Specification document of LM50B

Component manufacturer	Texas Instruments
Model number	LM50B
Datasheets	LM50 and LM50-Q1 SOT-23 Single-Supply Centigrade Temperature Sensor datasheet (Rev. G)
Specification Ver	01.00.00 Nov 1,2022 New release
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/

License

1. Component datasheet

Temperature accuracy	±2.0 ° C	Accuracy T_A = 25° C
	±3.0 ° C	Accuracy T_A = T_{MAX}(100° C)
	-3.5 ° C∼ 3° C	Accuracy T_A = T_{MIN}(-25° C)
Temperature range	-25 to +100° C	
Range of power supply voltage (Vdd)	4.5 to 10.0[V]	
Output voltage (Vout)	Linear 10 [mV/° C] Typ. (-25 to +100° C)	
	0 [° C] 500 [mV]	
Calculation	Vout = 0.5V + (0.01 V/° C × Ta)	
	Ta = (Vout – 0.5V) / (0.01 V/° C)	
Vdd vs Vout	Non-link	
Applications	IoT etc	

- Computers
- Disk Drives
- Battery Management
- FAX Machines
- Printers
- Portable Medical Instruments
- HVAC
- Power Supply Modules

2. Component Software IF specification

The software interface specifications based on the LM50B 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

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

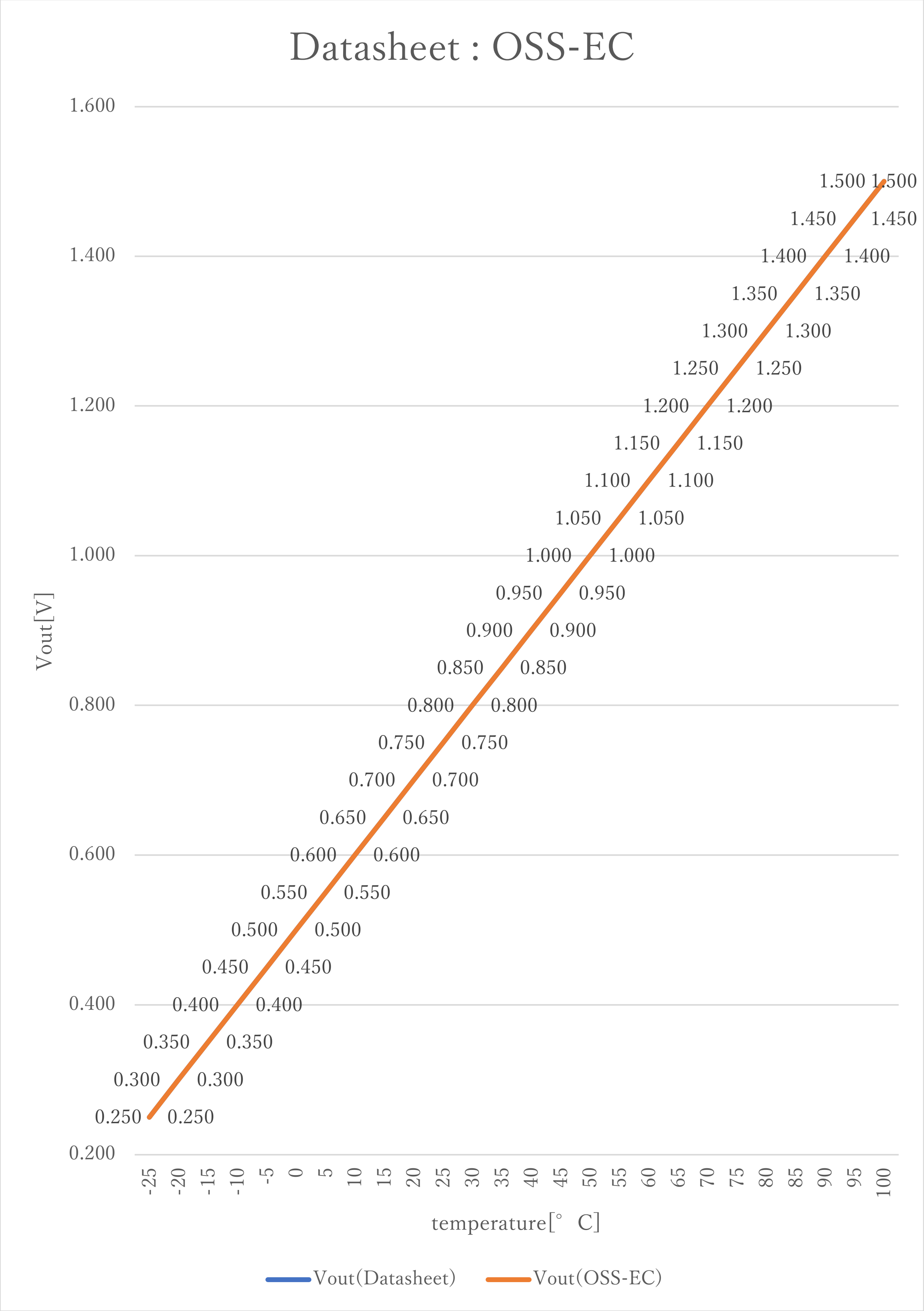
Voltage value to physical value conversion formula

