Specification document of S-8110C, S-8120C

Component manufacturer	ABLIC
Model number	S-8110C, S-8120C
Datasheets	S-8110C/8120C - CMOS 温度センサ IC S-8110C/8120C シリーズ - エイブリック株式会社 (ablic.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	S-8110C Series: ±5.0° C (–30° C to +100° C)
	S-8120C Series: ±2.5° C (–30° C to +100° C)
Range of power supply voltage(Vdd)	2.4 to 10.0[V]
Output voltage (Vout)	Linear - 8.20 [mV/° C] Typ. (-30° C to 100° C)
	Vdd = 5.0 [V]
	-30 [° C] 1.951 [V] Typ.
	30 [° C] 1.474 [V] Typ.
	100 [° C] 0.882 [V] Typ.
Vdd vs Vout	Non-link (ΔVout 0.004 to 0.005 [V])

Ta[° C]	Vdd[V]	Vout[V]
-40	2.48	2.032
	10.00	2.036
30	2.48	1.472
	10.00	1.477
100	2.48	0.880
	10.00	0.885

2. Component Software IF specification

The software interface specifications based on the S-8110C/S-8120C 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

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

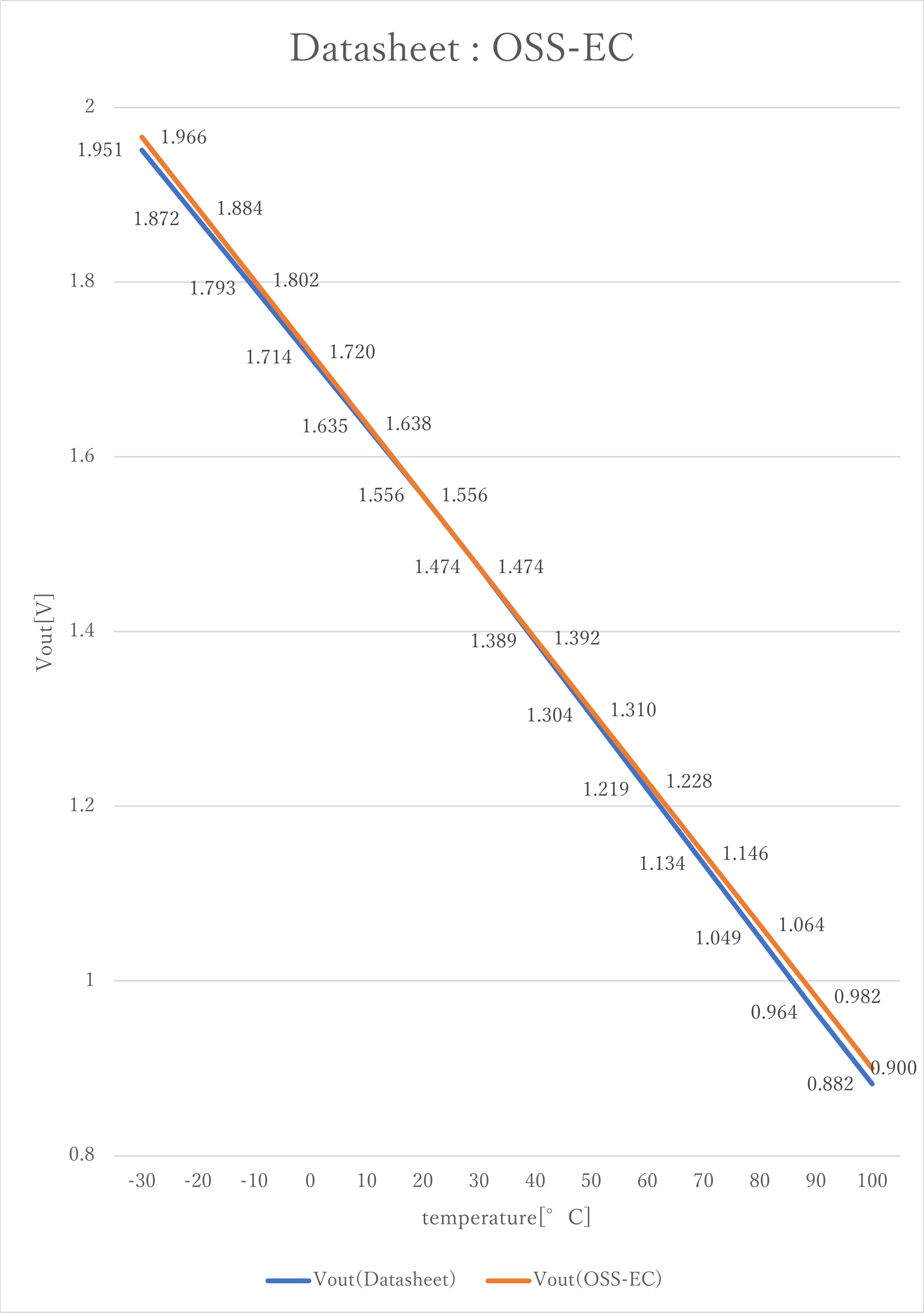
Voltage value to physical value conversion formula

