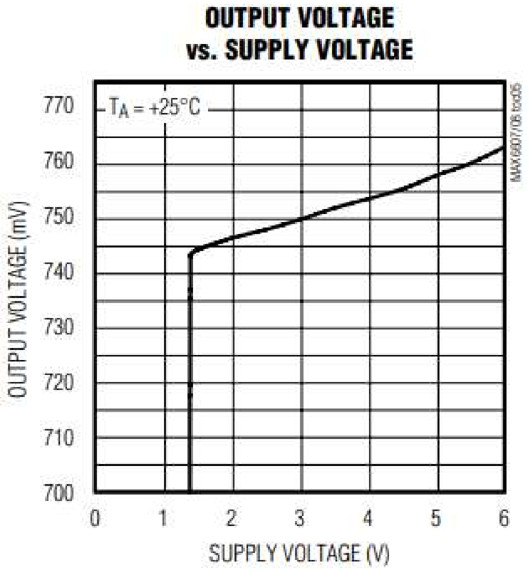
Specification document of MAX6607IXK-T, MAX6608IUK-T

Component manufacturer	Maxim Integrated		
Model number	MAX6607IXK-T, MAX6608IUK-T		
Datasheets	MAX6607/08 DS (maximintegrated.com)		
Specification Ver	01.00.00	Oct 09,2022	New release
	01.00.01	Oct 18,2022	Corrected license content
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/		

License

1. Component datasheet

Temperature accuracy	±2.0° C (Max, +20 to +50° C)
Temperature range	-20 to +85° C
Range of power supply voltage (Vdd)	1.8 to 3.6[V]
Output voltage (Vout)	Linear 10 [mV/° C] Typ.
	Vdd = 3.3 [V]
	0 [° C] 0.500[V] Typ.
Calculation	Vout = 0.5V + (0.01 V/° C × Ta)
	Ta = (Vout – 0.5V) / 0.01 V/° C



Applications IoT etc

- Digital Cameras
- Battery Packs
- Portable Equipment
- GPS Equipmen

2. Component Software IF specification

The software interface specifications based on the MAX6607IXK-T, MAX6608IUK-T 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

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

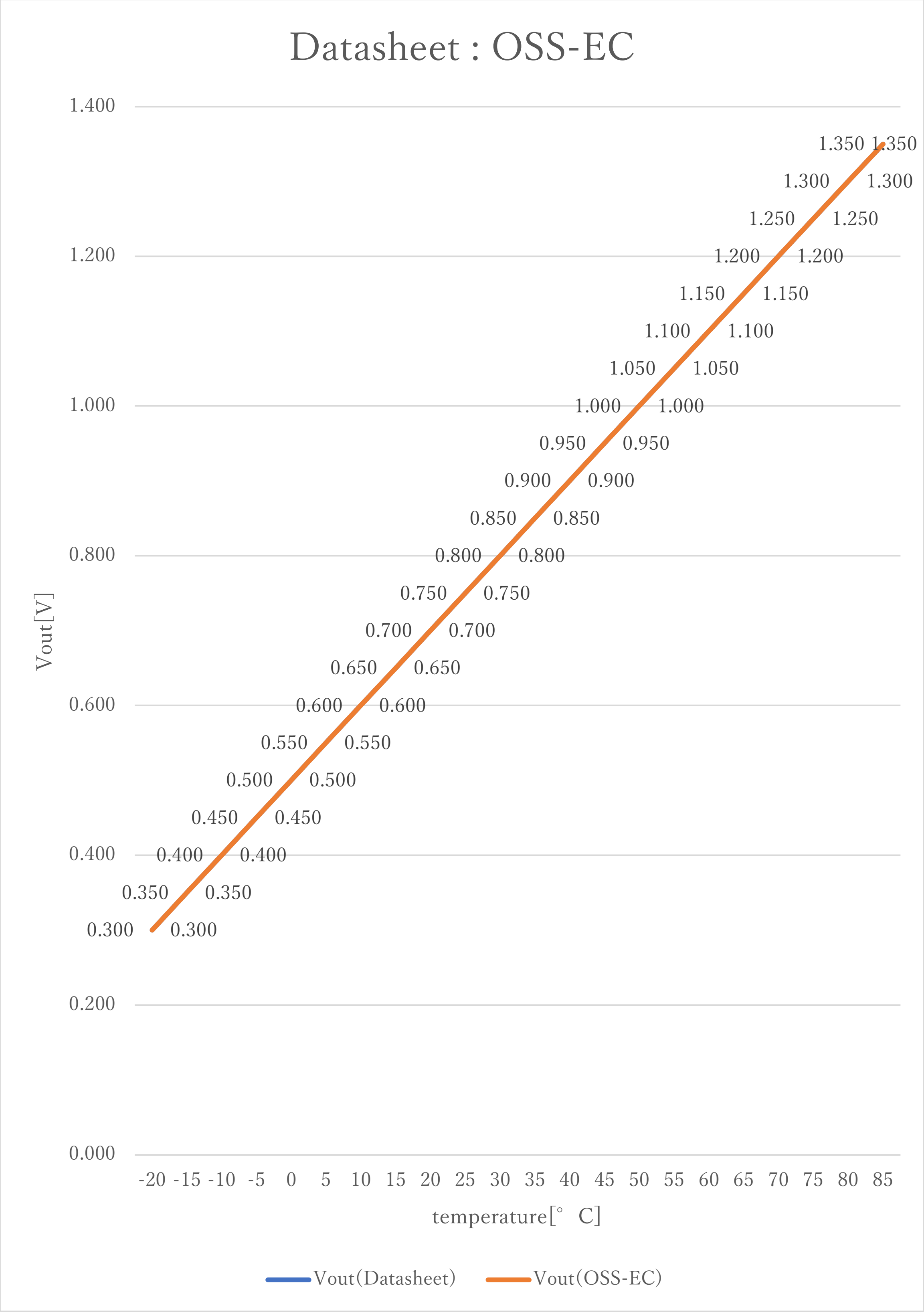
Voltage value to physical value conversion formula

