Specification document of AD22100K

Component manufacturer	Analog Devices
Model number	AD22100K
Datasheets	AD22100 (REV. D) (analog.com)
Specification Ver	01.00.00 Oct 03,2022 New release
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/

1. Component datasheet

Temperature accuracy	±0.75° C (0° C to +100° 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]
	0 [° C] 1.375[V] Typ.
	100 [° C] 3.625 [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

2. Component Software IF specification

The software interface specifications based on the AD22100K 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

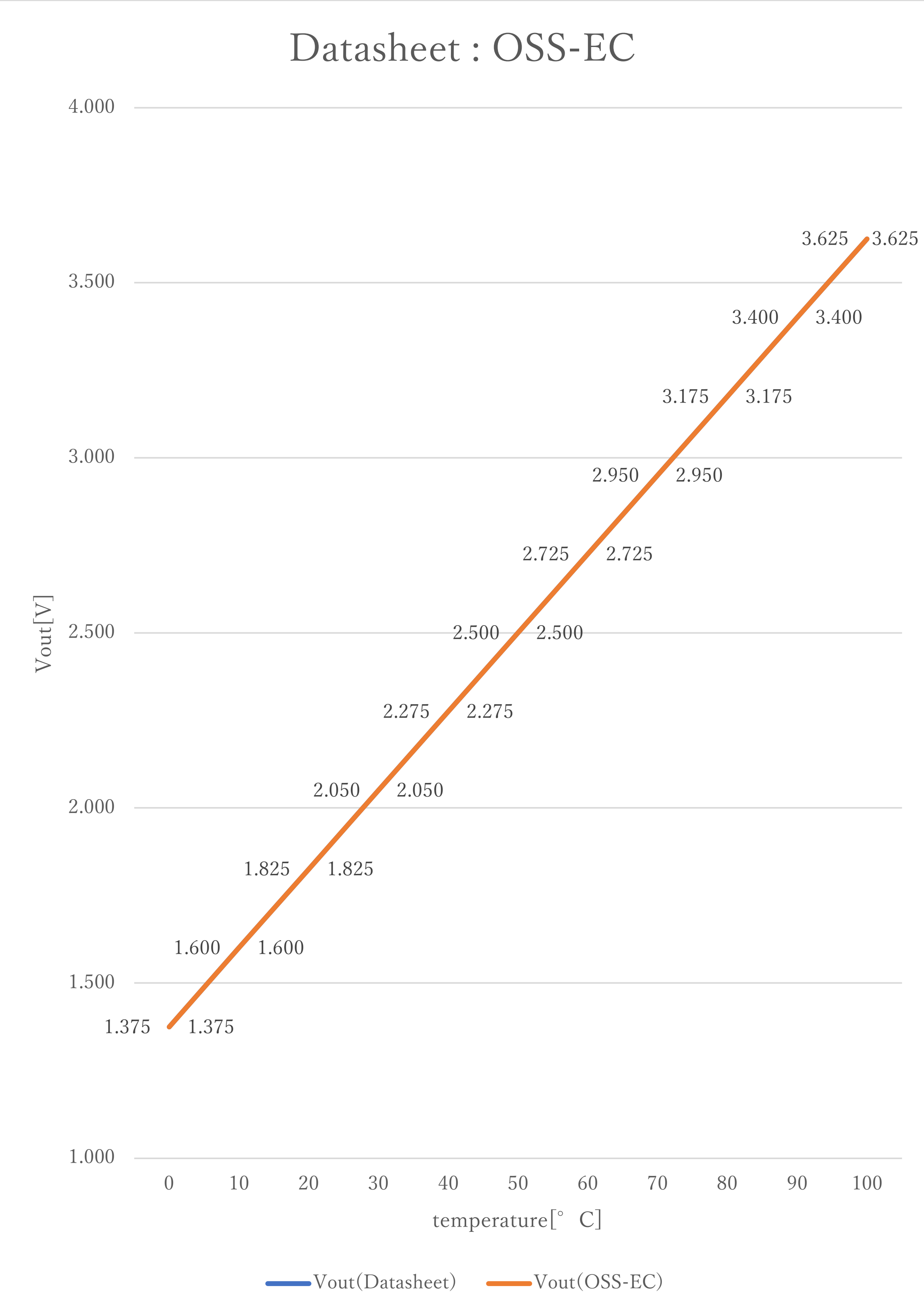
$$vi = (ai \times iADC\_vdd) / 2^{iADC\_bit} \quad [V]$$

