	5
<b>2</b>	<b>Bug List</b>	<b>6</b>
<b>3</b>	<b>Hierarchical Index</b>	<b>7</b>
3.1	Class Hierarchy	7
<b>4</b>	<b>Class Index</b>	<b>8</b>
4.1	Class List	8
<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	Tle493d Class Reference	9
5.1.1	Member Enumeration Documentation	10
5.1.1.1	AccessMode_e	10
5.1.1.2	TypeAddress_e	10
5.1.2	Constructor & Destructor Documentation	11
5.1.2.1	Tle493d(AccessMode_e mode=MASTERCONTROLLEDMODE, TypeAddress_e productType=TLE493D_A0)	11
5.1.3	Member Function Documentation	11
5.1.3.1	begin(TwoWire &bus, TypeAddress_e slaveAddress, bool reset, uint8_t oneByteRead)	11
5.1.3.2	getAzimuth(void)	11
5.1.3.3	getNorm(void)	11
5.1.3.4	getPolar(void)	11
5.1.3.5	getRegBits(uint8_t regMaskIndex)	11
5.1.3.6	getTemp(void)	12
5.1.3.7	getX(void)	12
5.1.3.8	getY(void)	12
5.1.3.9	getZ(void)	12
5.1.3.10	resetSensor(void)	12
5.1.3.11	setAccessMode(AccessMode_e mode)	12
5.1.3.12	setRegBits(uint8_t regMaskIndex, uint8_t data)	13
5.2	Tle493d_a2b6 Class Reference	13
5.3	Tle493d_w2b6 Class Reference	14
5.3.1	Member Function Documentation	15
5.3.1.1	setUpdateRate(uint8_t updateRate)	15
5.3.1.2	setWakeUpThreshold(float xh, float xl, float yh, float yl, float zh, float zl)	15



# 1 Arduino Library for Infineon's Magnetic 3D Sensor TLE493D-W2B6

## 1.1 Introduction

The TLE493D-W2B6 sensor family comes with I2C interface and wake-up function. This sensor family TLE493D offers accurate three dimensional sensing with extremely low-power consumption.



## 1.2 Main Features

### 1.2.1 Interrupts

Interrupts can be sent from the sensor to the microcontroller to notify the completion of a measurement and its ADC conversion. Values read directly after interrupts are guaranteed to be consistent. Without interrupts values read might

be stale.

### 1.2.2 Collision Avoidance and Clock Stretching

In case of a collision, the sensor interrupt disturbs the I2C clock, causing an additional SCL pulse which shifts the data read out by one bit. If collision avoidance is activated, the sensor monitors the start/stop conditions, and suppresses interrupts in between.

When interrupts are disabled, this feature becomes clock stretching, that is, the data read out only starts after the ADC conversion is finished. Thus it can be avoided that during an ADC conversion old or corrupted measurement results are read out, which may occur when the ADC is writing to a register while this is being read out by the microcontroller. When clock stretching is enabled, the sensor pulls the SCL line down during ongoing ADC conversions, reading of sensor registers or the transmission of valid ACKs.

Two register bits (CA and INT) work together for different configurations. APIs to modify these two bits are not offered since they have to be set according to different operating modes (master controlled mode, low power mode and fast mode) for the sensor to work.

### 1.2.3 Wake Up Mode

Wake up mode is intended to be used with low power mode or fast mode. This mode disables interrupts within a user-specified range, so that interrupts are generated only when relevant data are available.

## 2 Bug List

### File [Tle493d.cpp](#)

sensor not responding after powered off, needs to be reflashed to respond

reset freezes the sensor, fast mode not working

User manual recommends INT=0 in fast mode however only disabling INT works

wake up mode not configured correctly (WA bit = 0)

### Member [Tle493d::resetSensor](#) (void)

not working

## 3 Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Tle493d . . . . .	9
Tle493d_a2b6 . . . . .	13
Tle493d_w2b6 . . . . .	14