$$y = (vi - iMAX6607\_xoff) / iMAX6607\_gain + iMAX6607\_yoff \quad [℃]$$

$$iMAX6607\_min \leqq y \leqq iMAX6607\_max$$

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 iMAX6607_xoff          0.5F          // X offset [V]
#define iMAX6607_yoff          0.0F          // Y offset [℃]
#define iMAX6607_gain          0.01F         // Gain [V/℃]
#define iMAX6607_max           85.0F         // Temperature Max [℃]
#define iMAX6607_min           -20.0F        // Temperature Min [℃]
```


Datasheet : OSS-EC
Vout[V]
1.400
1.200
1.000
0.800
0.600
0.400
0.200
0.000
0.300 0.300
0.350 0.350
0.400 0.400
0.450 0.450
0.500 0.500
0.550 0.550
0.600 0.600
0.650 0.650
0.700 0.700
0.750 0.750
0.800 0.800
0.850 0.850
0.900 0.900
0.950 0.950
1.000 1.000
1.050 1.050
1.100 1.100
1.150 1.150
1.200 1.200
1.250 1.250
1.300 1.300
1.350 1.350
-20 -15 -10 -5 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85
temperature[° C]
Vout(Datasheet)
Vout(OSS-EC)

3. File Structure and Definitions

MAX6607.h

```
#include "user_define.h"

// Components number
#define iMAX6607        112U                    // Maxim Integrated MAX6607IXK/MAX6608IUK

// MAX6607 System Parts definitions
#define iMAX6607_xoff      0.5F                      // X offset [V]
#define iMAX6607_yoff      0.0F                      // Y offset [℃]
#define iMAX6607_gain      0.01F                     // Gain [V/℃]
#define iMAX6607_max       85.0F                     // Temperature Max [℃]
#define iMAX6607_min       -20.0F                    // Temperature Min [℃]

extern const tbl_adc_t tbl_MAX6607;
```

MAX6607.cpp

```
#include        ″MAX6607.h″
#if     iMAX6607_ma == iSMA                             // Simple moving average filter
static float32 MAX6607_sma_buf[iMAX6607_SMA_num];
static const sma_f32_t MAX6607_Phy_SMA =
{
        iInitial ,                                      // Initial state
        iMAX6607_SMA_num ,                         // Simple moving average number & buf size
        0U ,                                            // buffer position
        0.0F ,                                          // sum
        &MAX6607_sma_buf[0]                             // buffer
};
#elif   iMAX6607_ma == iEMA                             // Exponential moving average filter
static const ema_f32_t MAX6607_Phy_EMA =
{
        iInitial ,                                      // Initial state
        0.0F ,                                          // Xn-1
        iMAX6607_EMA_K                                  // Exponential smoothing factor
};
#elif   iMAX6607_ma == iWMA                             // Weighted moving average filter
static float32 MAX6607_wma_buf[iMAX6607_WMA_num];
static const wma_f32_t MAX6607_Phy_WMA =
{
        iInitial ,                                      // Initial state
        iMAX6607_WMA_num ,                       // Weighted moving average number & buf size
        0U ,                                            // buffer poition
        iMAX6607_WMA_num * (iMAX6607_WMA_num + 1)/2 ,   // kn sum
        &MAX6607_wma_buf[0]                             // Xn buffer
};
#else                                                   // Non-moving average filter
#endif

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

```
const tbl_adc_t tbl_MAX6607 =
{
        iMAX6607                ,
        iMAX6607_pin            ,
        iMAX6607_xoff           ,
        iMAX6607_yoff           ,
        iMAX6607_gain           ,
        iMAX6607_max            ,
        iMAX6607_min            ,
        iMAX6607_ma             ,

#if     iMAX6607_ma == iSMA                             // Simple moving average filter
        &MAX6607_Phy_SMA        ,
        (ema_f32_t*)iDummy_adr  ,
        (wma_f32_t*)iDummy_adr
#elif   iMAX6607_ma == iEMA                             // Exponential moving average filter
        (sma_f32_t*)iDummy_adr  ,
        &MAX6607_Phy_EMA        ,
        (wma_f32_t*)iDummy_adr
#elif   iMAX6607_ma == iWMA                             // Weighted moving average filter
        (sma_f32_t*)iDummy_adr  ,
        (ema_f32_t*)iDummy_adr  ,
        &MAX6607_Phy_WMA
#else                                                   // Non-moving average filter
        (sma_f32_t*)iDummy_adr  ,
        (ema_f32_t*)iDummy_adr  ,
        (wma_f32_t*)iDummy_adr
#endif

};
```