Voltage value to physical value conversion formula

$$y = (vi - iAD22100K\_xoff) / iAD22100K\_gain + iAD22100K\_yoff \quad [℃]$$

$$iAD22100K\_min \leqq y \leqq iAD22100K\_max$$

```
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 iAD22100K_xoff    (1.375F*(iADC_vdd/5.0))   // X offset [V]
#define iAD22100K_yoff    0.0F                      // Y offset [℃]
#define iAD22100K_gain    (0.0225F*(iADC_vdd/5.0))  // Gain [V/℃]
#define iAD22100K_max     100.0F                    // Temperature Max [℃]
#define iAD22100K_min     0.0F                      // Temperature Min [℃]
```


Datasheet : OSS-EC
4.000
3.500
3.000
2.500
2.000
1.500
1.000
Vout[V]
1.375
1.600
1.825
2.050
2.275
2.500
2.725
2.950
3.175
3.400
3.625
0
10
20
30
40
50
60
70
80
90
100
temperature[° C]
Vout(Datasheet)
Vout(OSS-EC)

3. File Structure and Definitions

AD22100K.h

```
#include "user_define.h"

// Components number
#define iAD22100K              107U                    // Analog devices AD22100K

// AD22100K System Parts definitions
#define iAD22100K_xoff      (1.375F*(iADC_vdd/5.0))  // X offset [V]
#define iAD22100K_yoff      0.0F                     // Y offset [℃]
#define iAD22100K_gain      (0.0225F*(iADC_vdd/5.0)) // Gain [V/℃]
#define iAD22100K_max       100.0F                   // Temperature Max [℃]
#define iAD22100K_min       0.0F                     // Temperature Min [℃]

extern const tbl_adc_t tbl_AD22100K;
```

AD22100K.cpp

```
#include        "AD22100K.h"
#if     iAD22100K_ma == iSMA                    // Simple moving average filter
static float32 AD22100K_sma_buf[iAD22100K_SMA_num];
static const sma_f32_t AD22100K_Phy_SMA =
{
        iInitial ,                              // Initial state
        iAD22100K_SMA_num ,                     // Simple moving average number & buf size
        0U ,                                    // buffer position
        0.0F ,                                  // sum
        &AD22100K_sma_buf[0]                    // buffer
};
#elif   iAD22100K_ma == iEMA                    // Exponential moving average filter
static const ema_f32_t AD22100K_Phy_EMA =
{
        iInitial ,                              // Initial state
        0.0F ,                                  // Xn-1
        iAD22100K_EMA_K                         // Exponential smoothing factor
};
#elif   iAD22100K_ma == iWMA                    // Weighted moving average filter
static float32 AD22100K_wma_buf[iAD22100K_WMA_num];
static const wma_f32_t AD22100K_Phy_WMA =
{
        iInitial ,                              // Initial state
        iAD22100K_WMA_num ,                     // Weighted moving average number & buf size
        0U ,                                    // buffer poition
        iAD22100K_WMA_num * (iAD22100K_WMA_num + 1)/2 ,     // kn sum
        &AD22100K_wma_buf[0]                    // Xn buffer
};
#else                                           // Non-moving average filter
#endif

#define iDummy_adr      0xffffffff              // Dummy address

const tbl_adc_t tbl_AD22100K =
```

```
{
        iAD22100K               ,
        iAD22100K_pin           ,
        iAD22100K_xoff          ,
        iAD22100K_yoff          ,
        iAD22100K_gain          ,
        iAD22100K_max           ,
        iAD22100K_min           ,
        iAD22100K_ma            ,

#if     iAD22100K_ma == iSMA                    // Simple moving average filter
        &AD22100K_Phy_SMA       ,
        (ema_f32_t*)iDummy_adr  ,
        (wma_f32_t*)iDummy_adr
#elif   iAD22100K_ma == iEMA                    // Exponential moving average filter
        (sma_f32_t*)iDummy_adr  ,
        &AD22100K_Phy_EMA       ,
        (wma_f32_t*)iDummy_adr
#elif   iAD22100K_ma == iWMA                    // Weighted moving average filter
        (sma_f32_t*)iDummy_adr  ,
        (ema_f32_t*)iDummy_adr  ,
        &AD22100K_Phy_WMA
#else                                           // Non-moving average filter
        (sma_f32_t*)iDummy_adr  ,
        (ema_f32_t*)iDummy_adr  ,
        (wma_f32_t*)iDummy_adr
#endif

};
```