## 4 Class Index

### 4.1 Class List

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

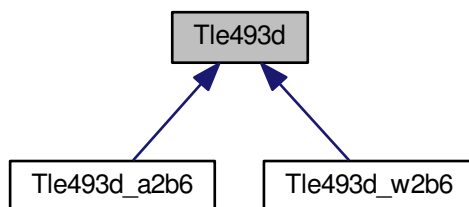
<a href="#">Tle493d</a>	9
<a href="#">Tle493d_a2b6</a>	13
<a href="#">Tle493d_w2b6</a>	14



## 5 Class Documentation

### 5.1 Tle493d Class Reference

Inheritance diagram for Tle493d:



#### Public Types

- enum [TypeAddress\\_e](#) { **TLE493D\_A0** = 0x35, **TLE493D\_A1** = 0x22, **TLE493D\_A2** = 0x78, **TLE493D\_A3** = 0x44 }
- enum [AccessMode\\_e](#) { **LOWPOWERMODE** = 0, **MASTERCONTROLLEDMODE** = 1, **FASTMODE** = 3 }

#### Public Member Functions

- bool [setAccessMode](#) ([AccessMode\\_e](#) mode)  
*Sets the operating mode of the sensor.*
- [Tle493d](#) ([AccessMode\\_e](#) mode=MASTERCONTROLLEDMODE, [TypeAddress\\_e](#) productType=TLE493D\_A0)  
*Constructor of the sensor class.*
- [~Tle493d](#) (void)  
*Destructor.*
- void [begin](#) (void)  
*Starts the sensor.*
- void [begin](#) (TwoWire &bus, [TypeAddress\\_e](#) slaveAddress, bool reset, uint8\_t oneByteRead)  
*Starts the sensor.*
- void [enableTemp](#) (void)

*Enables temperature measurement; by default already enabled.*

- void `disableTemp` (void)

*Disables temperature measurement to reduce power consumption.*

- Tle493d\_Error `updateData` (void)

*Reads measurement results from sensor.*

- float `getX` (void)
- float `getY` (void)
- float `getZ` (void)
- float `getNorm` (void)
- float `getAzimuth` (void)
- float `getPolar` (void)
- float `getTemp` (void)
- void `resetSensor` (void)

*Resets the sensor.*

- void `readDiagnosis` (uint8\_t(&diag)[7])

## Protected Member Functions

- void `setRegBits` (uint8\_t regMaskIndex, uint8\_t data)

*Stores new values into the bus interface; for this function to take effect the function `writeOut()` should be called afterwards.*

- uint8\_t `getRegBits` (uint8\_t regMaskIndex)

*Returns the value of a register field.*

## Protected Attributes

- tle493d::BusInterface\_t **`mInterface`**

## 5.1.1 Member Enumeration Documentation

### 5.1.1.1 enum Tle493d::AccessMode\_e

Enumerates the three available modes; number 2 is reserved In low power mode cyclic measurements and AD↔C-conversions are carried out with a update rate; the wake-up function is already configured for this mode, so that the sensor can continue making magnetic field measurements. With this configuration the microcontroller will only consume power and access the sensor if relevant measurement data is available. In master controlled mode the sensor powered down if when it is not triggered. This library configures to ADC start before sending first MSB of data registers In fast mode the measurements and ADC-conversions are running continuously.

### 5.1.1.2 enum Tle493d::TypeAddress\_e

Defines four types of the sensor family which are supported by this library and their corresponding addresses The addresses can be concatenated with 0 or 1 for reading or writing

## 5.1.2 Constructor & Destructor Documentation

### 5.1.2.1 Tle493d::Tle493d ( AccessMode\_e mode = MASTERCONTROLLEDMODE, TypeAddress\_e productType = TLE493D\_A0 )

Constructor of the sensor class.

#### Parameters

<i>mode</i>	Operating mode of the sensor; default is the master controlled mode
<i>productType</i>	The library supports product types from A0 to A3; default is type A0

## 5.1.3 Member Function Documentation

### 5.1.3.1 void Tle493d::begin ( TwoWire & bus, TypeAddress\_e slaveAddress, bool reset, uint8\_t oneByteRead )

Starts the sensor.

