Specification document of LM35, LM35A

Component manufacturer	Texas Instruments		
Model number	LM35, LM35A		
Datasheets	LM35 Precision Centigrade Temperature Sensors datasheet (Rev. H)		
Specification Ver	01.00.00	Nov 3,2022	New release
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/		

License

1. Component datasheet

Temperature accuracy	±0.4 ° C Typ	Accuracy T_A = 25° C
	±0.8 ° C Typ	Accuracy T_A = T_{MAX}(150° C)
	±0.8 ° C Typ	Accuracy T_A = T_{MIN}(-55° C)
Temperature range	-55 to +150° C	
Range of power supply voltage (Vdd)	4.0 to 30.0[V]	
Output voltage (Vout)	Linear 10 [mV/° C]	Typ
Calculation	Vout = 0.01 V/° C × Ta	
	Ta = Vout / (0.01 V/° 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

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

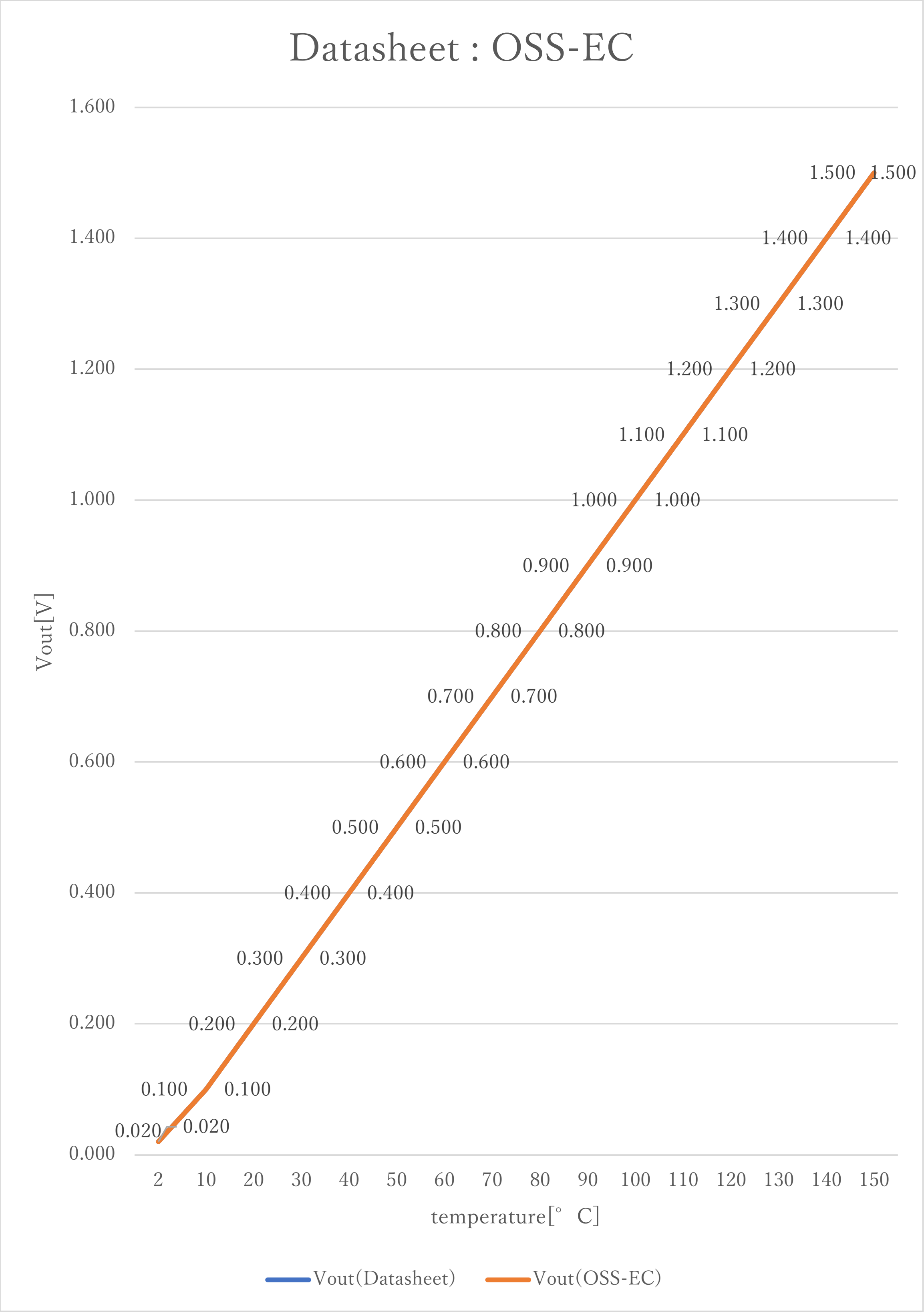
Voltage value to physical value conversion formula

$$y = (vi - iLM35\_xoff) / iLM35\_gain + iLM35\_yoff \quad [℃]$$

$$iLM35\_min \leqq y \leqq iLM35\_max$$

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 iLM35_xoff          0.0F          // X offset [V]
#define iLM35_yoff          0.0F          // Y offset [℃]
#define iLM35_gain          0.01F         // Gain [V/℃]
#define iLM35_max           150.0F        // Temperature Max [℃]
#define iLM35_min           2.0F          // Temperature Min [℃]
                                          // CAUTION:-55[° C],the circuit
                                             needs a voltage Offset
```



Vout(Datasheet) = 10 mV/° C × T° 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          0.0F                      // X offset [V]
#define iLM35_yoff          0.0F                      // Y offset [℃]
#define iLM35_gain          0.01F                     // Gain [V/℃]
#define iLM35_max           150.0F                    // Temperature Max [℃]
#define iLM35_min           2.0F                      // Temperature Min [℃]
                                                      // 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
        0U ,                                            // 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
        0U ,                                            // buffer poition
        iLM35_WMA_num * (iLM35_WMA_num + 1)/2 ,         // kn sum
        &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   iLM35_ma == iEMA                                // Exponential moving average filter
        (sma_f32_t*)iDummy_adr ,
        &LM35_Phy_EMA         ,
        (wma_f32_t*)iDummy_adr
#elif   iLM35_ma == iWMA                                // Weighted moving average filter
        (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

};
```