$$y = (vi - iLM50B\_xoff) / iLM50B\_gain + iLM50B\_yoff \quad [℃]$$

$$iLM50B\_min \leqq y \leqq iLM50B\_max$$

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 iLM50B_xoff           0.5F           // X offset [V]
#define iLM50B_yoff           0.0F           // Y offset [℃]
#define iLM50B_gain           0.01F          // Gain [V/℃]
#define iLM50B_max            100.0F         // Temperature Max [℃]
#define iLM50B_min            -25.0F         // Temperature Min [℃]
```



Vout(Datasheet) = 10 mV/° C × T ° C + 500 mV

3. File Structure and Definitions

LM50B.h

```
#include "user_define.h"

// Components number
#define iLM50B               130U                          // Texas Instruments LM50B

// LM50B System Parts definitions
#define iLM50B_xoff          0.5F                          // X offset [V]
#define iLM50B_yoff          0.0F                          // Y offset [℃]
#define iLM50B_gain          0.01F                         // Gain [V/℃]
#define iLM50B_max           100.0F                        // Temperature Max [℃]
#define iLM50B_min           -25.0F                        // Temperature Min [℃]

extern const tbl_adc_t tbl_LM50B;
```

LM50B.cpp

```
#include        "LM50B.h"
#if     iLM50B_ma == iSMA                               // Simple moving average filter
static float32 LM50B_sma_buf[iLM50B_SMA_num];
static const sma_f32_t LM50B_Phy_SMA =
{
        iInitial ,                                      // Initial state
        iLM50B_SMA_num ,                              // Simple moving average number & buf size
        0U ,                                            // buffer position
        0.0F ,                                          // sum
        &LM50B_sma_buf[0]                               // buffer
};
#elif   iLM50B_ma == iEMA                               // Exponential moving average filter
static const ema_f32_t LM50B_Phy_EMA =
{
        iInitial ,                                      // Initial state
        0.0F ,                                          // Xn-1
        iLM50B_EMA_K                                    // Exponential smoothing factor
};
#elif   iLM50B_ma == iWMA                               // Weighted moving average filter
static float32 LM50B_wma_buf[iLM50B_WMA_num];
static const wma_f32_t LM50B_Phy_WMA =
{
        iInitial ,                                      // Initial state
        iLM50B_WMA_num ,                       // Weighted moving average number & buf size
        0U ,                                            // buffer poition
        iLM50B_WMA_num * (iLM50B_WMA_num + 1)/2 ,       // kn sum
        &LM50B_wma_buf[0]                               // Xn buffer
};
#else                                                   // Non-moving average filter
#endif

#define iDummy_adr       0xffffffff                     // Dummy address
```

```
const tbl_adc_t tbl_LM50B =
{
        iLM50B                 ,
        iLM50B_pin             ,
        iLM50B_xoff            ,
        iLM50B_yoff            ,
        iLM50B_gain            ,
        iLM50B_max             ,
        iLM50B_min             ,
        iLM50B_ma              ,

#if     iLM50B_ma == iSMA                                 // Simple moving average filter
        &LM50B_Phy_SMA         ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#elif   iLM50B_ma == iEMA                                 // Exponential moving average filter
        (sma_f32_t*)iDummy_adr ,
        &LM50B_Phy_EMA         ,
        (wma_f32_t*)iDummy_adr
#elif   iLM50B_ma == iWMA                                 // Weighted moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        &LM50B_Phy_WMA
#else                                                     // Non-moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#endif

};
```