Specification document of TMP9A00-EP

Component manufacturer	Texas Instruments
Model number	TMP9A00-EP
Datasheets	TMP9A00-EP ±2.5 ° C Low-Power, Analog Out Temperature Sensor datasheet (Rev. A) (ti.com)
Specification Ver	01.00.00 Oct 31,2022 New release
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/

1. Component datasheet

Temperature accuracy	±2.5°C Accuracy from –55°C to +130°C
	±3.5°C Accuracy from –55°C to +150°C
Temperature range	-55 to +150°C
Range of power supply voltage (Vdd)	1.8 to 5.5[V]
	Normal : 3.3[V]
Output voltage (Vout)	Linear -11.77 [mV/°C] Typ. (-55 to +150°C)
	0 [°C] 1863.9 [mV] Typ.
	25[°C] 1547 [mV] Typ.
Calculation	Vout = 1.8639V + (-0.01177 V/°C × Ta)
	Ta = (Vout – 1.8639V) / (-0.01177 V/°C)
Vdd vs Vout	Non-link
Applications	IoT etc

- Defense radio
- Radar
- Avionics
- Sensors and imaging

2. Component Software IF specification

The software interface specifications based on the TMP9A00-EP 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

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

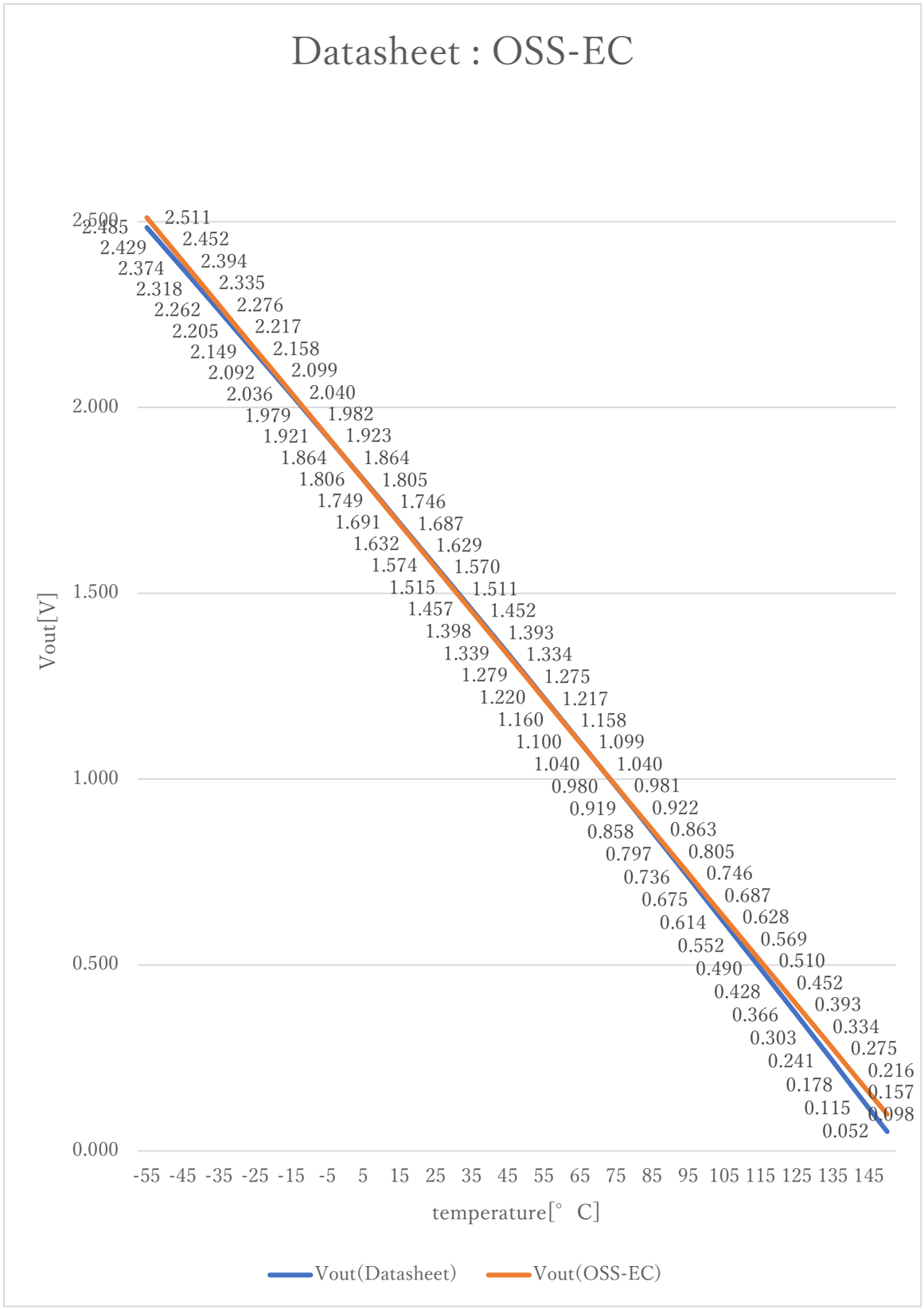
Voltage value to physical value conversion formula