#### Parameters

<i>bus</i>	The I2C bus
<i>slaveAddress</i>	The 7-bit slave address as defined in Tle493d_Type
<i>reset</i>	If a reset should be initiated before starting the sensor
<i>oneByteRead</i>	If one-byte read protocol should be used. Otherwise the two-byte protocol is available

### 5.1.3.2 float Tle493d::getAzimuth ( void )

#### Returns

the Azimuth angle  $\arctan(y/x)$

### 5.1.3.3 float Tle493d::getNorm ( void )

#### Returns

norm of the magnetic field vector  $\sqrt{x^2 + y^2 + z^2}$

### 5.1.3.4 float Tle493d::getPolar ( void )

#### Returns

the angle in polar coordinates  $\arctan(z/(\sqrt{x^2+y^2}))$

### 5.1.3.5 uint8\_t Tle493d::getRegBits ( uint8\_t regMaskIndex ) [protected]

Returns the value of a register field.

#### Parameters

<i>Register</i>	mask index as defined in Registers_e
-----------------	--------------------------------------

#### 5.1.3.6 float Tle493d::getTemp ( void )

##### Returns

the temperature value

#### 5.1.3.7 float Tle493d::getX ( void )

##### Returns

the Cartesian x-coordinate

#### 5.1.3.8 float Tle493d::getY ( void )

##### Returns

the Cartesian y-coordinate

#### 5.1.3.9 float Tle493d::getZ ( void )

##### Returns

the Cartesian z-coordinate

#### 5.1.3.10 void Tle493d::resetSensor ( void )

Resets the sensor.

**Bug** not working

#### 5.1.3.11 bool Tle493d::setAccessMode ( AccessMode\_e mode )

Sets the operating mode of the sensor.

#### Parameters

<i>mode</i>	MASTERCONTROLLEDMODE, LOWPOWERMODE or FASTMODE, default is MASTERCONTROLLEDMODE
-------------	---

### 5.1.3.12 void Tle493d::setRegBits ( uint8\_t regMaskIndex, uint8\_t data ) [protected]

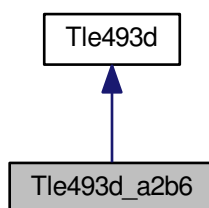
Stores new values into the bus interface; for this function to take effect the function writeOut() should be called afterwards.

#### Parameters

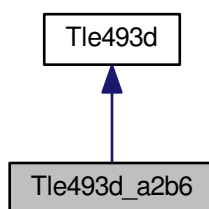
<i>Register</i>	mask index as defined in Registers_e
<i>Value</i>	to be written into the register field specified by the register index

## 5.2 Tle493d\_a2b6 Class Reference

Inheritance diagram for Tle493d\_a2b6:



Collaboration diagram for Tle493d\_a2b6:



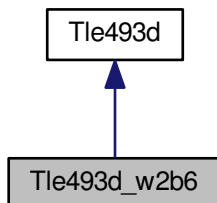
#### Public Member Functions

- void **setUpdateRate** (uint8\_t updateRate)

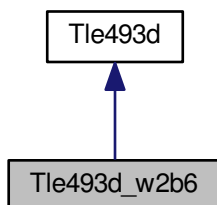
## Additional Inherited Members

### 5.3 Tle493d\_w2b6 Class Reference

Inheritance diagram for Tle493d\_w2b6:



Collaboration diagram for Tle493d\_w2b6:



## Public Member Functions

- void [setWakeUpThreshold](#) (float xh, float xl, float yh, float yl, float zh, float zl)

*The Wake Up threshold range disabling /INT pulses between upper threshold and lower threshold is limited to a window of the half output range. Here the adjustable range can be set with a ratio of size [-1,1] When all the measurement values Bx, By and Bz are within this range INT is disabled. If the arguments are out of range or any upper threshold is smaller than the lower one, the function returns without taking effect.*

- bool [wakeUpEnabled](#) (void)

*Checks if WA bit is set. When not interrupt configuration is as specified by the CA and INT bits.*

- void [setUpdateRate](#) (uint8\_t updateRate)

*Sets the update rate in low power mode.*

## Additional Inherited Members

### 5.3.1 Member Function Documentation

#### 5.3.1.1 void Tle493d\_w2b6::setUpdateRate ( uint8\_t *updateRate* )

Sets the update rate in low power mode.

