Specification document of LM45B, LM45C

Component manufacturer	Texas Instruments
Model number	LM45B, LM45C
Datasheets	LM45 SOT-23 Precision Centigrade Temperature Sensors datasheet (Rev. C)
Specification Ver	01.00.00 Nov 1,2022 New release
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/

1. Component datasheet

Temperature accuracy	LM45B	±2.0°C	Accuracy T_A = 25°C
		±3.0°C	Accuracy T_A = T_{MAX}(100°C)
		±3.0°C	Accuracy T_A = T_{MIN}(-20°C)
	LM45C	±3.0°C	Accuracy T_A = 25°C
		±4.0°C	Accuracy T_A = T_{MAX}(100°C)
		±4.0°C	Accuracy T_A = T_{MIN}(-20°C)

Temperature range: -20 to +100°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 [mV/°C] Typ. (-20 to +100°C)

Calculation: Vout = 0.01 V/°C × Ta
Ta = Vout / (0.01 V/°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

$$vi = (ai \times iADC\_vdd) / 2^{iADC\_bit} \quad [V]$$

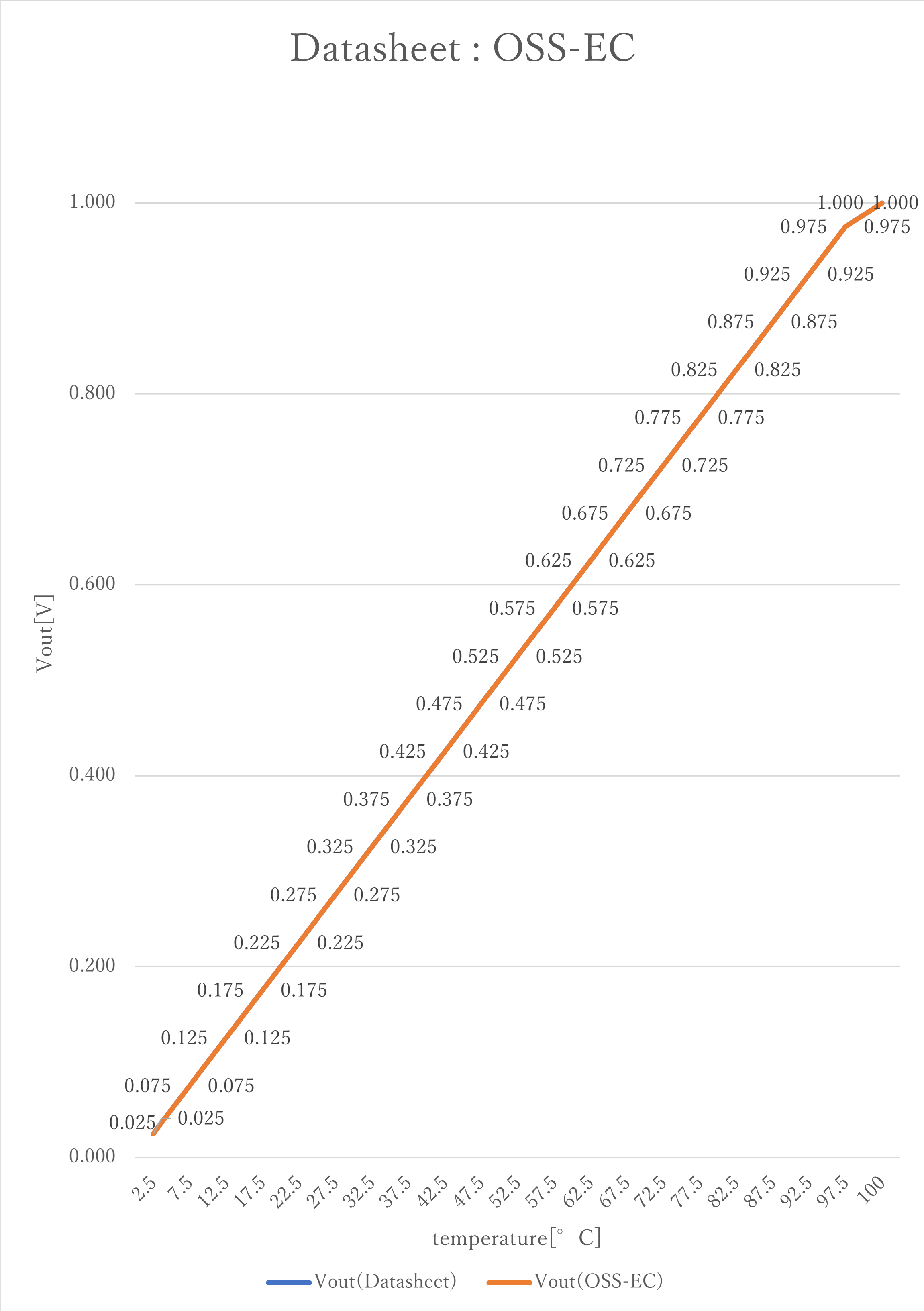
Voltage value to physical value conversion formula

$$y = (vi - iLM45B\_xoff) / iLM45B\_gain + iLM45B\_yoff \quad [℃]$$

$$iLM45B\_min \leqq y \leqq iLM45B\_max$$

ai	A/D conversion value
vi	Sensor output voltage value [V]
iADC_vdd	Sensor supply voltage value [V]
iADC_bit	A/D conversion bit length
y	Temperature value [℃]

```
#define iLM45B_xoff        0.0F        // X offset [V]
#define iLM45B_yoff        0.0F        // Y offset [℃]
#define iLM45B_gain        0.01F       // Gain [V/℃]
#define iLM45B_max         100.0F      // Temperature Max [℃]
#define iLM45B_min         2.5F        // Temperature Min [℃]
                                       // CAUTION:-20[° C],the circuit
                                          needs a voltage Offset
```



Vout(Datasheet) = 10 mV/° C × T ° C

3. File Structure and Definitions

LM45B.h

```
#include "user_define.h"

// Components number
#define iLM45B              129U                    // Texas Instruments LM45B, LM45C

// LM45B, LM45C System Parts definitions
#define iLM45B_xoff         0.0F                    // X offset [V]
#define iLM45B_yoff         0.0F                    // Y offset [℃]
#define iLM45B_gain         0.01F                   // Gain [V/℃]
#define iLM45B_max          100.0F                  // Temperature Max [℃]
#define iLM45B_min          2.5F                    // Temperature Min [℃]
                                                    // CAUTION:-20[° C], the circuit
                                                       needs a voltage Offset


extern const tbl_adc_t tbl_LM45B;
```

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
        0U ,                                         // 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
        0U ,                                         // buffer poition
        iLM45B_WMA_num * (iLM45B_WMA_num + 1)/2 ,    // kn sum
        &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

};
```