Specification document of AD22100A

Component manufacturer	Analog Devices
Model number	AD22100A
Datasheets	[AD22100 (REV. D) (analog.com)]
Specification Ver	01.00.00 Oct 03,2022 New release
	01.00.01 Oct 18,2022 Corrected license content Application item add
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/

1. Component datasheet

Temperature accuracy	±2.0° C (–40° C to +85° C)
Range of power supply voltage (Vdd)	4.0 to 6.5[V]
Output voltage (Vout)	Linear 22.5×Vdd/5 [mV/° C] Typ.
	Vdd = 5.0 [V]
	-40 [° C] 0.475[V] Typ.
	85 [° C] 3.288 [V] Typ.
Calculation	Vout = (Vdd/5 V) × (1.375 V + 22.5 mV/° C × Ta)
	Ta = (Vout / (Vdd/5V)) - 1.375V) / 22.5 mV/° C
Applications	IoT etc
	・ HVAC systems
	・ System temperature compensation
	・ Board level temperature sensing
	・ Electronic thermostats
	Automotive

2. Component Software IF specification

The software interface specifications based on the AD22100A 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

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

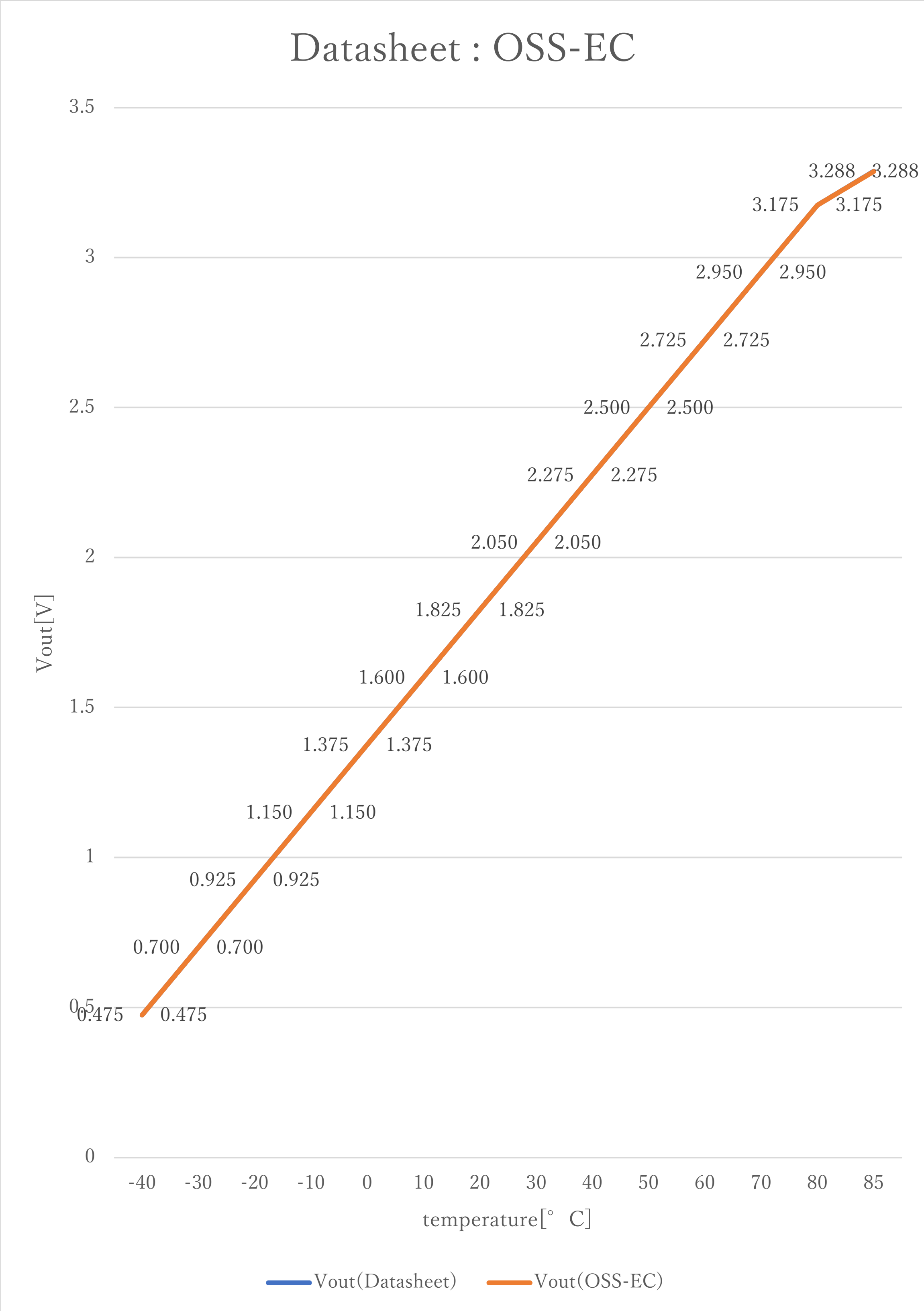
Voltage value to physical value conversion formula

