Specification document of MAX6613MXK-T, MAX6613MXK/V-T

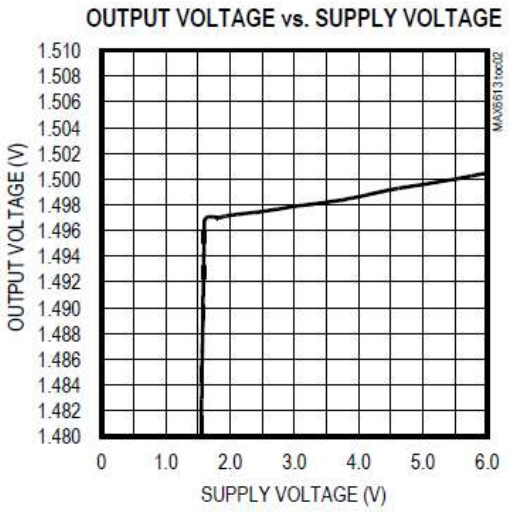
Component manufacturer	Maxim Integrated		
Model number	MAX6613MXK-T, MAX6613MXK/V-T		
Datasheets	MAX6613.pdf (maximintegrated.com)		
Specification Ver	01.00.00	Oct 10,2022	New release
	01.00.01	Oct 18,2022	Corrected license content
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/		

1. Component datasheet

Temperature accuracy	±4.0° C (Max, 0 to +50° C)
	±4.4° C (Max, -20 to +80° C)
Temperature range	-55 to +130° C
Range of power supply voltage (Vdd)	1.8 to 5.5[V]
Output voltage (Vout)	Linear 11.23 [mV/° C] Typ.
	0 [° C] 1.8455 [V] Typ.
Calculation	Vout = 1.8455V + (-0.01123 V/° C × Ta)
	Ta = (Vout – 1.8455V) / (-0.01123 V/° C)

More accurate temperature calculation

Vout = 1.8455V - (0.01105 V/° C × Ta) - (2.25 ×10^{-6} × Ta^2)



Applications	IoT etc
	- Cellular Phones
	- GPS Equipment
	- Medical Instruments
	- Battery Management
	- Appliances
	- Disk Drives
	- Printers
	- Fax Machines
	- HVAC Digital Cameras
	Automotive

2. Component Software IF specification

The software interface specifications based on the MAX6613MXK-T, MAX6613MXK/V-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

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

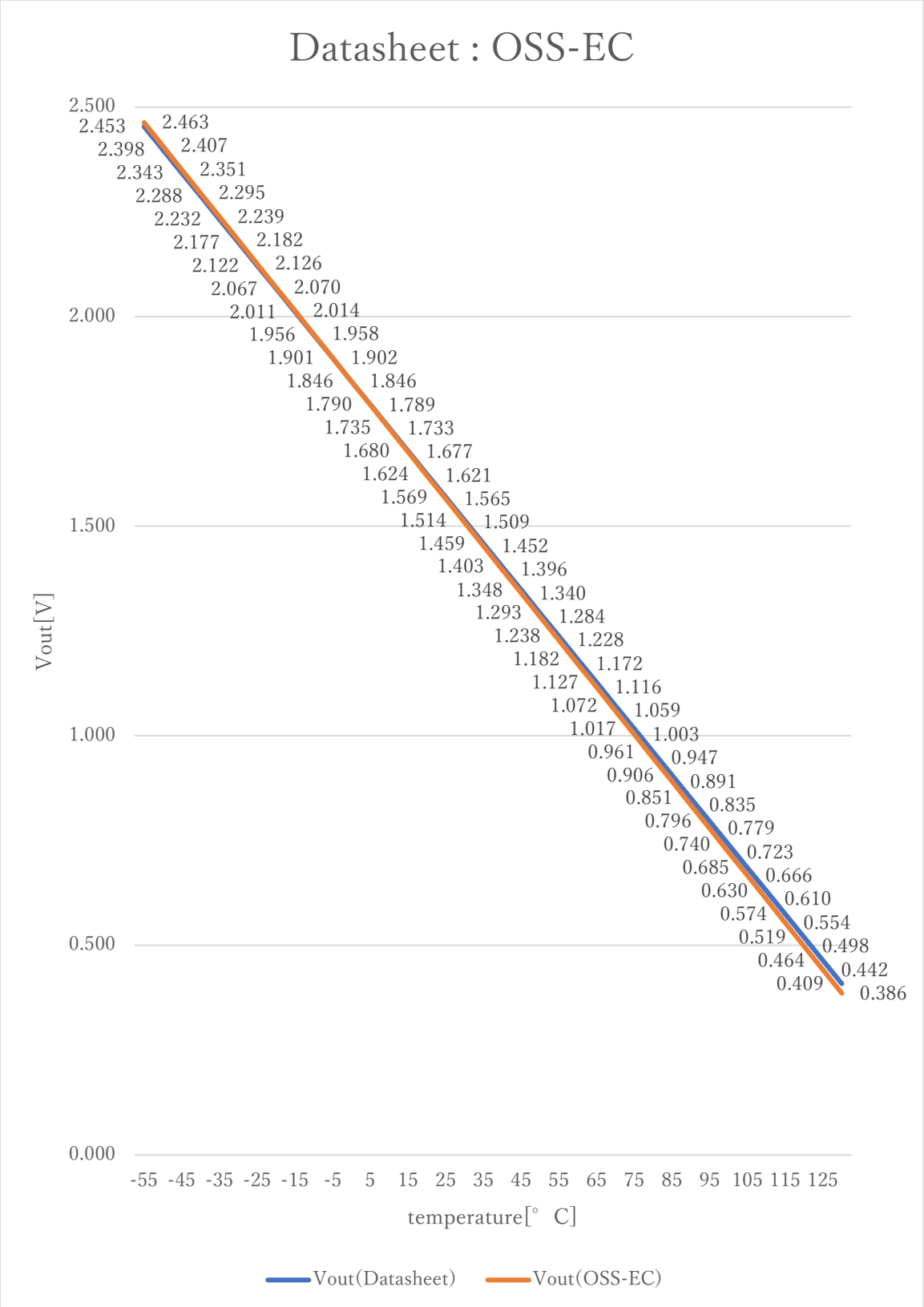
Voltage value to physical value conversion formula

