Specification document of LM35D

Component manufacturer	Texas Instruments		
Model number	LM35D		
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/		

1. Component datasheet

Temperature accuracy	±0.6° C Typ	Accuracy T_A = 25° C
	±0.9° C Typ	Accuracy T_A = T_{MAX}(100° C)
	±0.9° C Typ	Accuracy T_A = T_{MIN}(0° C)
Temperature range	0 to +100° 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 LM35D 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

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

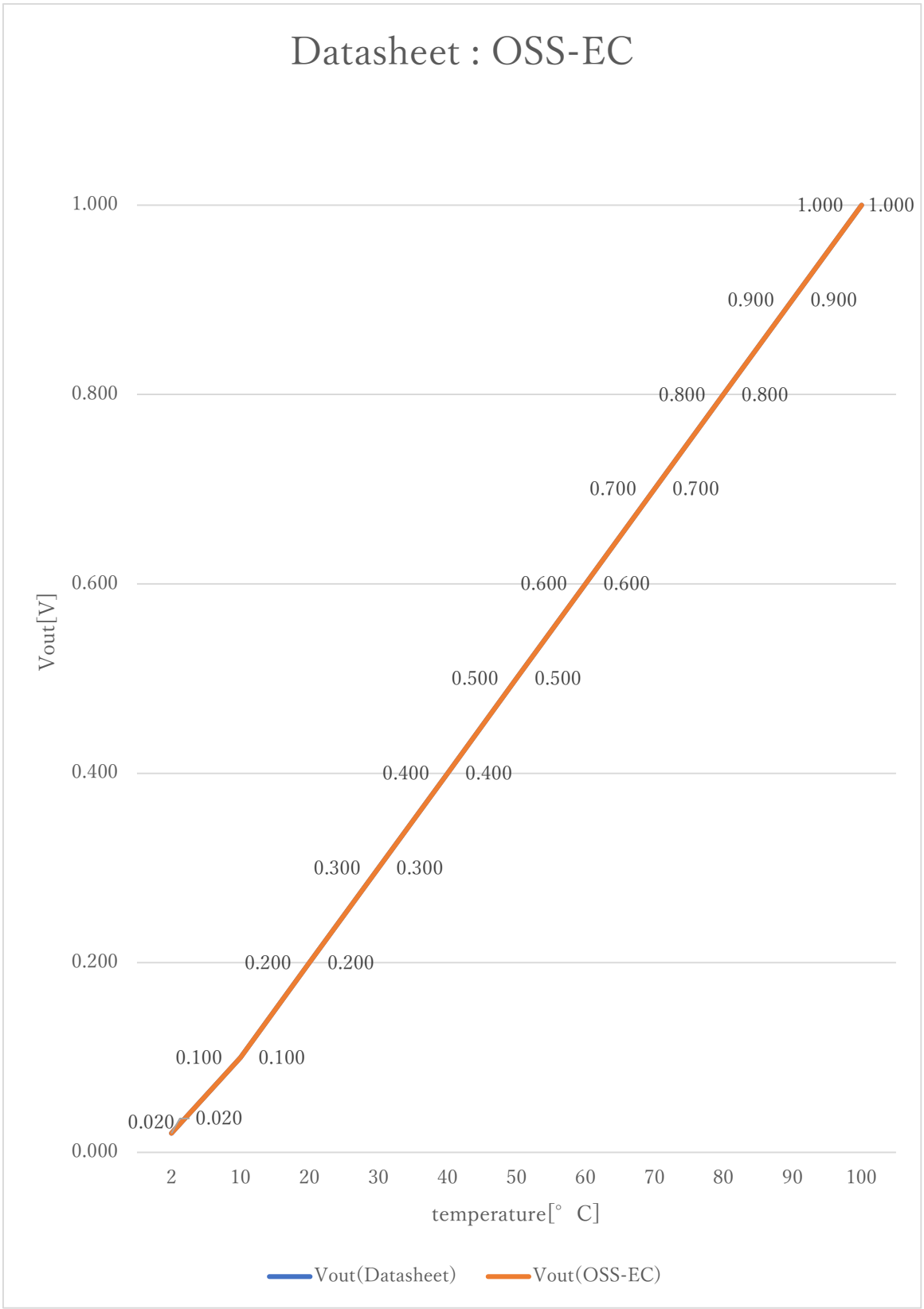
Voltage value to physical value conversion formula

