

Component manufacturer

Specification document of TMP9A00-EP

Model number	TMP9A00-EP
Datashasta	TMD0A00 ED +25° C Low Down Analog Out T

Texas Instruments

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	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 ± 2.5 ° C Accuracy from -55 ° C to +130 ° C

 $\pm 3.5\,^{\circ}$ C Accuracy from $-55\,^{\circ}$ C to $+150\,^{\circ}$ C

Temperature range $-55 \text{ to } +150^{\circ} \text{ C}$

Range of power supply voltage (Vdd) 1.8 to 5.5[V]

Normal : 3.3[V]

Output voltage (Vout) Linear $-11.77 \text{ [mV/}^{\circ} \text{ C] Typ. } (-55 \text{ to } +150^{\circ} \text{ C})$

0 [° C] 1863.9 [mV] Typ. 25[° C] 1547 [mV] Typ.

Calculation $Vout = 1.8639V + (-0.01177 \text{ V/}^{\circ} \text{ C} \times \text{Ta})$

 $Ta = (Vout - 1.8639V) / (-0.01177 V/^{\circ} C)$

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

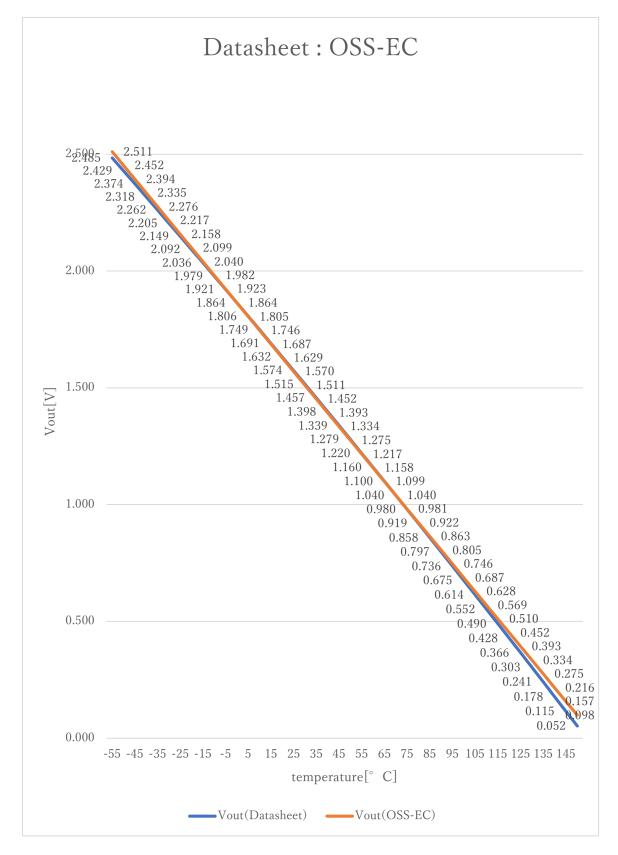
$$vi = (ai \times iADC_vdd) / 2^{iADC_bit}$$
 [V]

Voltage value to physical value conversion formula

$$y = (vi - iTMP9A00_xoff) / iTMP9A00_gain + iTMP9A00_yoff [°C] iTMP9A00_min $\leq y \leq iTMP9A00_max$$$

```
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 iTMP9A00_xoff
                                    1.8639F
#define iTMP9A00_yoff
                                    <u>0. 0F</u>
                                                     // Y offset [°C]
                                                     // Gain [V/°C]
#define iTMP9A00_gain
                                    -0. 01177F
#define iTMP9A00_max
                                                     // Temperature Max [°C]
                                    <u>150. 0F</u>
#define iTMP9A00_min
                                    -55. OF
                                                     // Temperature Min [°C]
```





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
                                                         // Texas Instruments TMP9A00-EP
                             132U
// TMP9A00-EP System Parts definitions
#define iTMP9A00_xoff
                             1.8639F
                                                         // X offset [V]
#define iTMP9A00_yoff
                             <u>0. 0F</u>
                                                         // Y offset [°C]
#define iTMP9A00_gain
                             -0. 01177F
                                                         // Gain [V/°C]
#define iTMP9A00_max
                             150. OF
                                                         // Temperature Max [°C]
#define iTMP9A00_min
                             <u>-55. 0F</u>
                                                          // Temperature Min [°C]
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
        OU ,
                                                        // 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
                                                        // buffer poition
        OU ,
        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
};
```