$$y = (vi - iS8110C\_xoff) / iS8110C\_gain + iS8110C\_yoff \quad [℃]$$

$$iS8110C\_min \leqq y \leqq iS8110C\_max$$

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 iS8110C_xoff        1.474F            // X offset [V]
#define iS8110C_yoff        30.0F             // Y offset [℃]
#define iS8110C_gain        -0.0082F          // Gain [V/℃]
#define iS8110C_max         100.0F            // Temperature Max [℃]
#define iS8110C_min         -30.0F            // Temperature Min [℃]
```


Datasheet : OSS-EC
2
1.8
1.6
1.4
1.2
1
0.8
Vout[V]
-30
-20
-10
0
10
20
30
40
50
60
70
80
90
100
temperature[° C]
1.951
1.966
1.872
1.884
1.793
1.802
1.714
1.720
1.635
1.638
1.556
1.556
1.474
1.474
1.389
1.392
1.304
1.310
1.219
1.228
1.134
1.146
1.049
1.064
0.964
0.982
0.882
0.900
Vout(Datasheet)
Vout(OSS-EC)

3. File Structure and Definitions

S8110C.h

```
#include "user_define.h"

// Components number
#define iS8110C              105U                       // ABLIC S-8110C, S-8120C

// S-8110C,S-8120C System Parts definitions
#define iS8110C_xoff         1.474F                     // X offset [V]
#define iS8110C_yoff         30.0F                      // Y offset [℃]
#define iS8110C_gain         -0.0082F                   // Gain [V/℃]
#define iS8110C_max          100.0F                     // Temperature Max [℃]
#define iS8110C_min          -30.0F                     // Temperature Min [℃]

extern const tbl_adc_t tbl_S8110C;
```

S8110C.cpp

```
#include        "S8110C.h"
#if     iS8110C_ma == iSMA                            // Simple moving average filter
static float32 S8110C_sma_buf[iS8110C_SMA_num];
static const sma_f32_t S8110C_Phy_SMA =
{
        iInitial ,                                      // Initial state
        iS8110C_SMA_num ,                             // Simple moving average number & buf size
        0U ,                                            // buffer position
        0.0F ,                                          // sum
        &S8110C_sma_buf[0]                              // buffer
};
#elif   iS8110C_ma == iEMA                              // Exponential moving average filter
static const ema_f32_t S8110C_Phy_EMA =
{
        iInitial ,                                      // Initial state
        0.0F ,                                          // Xn-1
        iS8110C_EMA_K                                   // Exponential smoothing factor
};
#elif   iS8110C_ma == iWMA                              // Weighted moving average filter
static float32 S8110C_wma_buf[iS8110C_WMA_num];
static const wma_f32_t S8110C_Phy_WMA =
{
        iInitial ,                                      // Initial state
        iS8110C_WMA_num ,                          // Weighted moving average number & buf size
        0U ,                                            // buffer poition
        iS8110C_WMA_num * (iS8110C_WMA_num + 1)/2 ,     // kn sum
        &S8110C_wma_buf[0]                              // Xn buffer
};
#else                                                   // Non-moving average filter
#endif

#define iDummy_adr       0xffffffff                     // Dummy address

const tbl_adc_t tbl_S8110C =
{
```

```
            iS8110C                     ,
            iS8110C_pin                 ,
            iS8110C_xoff                ,
            iS8110C_yoff                ,
            iS8110C_gain                ,
            iS8110C_max                 ,
            iS8110C_min                 ,
            iS8110C_ma                  ,

    #if     iS8110C_ma == iSMA                          // Simple moving average filter
            &S8110C_Phy_SMA             ,
            (ema_f32_t*)iDummy_adr        ,
            (wma_f32_t*)iDummy_adr
    #elif   iS8110C_ma == iEMA                          // Exponential moving average filter
            (sma_f32_t*)iDummy_adr        ,
            &S8110C_Phy_EMA             ,
            (wma_f32_t*)iDummy_adr
    #elif   iS8110C_ma == iWMA                          // Weighted moving average filter
            (sma_f32_t*)iDummy_adr        ,
            (ema_f32_t*)iDummy_adr        ,
            &S8110C_Phy_WMA
    #else                                                 // Non-moving average filter
            (sma_f32_t*)iDummy_adr        ,
            (ema_f32_t*)iDummy_adr        ,
            (wma_f32_t*)iDummy_adr
    #endif

    };
```