$$y = (vi - iLM35D\_xoff) / iLM35D\_gain + iLM35D\_yoff \quad [℃]$$

$$iLM35D\_min \leqq y \leqq iLM35D\_max$$

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 iLM35D_xoff             0.0F          // X offset [V]
#define iLM35D_yoff             0.0F          // Y offset [℃]
#define iLM35D_gain             0.01F         // Gain [V/℃]
#define iLM35D_max              100.0F        // Temperature Max [℃]
#define iLM35D_min              2.0F          // Temperature Min [℃]
```



Vout(Datasheet) = 10 mV/° C × T ° C

3. File Structure and Definitions

LM35D.h

```
#include "user_define.h"

// Components number
#define iLM35D              128U                    // Texas Instruments LM35D

// LM35D System Parts definitions
#define iLM35D_xoff         0.0F                    // X offset [V]
#define iLM35D_yoff         0.0F                    // Y offset [℃]
#define iLM35D_gain         0.01F                   // Gain [V/℃]
#define iLM35D_max          100.0F                  // Temperature Max [℃]
#define iLM35D_min          2.0F                    // Temperature Min [℃]

extern const tbl_adc_t tbl_LM35D;
```

LM35D.cpp

```
#include        "LM35D.h"
#if     iLM35D_ma == iSMA                               // Simple moving average filter
static float32 LM35D_sma_buf[iLM35D_SMA_num];
static const sma_f32_t LM35D_Phy_SMA =
{
        iInitial ,                                      // Initial state
        iLM35D_SMA_num ,                                // Simple moving average number & buf size
        0U ,                                            // buffer position
        0.0F ,                                          // sum
        &LM35D_sma_buf[0]                               // buffer
};
#elif   iLM35D_ma == iEMA                               // Exponential moving average filter
static const ema_f32_t LM35D_Phy_EMA =
{
        iInitial ,                                      // Initial state
        0.0F ,                                          // Xn-1
        iLM35D_EMA_K                                    // Exponential smoothing factor
};
#elif   iLM35D_ma == iWMA                               // Weighted moving average filter
static float32 LM35D_wma_buf[iLM35D_WMA_num];
static const wma_f32_t LM35D_Phy_WMA =
{
        iInitial ,                                      // Initial state
        iLM35D_WMA_num ,                        // Weighted moving average number & buf size
        0U ,                                            // buffer poition
        iLM35D_WMA_num * (iLM35D_WMA_num + 1)/2 ,       // kn sum
        &LM35D_wma_buf[0]                               // Xn buffer
};
#else                                                   // Non-moving average filter
#endif

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

```
const tbl_adc_t tbl_LM35D =
{
        iLM35D                  ,
        iLM35D_pin              ,
        iLM35D_xoff             ,
        iLM35D_yoff             ,
        iLM35D_gain             ,
        iLM35D_max              ,
        iLM35D_min              ,
        iLM35D_ma               ,

#if     iLM35D_ma == iSMA                       // Simple moving average filter
        &LM35D_Phy_SMA          ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#elif   iLM35D_ma == iEMA                       // Exponential moving average filter
        (sma_f32_t*)iDummy_adr ,
        &LM35D_Phy_EMA          ,
        (wma_f32_t*)iDummy_adr
#elif   iLM35D_ma == iWMA                       // Weighted moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        &LM35D_Phy_WMA
#else                                           // Non-moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#endif

};
```