##### Parameters

<i>updateRate</i>	Update rate which is an unsigned integer from the 0 (the fastest) to 7 (the highest)
-------------------	--

#### 5.3.1.2 void Tle493d\_w2b6::setWakeUpThreshold ( float *xh*, float *xl*, float *yh*, float *yl*, float *zh*, float *zl* )

The Wake Up threshold range disabling /INT pulses between upper threshold and lower threshold is limited to a window of the half output range. Here the adjustable range can be set with a ratio of size [-1,1] When all the measurement values Bx, By and Bz are within this range INT is disabled. If the arguments are out of range or any upper threshold is smaller than the lower one, the function returns without taking effect.

##### Parameters

<i>xh</i>	Upper threshold in x direction
<i>xl</i>	Lower threshold in x direction
<i>yh</i>	Upper threshold in y direction
<i>yl</i>	Lower threshold in y direction
<i>zh</i>	Upper threshold in z direction
<i>zl</i>	Lower threshold in z direction

## Index

AccessMode\_e  
  Tle493d, [10](#)

begin  
  Tle493d, [11](#)

getAzimuth  
  Tle493d, [11](#)

getNorm  
  Tle493d, [11](#)

getPolar  
  Tle493d, [11](#)

getRegBits  
  Tle493d, [11](#)

getTemp  
  Tle493d, [12](#)

getX  
  Tle493d, [12](#)

getY  
  Tle493d, [12](#)

getZ  
  Tle493d, [12](#)

resetSensor  
  Tle493d, [12](#)

setAccessMode  
  Tle493d, [12](#)

setRegBits  
  Tle493d, [12](#)

setUpdateRate  
  Tle493d\_w2b6, [15](#)

setWakeUpThreshold  
  Tle493d\_w2b6, [15](#)

Tle493d, [9](#)  
  AccessMode\_e, [10](#)  
  begin, [11](#)  
  getAzimuth, [11](#)  
  getNorm, [11](#)  
  getPolar, [11](#)  
  getRegBits, [11](#)  
  getTemp, [12](#)  
  getX, [12](#)  
  getY, [12](#)  
  getZ, [12](#)  
  resetSensor, [12](#)  
  setAccessMode, [12](#)  
  setRegBits, [12](#)  
  Tle493d, [11](#)  
  TypeAddress\_e, [10](#)  
  Tle493d\_a2b6, [13](#)  
  Tle493d\_w2b6, [14](#)  
  setUpdateRate, [15](#)  
  setWakeUpThreshold, [15](#)  
  TypeAddress\_e  
    Tle493d, [10](#)



#### Trademarks of Infineon Technologies AG

μHVIC™, μIPM™, μPFC™, AU-ConvertIR™, AURIX™, C166™, CanPAK™, CIPOS™, CIPURSE™, CoolDP™, CoolGaN™, COOLiR™, CoolMOS™, CoolSET™, CoolSiC™, DAVE™, DI-POL™, DirectFET™, DrBlade™, EasyPIM™, EconoBRIDGE™, EconoDUAL™, EconoPACK™, EconoPIM™, EiceDRIVER™, eupec™, FCOS™, GaNpowIR™, HEXFET™, HITFET™, HybridPACK™, iMOTION™, IRAM™, ISOFACE™, IsoPACK™, LEDriviR™, LITIX™, MIPAQ™, ModSTACK™, my-d™, NovalithiC™, OPTIGA™, OptiMOS™, ORIGA™, PowIRaudio™, PowIRstage™, PrimePACK™, PrimeSTACK™, PROFET™, PRO-SiL™, RASIC™, REAL3™, SmartLEWIS™, SOLID FLASH™, SPOC™, StrongIRFET™, SuplIRBuck™, TEMPFET™, TRENCHSTOP™, TriCore™, UHVIC™, XHP™, XMC™.

Trademarks Update 2015-12-22

#### Other Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

#### IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

#### WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury