

Component manufacturer

Specification document of LM45B, LM45C

Model number	LM45B, LM45C					
Datasheets	LM45	SOT-23	Precision	Centigrade	Temperature	Sensors

datasheet (Rev. C)

Texas Instruments

Specification Ver 01.00.00 Nov 1,2022 New release Documentation provided Rui Long Lab Inc. https://rui-long-lab.com/

1.	Component datasheet	2
	·	
2.	Component Software IF specification	3
3	File Structure and Definitions	5

License

Open Source Software for Embedded Components ("OSS-EC") is open source software files and related documentation files for component products used in computer systems and other applications. OSS-EC is provided to those who accept the OSS-EC Terms of Use for the OSS-EC site; see https://oss-ec.com/license_agreement/ for the OSS-EC Terms of Use. By downloading the OSS-EC from the OSS-EC site or obtaining the OSS-EC by any means, you accept the Terms of Use. Please read and accept the Terms of Use before using the OSS-EC. If you do not agree to the Terms of Use, please do not use the OSS-EC.



1. Component datasheet

Temperature accuracy	LM45B	±2.0 $^{\circ}$	C	Accuracy $T_A = 25^{\circ}$ C
		$\pm3.0^{\circ}$	С	Accuracy $T_A = T_{\text{MAX}}(100^{\circ} \text{ C})$
		$\pm3.0^{\circ}$	С	Accuracy $T_A = T_{\text{min}}(-20^{\circ} \text{ C})$
	LM45C	±3.0 $^{\circ}$	C	Accuracy $T_A = 25^{\circ}$ C
		±4.0 $^{\circ}$	C	Accuracy $T_A = T_{\text{MAX}}(100^{\circ} \text{ C})$
		±4.0 $^{\circ}$	C	Accuracy $T_A = T_{\text{min}}(-20^{\circ} \text{ C})$

Temperature range $-20 \text{ to } +100^{\circ} \text{ C}$

CAUTION:-20[° C], the circuit needs a voltage Offset

Range of power supply voltage (Vdd) 4.0 to 10.0[V]

Output voltage (Vout) Linear $10 \text{ [mV/}^{\circ} \text{ C] Typ. (-20 to } +100^{\circ} \text{ C)}$

Calculation $Vout = 0.01 \text{ V/}^{\circ} \text{ C} \times \text{Ta}$

 $Ta = Vout / (0.01 \text{ V/}^{\circ} \text{ C})$

Vdd vs Vout Non-link

Applications IoT etc

· Battery Management

· FAX Machines

· Printers

· Portable Medical Instruments

• HVAC

· Power Supply Modules

· Disk Drivers

· Computers

Automotive



2. Component Software IF specification

The software interface specifications based on the LM45B, LM45C component specifications are as follows.

The voltage value-to-physical value conversion equation is a linear conversion equation as shown in the equation below.

ADC value to voltage value conversion formula

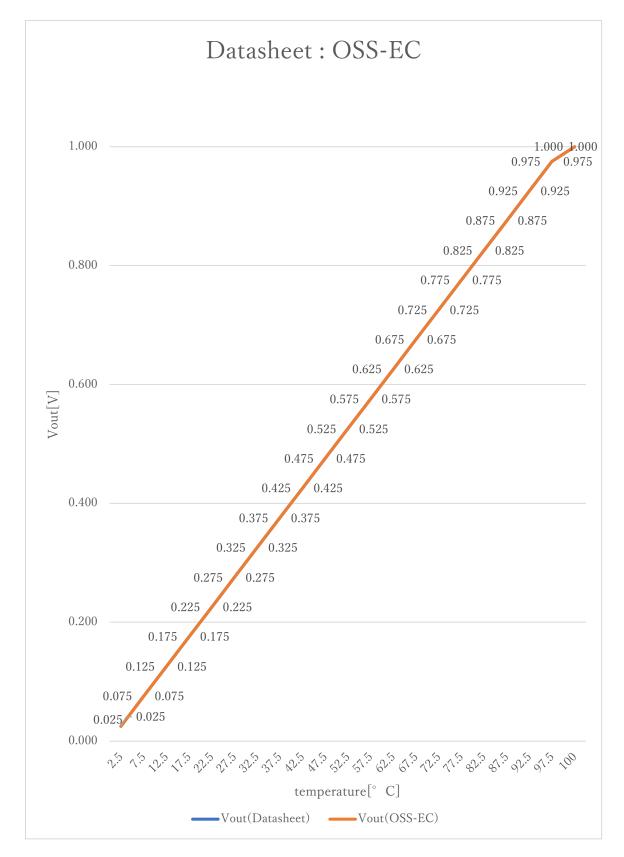
$$vi = (ai \times iADC_vdd) / 2^{iADC_bit}$$
 [V]

Voltage value to physical value conversion formula

y = (vi - iLM45B_xoff) / iLM45B_gain + iLM45B_yoff [°C] iLM45B_min
$$\leq$$
 y \leq iLM45B_max

```
A/D conversion value
ai
٧i
                  Sensor output voltage value [V]
i ADC vdd
                  Sensor supply voltage value [V]
iADC_bit
                  A/D conversion bit length
                  Temperature value [°C]
                                                     // X offset [V]
#define iLM45B_xoff
                                    0. OF
#define iLM45B_yoff
                                    <u>0. 0F</u>
                                                     // Y offset [°C]
                                                     // Gain [V/°C]
#define iLM45B_gain
                                    0. 01F
#define iLM45B_max
                                                     // Temperature Max [°C]
                                    <u>100. 0F</u>
#define iLM45B_min
                                    2. 5F
                                                     // Temperature Min [°C]
                                                     // CAUTION:-20[° C], the circuit
                                                          needs a voltage Offset
```





 $Vout(Datasheet) = 10 \text{ mV/}^{\circ} \text{ C} \times \text{ T}^{\circ} \text{ C}$



3. File Structure and Definitions

extern const tbl_adc_t tbl_LM45B;

LM45B.h

```
#include "user_define.h"
// Components number
#define iLM45B
                             129U
                                                          // Texas Instruments LM45B, LM45C
// LM45B, LM45C System Parts definitions
#define iLM45B_xoff
                             <u>0. 0F</u>
                                                          // X offset [V]
#define iLM45B_yoff
                             <u>0. 0F</u>
                                                          // Y offset [°C]
                             0.01F
                                                          // Gain [V/°C]
#define iLM45B_gain
#define iLM45B_max
                             100. OF
                                                          // Temperature Max [°C]
#define iLM45B_min
                             2. 5F
                                                          // Temperature Min [°C]
                                                          // CAUTION:-20[° C], the circuit
                                                             needs a voltage Offset
```



LM45B.cpp

```
#include
                "LM45B. h"
#if
        iLM45B_ma == iSMA
                                                        // Simple moving average filter
static float32 LM45B_sma_buf[iLM45B_SMA_num];
static const sma_f32_t LM45B_Phy_SMA =
        iInitial ,
                                                        // Initial state
        iLM45B_SMA_num ,
                                                       // Simple moving average number & buf size
        OU ,
                                                         // buffer position
        0.0F,
                                                        // sum
        &LM45B_sma_buf[0]
                                                        // buffer
};
#elif
        iLM45B_ma == iEMA
                                                         // Exponential moving average filter
static const ema_f32_t LM45B_Phy_EMA =
{
        iInitial ,
                                                        // Initial state
        0.0F,
                                                         // Xn-1
        iLM45B_EMA_K
                                                        // Exponential smoothing factor
};
#elif
        iLM45B_ma == iWMA
                                                        // Weighted moving average filter
static float32 LM45B_wma_buf[iLM45B_WMA_num];
static const wma_f32_t LM45B_Phy_WMA =
{
        iInitial ,
                                                         // Initial state
        iLM45B_WMA_num ,
                                               // Weighted moving average number & buf size
                                                        // buffer poition
        OU ,
                                                        // kn sum
        iLM45B_WMA_num * (iLM45B_WMA_num + 1)/2,
        &LM45B_wma_buf[0]
                                                        // Xn buffer
};
#else
                                                         // Non-moving average filter
#endif
#define iDummy_adr
                         0xffffffff
                                                        // Dummy address
```



```
const tbl_adc_t tbl_LM45B =
        iLM45B
        iLM45B_pin
        iLM45B_xoff
        iLM45B_yoff
        iLM45B_gain
        iLM45B_max
        iLM45B_min
        iLM45B_ma
#if
        iLM45B_ma == iSMA
                                                          // Simple moving average filter
        &LM45B_Phy_SMA
        (ema_f32_t*) iDummy_adr,
        (wma_f32_t*) iDummy_adr
#elif
        iLM45B_ma == iEMA
                                                          // Exponential moving average filter
        (sma_f32_t*) iDummy_adr,
        &LM45B_Phy_EMA
        (wma_f32_t*) iDummy_adr
#elif
        iLM45B_ma == iWMA
                                                          // Weighted moving average filter
        (sma_f32_t*) iDummy_adr ,
        (ema_f32_t*) iDummy_adr ,
        &LM45B_Phy_WMA
#else
                                                          // Non-moving average filter
         (sma_f32_t*)iDummy_adr ,
         (ema_f32_t*) iDummy_adr ,
        (wma_f32_t*) iDummy_adr
#endif
};
```