$$y = (vi - iAD22100A\_xoff) / iAD22100A\_gain + iAD22100A\_yoff \quad [℃]$$

$$iAD22100A\_min \leqq y \leqq iAD22100A\_max$$

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 iAD22100A_xoff    (1.375F*(iADC_vdd/5.0))   // X offset [V]
#define iAD22100A_yoff    0.0F                      // Y offset [℃]
#define iAD22100A_gain    (0.0225F*(iADC_vdd/5.0))  // Gain [V/℃]
#define iAD22100A_max     85.0F                     // Temperature Max [℃]
#define iAD22100A_min     -40.0F                    // Temperature Min [℃]
```


Datasheet : OSS-EC
3.5
3
2.5
2
1.5
1
0.5
0
Vout[V]
3.288 3.288
3.175 3.175
2.950 2.950
2.725 2.725
2.500 2.500
2.275 2.275
2.050 2.050
1.825 1.825
1.600 1.600
1.375 1.375
1.150 1.150
0.925 0.925
0.700 0.700
0.475 0.475
-40 -30 -20 -10 0 10 20 30 40 50 60 70 80 85
temperature[° C]
Vout(Datasheet)
Vout(OSS-EC)

3. File Structure and Definitions

AD22100A.h

```
#include "user_define.h"

// Components number
#define iAD22100A              106U                      // Analog devices AD22100A

// AD22100A System Parts definitions
#define iAD22100A_xoff       (1.375F*(iADC_vdd/5.0))   // X offset [V]
#define iAD22100A_yoff       0.0F                      // Y offset [℃]
#define iAD22100A_gain       (0.0225F*(iADC_vdd/5.0))  // Gain [V/℃]
#define iAD22100A_max        85.0F                     // Temperature Max [℃]
#define iAD22100A_min        -40.0F                    // Temperature Min [℃]

extern const tbl_adc_t tbl_AD22100A;
```

AD22100A.cpp

```
#include        "AD22100A.h"
#if     iAD22100A_ma == iSMA                        // Simple moving average filter
static float32 AD22100A_sma_buf[iAD22100A_SMA_num];
static const sma_f32_t AD22100A_Phy_SMA =
{
        iInitial ,                                  // Initial state
        iAD22100A_SMA_num ,                        // Simple moving average number & buf size
        0U ,                                        // buffer position
        0.0F ,                                      // sum
        &AD22100A_sma_buf[0]                        // buffer
};
#elif   iAD22100A_ma == iEMA                        // Exponential moving average filter
static const ema_f32_t AD22100A_Phy_EMA =
{
        iInitial ,                                  // Initial state
        0.0F ,                                      // Xn-1
        iAD22100A_EMA_K                             // Exponential smoothing factor
};
#elif   iAD22100A_ma == iWMA                        // Weighted moving average filter
static float32 AD22100A_wma_buf[iAD22100A_WMA_num];
static const wma_f32_t AD22100A_Phy_WMA =
{
        iInitial ,                                  // Initial state
        iAD22100A_WMA_num ,                        // Weighted moving average number & buf size
        0U ,                                        // buffer poition
        iAD22100A_WMA_num * (iAD22100A_WMA_num + 1)/2 , // kn sum
        &AD22100A_wma_buf[0]                        // Xn buffer
};
#else                                               // Non-moving average filter
#endif

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

```
const tbl_adc_t tbl_AD22100A =
{
        iAD22100A               ,
        iAD22100A_pin           ,
        iAD22100A_xoff          ,
        iAD22100A_yoff          ,
        iAD22100A_gain          ,
        iAD22100A_max           ,
        iAD22100A_min           ,
        iAD22100A_ma            ,

#if     iAD22100A_ma == iSMA                    // Simple moving average filter
        &AD22100A_Phy_SMA       ,
        (ema_f32_t*)iDummy_adr  ,
        (wma_f32_t*)iDummy_adr
#elif   iAD22100A_ma == iEMA                    // Exponential moving average filter
        (sma_f32_t*)iDummy_adr  ,
        &AD22100A_Phy_EMA       ,
        (wma_f32_t*)iDummy_adr
#elif   iAD22100A_ma == iWMA                    // Weighted moving average filter
        (sma_f32_t*)iDummy_adr  ,
        (ema_f32_t*)iDummy_adr  ,
        &AD22100A_Phy_WMA
#else                                           // Non-moving average filter
        (sma_f32_t*)iDummy_adr  ,
        (ema_f32_t*)iDummy_adr  ,
        (wma_f32_t*)iDummy_adr
#endif

};
```