$$y = (vi - iMAX6613\_xoff) / iMAX6613\_gain + iMAX6613\_yoff \quad [℃]$$

$$iMAX6613\_min \leqq y \leqq iMAX6613\_max$$

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 iMAX6613_xoff          1.8455F        // X offset [V]
#define iMAX6613_yoff          0.0F           // Y offset [℃]
#define iMAX6613_gain          -0.01123F      // Gain [V/℃]
#define iMAX6613_max           130.0F         // Temperature Max [℃]
#define iMAX6613_min           -55.0F         // Temperature Min [℃]
```



Vout(Datasheet) = 1.8455V - (0.01105 V/° C × Ta) - (2.25 ×10^{-6} × Ta^2)

3. File Structure and Definitions

MAX6613.h

```
#include "user_define.h"

// Components number
#define iMAX6613            113U                // Maxim Integrated MAX6613MXK/MAX6613MXK/V

// MAX6613 System Parts definitions
#define iMAX6613_xoff       1.8455F             // X offset [V]
#define iMAX6613_yoff       0.0F                // Y offset [℃]
#define iMAX6613_gain       -0.01123F           // Gain [V/℃]
#define iMAX6613_max        130.0F              // Temperature Max [℃]
#define iMAX6613_min        -55.0F              // Temperature Min [℃]

extern const tbl_adc_t tbl_MAX6613;
```

MAX6613.cpp

```
#include        "MAX6613.h"
#if     iMAX6613_ma == iSMA                             // Simple moving average filter
static float32 MAX6613_sma_buf[iMAX6613_SMA_num];
static const sma_f32_t MAX6613_Phy_SMA =
{
        iInitial ,                                      // Initial state
        iMAX6613_SMA_num ,                          // Simple moving average number & buf size
        0U ,                                            // buffer position
        0.0F ,                                          // sum
        &MAX6613_sma_buf[0]                             // buffer
};
#elif   iMAX6613_ma == iEMA                             // Exponential moving average filter
static const ema_f32_t MAX6613_Phy_EMA =
{
        iInitial ,                                      // Initial state
        0.0F ,                                          // Xn-1
        iMAX6613_EMA_K                                  // Exponential smoothing factor
};
#elif   iMAX6613_ma == iWMA                             // Weighted moving average filter
static float32 MAX6613_wma_buf[iMAX6613_WMA_num];
static const wma_f32_t MAX6613_Phy_WMA =
{
        iInitial ,                                      // Initial state
        iMAX6613_WMA_num ,                        // Weighted moving average number & buf size
        0U ,                                            // buffer poition
        iMAX6613_WMA_num * (iMAX6613_WMA_num + 1)/2 ,   // kn sum
        &MAX6613_wma_buf[0]                             // Xn buffer
};
#else                                                   // Non-moving average filter
#endif

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

```
const tbl_adc_t tbl_MAX6613 =
{
        iMAX6613                ,
        iMAX6613_pin            ,
        iMAX6613_xoff           ,
        iMAX6613_yoff           ,
        iMAX6613_gain           ,
        iMAX6613_max            ,
        iMAX6613_min            ,
        iMAX6613_ma             ,

#if     iMAX6613_ma == iSMA                     // Simple moving average filter
        &MAX6613_Phy_SMA        ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#elif   iMAX6613_ma == iEMA                     // Exponential moving average filter
        (sma_f32_t*)iDummy_adr ,
        &MAX6613_Phy_EMA        ,
        (wma_f32_t*)iDummy_adr
#elif   iMAX6613_ma == iWMA                     // Weighted moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        &MAX6613_Phy_WMA
#else                                           // Non-moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#endif

};
```