

Specification document of LM35, LM35A

Component manufacturer	Texas Instruments
Model number	LM35, LM35A
D . 1 .	IMOE Describe Continued Towns and Consult I

LM35 Precision Centigrade Temperature Sensors datasheet (Rev. Datasheets

H)

01.00.00 Specification Ver Nov 3,2022 New release Rui Long Lab Inc. https://rui-long-lab.com/ Documentation provided

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

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://ossec.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 ± 0.4 ° C Typ Accuracy $T_A = 25$ ° C

 $\pm\,0.8\,^{\circ}\ C\quad Typ \qquad \qquad Accuracy\ T_A = T_{\text{\tiny MAX}}(150^{\circ}\ C)$

 ± 0.8 ° C Typ Accuracy $T_A = T_{MIN}(-55$ ° C)

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

Range of power supply voltage (Vdd) 4.0 to 30.0[V]

Output voltage (Vout) Linear $10 \text{ [mV/}^{\circ} \text{ C]}$ Typ

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

· Power Supplies

· Battery Management

• HVAC

Aplicances



2. Component Software IF specification

The software interface specifications based on the LM35, LM35A 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

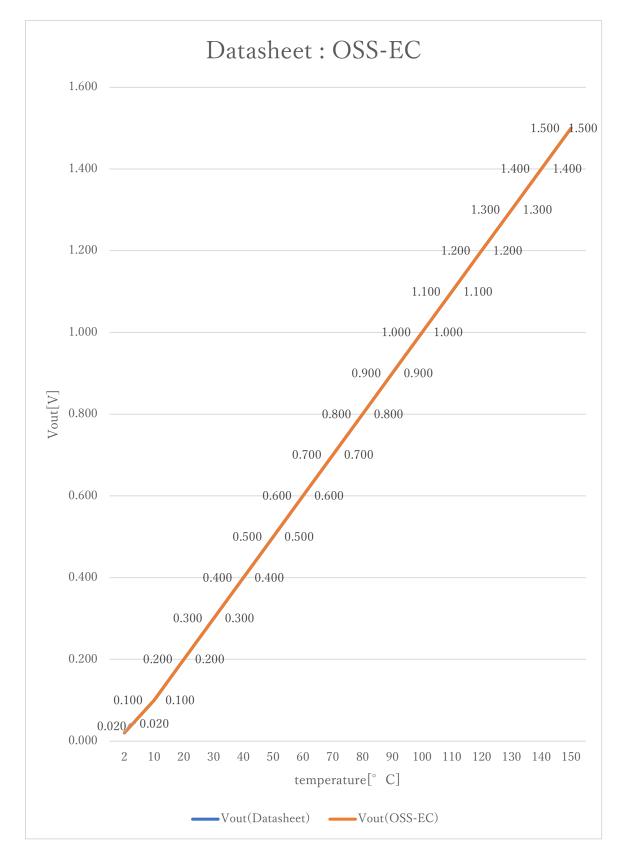
$$vi = (ai \times iADC_vdd) / 2^{iADC_bit}$$
 [V]

Voltage value to physical value conversion formula

y = (vi - iLM35_xoff) / iLM35_gain + iLM35_yoff [°C] iLM35_min
$$\leq$$
 y \leq iLM35_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 iLM35_xoff
                                    0. OF
#define iLM35_yoff
                                    <u>0. 0F</u>
                                                     // Y offset [°C]
                                                     // Gain [V/°C]
#define iLM35_gain
                                    0. 01F
#define iLM35_max
                                                     // Temperature Max [°C]
                                    <u>150. 0F</u>
#define iLM35_min
                                    2. OF
                                                     // Temperature Min [°C]
                                                     // CAUTION:-55[° 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

LM35.h

```
#include "user_define.h"
// Components number
#define iLM35
                             126U
                                                          // Texas Instruments LM35, LM35A
// LM35, LM35A System Parts definitions
#define iLM35_xoff
                             <u>0. 0F</u>
                                                          // X offset [V]
#define iLM35_yoff
                             <u>0. 0F</u>
                                                          // Y offset [°C]
                             0.01F
                                                          // Gain [V/°C]
#define iLM35_gain
#define iLM35_max
                             150. OF
                                                          // Temperature Max [°C]
#define iLM35_min
                             2. 0F
                                                          // Temperature Min [°C]
                                                          // CAUTION:-55[° C], the circuit
                                                             needs a voltage Offset
extern const tbl_adc_t tbl_LM35;
```



LM35.cpp

```
#include
                "LM35. h"
#if
        iLM35_ma == iSMA
                                                         // Simple moving average filter
static float32 LM35_sma_buf[iLM35_SMA_num];
static const sma_f32_t LM35_Phy_SMA =
        iInitial ,
                                                         // Initial state
        iLM35_SMA_num ,
                                                       // Simple moving average number & buf size
        OU ,
                                                         // buffer position
        0.0F,
                                                         // sum
        &LM35_sma_buf[0]
                                                         // buffer
};
#elif
        iLM35_ma == iEMA
                                                         // Exponential moving average filter
static const ema_f32_t LM35_Phy_EMA =
{
        iInitial ,
                                                         // Initial state
        0.0F,
                                                         // Xn-1
        iLM35_EMA_K
                                                         // Exponential smoothing factor
};
#elif iLM35_ma == iWMA
                                                         // Weighted moving average filter
static float32 LM35_wma_buf[iLM35_WMA_num];
static const wma_f32_t LM35_Phy_WMA =
{
        iInitial ,
                                                         // Initial state
        iLM35_WMA_num ,
                                               // Weighted moving average number & buf size
                                                         // buffer poition
        OU ,
                                                         // kn sum
        iLM35\_WMA\_num * (iLM35\_WMA\_num + 1)/2,
        &LM35_wma_buf[0]
                                                         // Xn buffer
};
#else
                                                         // Non-moving average filter
#endif
#define iDummy_adr
                         0xffffffff
                                                         // Dummy address
```



```
const tbl_adc_t tbl_LM35 =
        iLM35
        iLM35_pin
        iLM35_xoff
        iLM35_yoff
        iLM35_gain
        iLM35_max
        iLM35_min
        iLM35_ma
#if
        iLM35_ma == iSMA
                                                          // Simple moving average filter
        &LM35_Phy_SMA
        (ema_f32_t*)iDummy_adr,
        (wma_f32_t*) iDummy_adr
#elif
                                                          // Exponential moving average filter
        iLM35_ma == iEMA
        (sma_f32_t*) iDummy_adr,
        &LM35_Phy_EMA
        (wma_f32_t*) iDummy_adr
                                                          // Weighted moving average filter
#elif
        iLM35_ma == iWMA
        (sma_f32_t*) iDummy_adr ,
        (ema_f32_t*) iDummy_adr ,
        &LM35_Phy_WMA
#else
                                                          // Non-moving average filter
        (sma_f32_t*) iDummy_adr ,
         (ema_f32_t*) iDummy_adr ,
        (wma_f32_t*) iDummy_adr
#endif
};
```