$$y = (vi - iTMP9A00\_xoff) / iTMP9A00\_gain + iTMP9A00\_yoff \quad [℃]$$

$$iTMP9A00\_min \leqq y \leqq iTMP9A00\_max$$

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 iTMP9A00_xoff          1.8639F        // X offset [V]
#define iTMP9A00_yoff          0.0F           // Y offset [℃]
#define iTMP9A00_gain          -0.01177F      // Gain [V/℃]
#define iTMP9A00_max           150.0F         // Temperature Max [℃]
#define iTMP9A00_min           -55.0F         // Temperature Min [℃]
```



$Vout(Datasheet) = (-3.88 \times 10^{-6} \times Ta^2) + (-1.15 \times 10^{-2} \times Ta) + 1.8639V$

3. File Structure and Definitions

TMP9A00.h

```
#include "user_define.h"

// Components number
#define iTMP9A00            132U                    // Texas Instruments TMP9A00-EP

// TMP9A00-EP System Parts definitions
#define iTMP9A00_xoff       1.8639F                 // X offset [V]
#define iTMP9A00_yoff       0.0F                    // Y offset [℃]
#define iTMP9A00_gain       -0.01177F               // Gain [V/℃]
#define iTMP9A00_max        150.0F                  // Temperature Max [℃]
#define iTMP9A00_min        -55.0F                  // Temperature Min [℃]

extern const tbl_adc_t tbl_TMP9A00;
```

TMP9A00.cpp

```
#include        "TMP9A00.h"
#if     iTMP9A00_ma == iSMA                             // Simple moving average filter
static float32 TMP9A00_sma_buf[iTMP9A00_SMA_num];
static const sma_f32_t TMP9A00_Phy_SMA =
{
        iInitial ,                                      // Initial state
        iTMP9A00_SMA_num ,                          // Simple moving average number & buf size
        0U ,                                            // buffer position
        0.0F ,                                          // sum
        &TMP9A00_sma_buf[0]                             // buffer
};
#elif   iTMP9A00_ma == iEMA                             // Exponential moving average filter
static const ema_f32_t TMP9A00_Phy_EMA =
{
        iInitial ,                                      // Initial state
        0.0F ,                                          // Xn-1
        iTMP9A00_EMA_K                                  // Exponential smoothing factor
};
#elif   iTMP9A00_ma == iWMA                             // Weighted moving average filter
static float32 TMP9A00_wma_buf[iTMP9A00_WMA_num];
static const wma_f32_t TMP9A00_Phy_WMA =
{
        iInitial ,                                      // Initial state
        iTMP9A00_WMA_num ,                        // Weighted moving average number & buf size
        0U ,                                            // buffer poition
        iTMP9A00_WMA_num * (iTMP9A00_WMA_num + 1)/2 ,   // kn sum
        &TMP9A00_wma_buf[0]                             // Xn buffer
};
#else                                                   // Non-moving average filter
#endif

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

```
const tbl_adc_t tbl_TMP9A00 =
{
        iTMP9A00                ,
        iTMP9A00_pin            ,
        iTMP9A00_xoff           ,
        iTMP9A00_yoff           ,
        iTMP9A00_gain           ,
        iTMP9A00_max            ,
        iTMP9A00_min            ,
        iTMP9A00_ma             ,

#if     iTMP9A00_ma == iSMA                     // Simple moving average filter
        &TMP9A00_Phy_SMA        ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#elif   iTMP9A00_ma == iEMA                     // Exponential moving average filter
        (sma_f32_t*)iDummy_adr ,
        &TMP9A00_Phy_EMA        ,
        (wma_f32_t*)iDummy_adr
#elif   iTMP9A00_ma == iWMA                     // Weighted moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        &TMP9A00_Phy_WMA
#else                                           // Non-moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#endif

};
```