

# 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	2
2.	Component Software IF specification	3
3	File Structure and Definitions	5

#### License

Open Source Software for Embedded Components ("OSS-EC") is open source software files and related documentation files for component products used in computer systems and other applications. OSS-EC is provided to those who accept the OSS-EC Terms of Use for the OSS-EC site; see <a href="https://oss-ec.com/license\_agreement/">https://oss-ec.com/license\_agreement/</a> for the OSS-EC Terms of Use. By downloading the OSS-EC from the OSS-EC site or obtaining the OSS-EC by any means, you accept the Terms of Use. Please read and accept the Terms of Use before using the OSS-EC. If you do not agree to the Terms of Use, please do not use the OSS-EC.



### 1. Component datasheet

Temperature accuracy  $\pm 4.0^{\circ}$  C (Max, 0 to +50° C)

 $\pm 4.4^{\circ}$  C (Max, -20 to +80° C)

Temperature range  $-55 \text{ to } +130^{\circ} \text{ C}$ 

Range of power supply voltage (Vdd) 1.8 to 5.5[V]

Output voltage (Vout) Linear 11.23 [mV/° C] Typ.

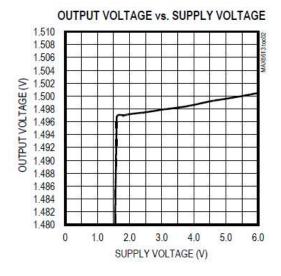
0 [° C] 1.8455 [V] Typ.

Calculation  $Vout = 1.8455V + (-0.01123 \text{ V/}^{\circ} \text{ C} \times \text{Ta})$ 

 $Ta = (Vout - 1.8455V) / (-0.01123 V/^{\circ} C)$ 

More accurate temperature calculation

Vout = 1.8455V - ( 0.01105 V/° C × Ta ) - ( $2.25 \times 10^{-6}$  × Ta<sup>2</sup>)



#### **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

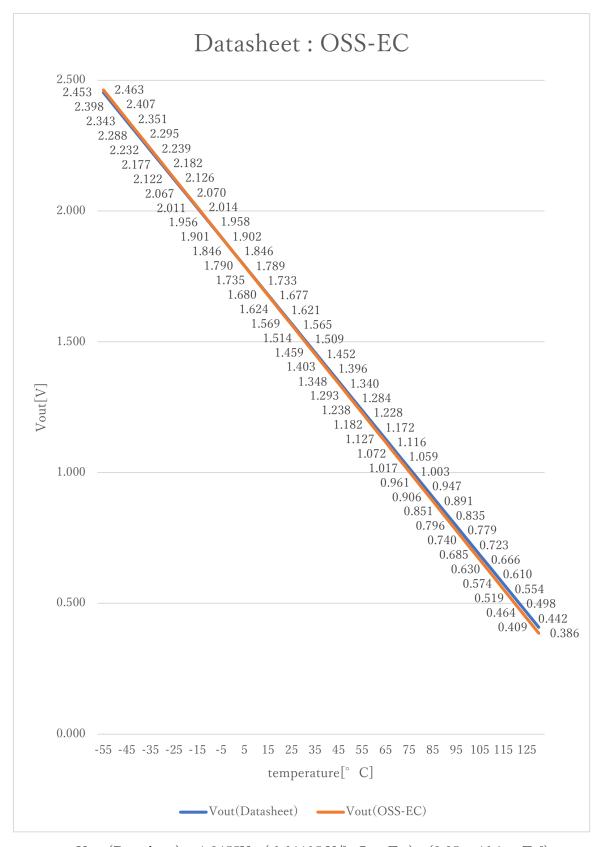
$$vi = (ai \times iADC_vdd) / 2^{iADC_bit}$$
 [V]

Voltage value to physical value conversion formula

y = (vi - iMAX6613\_xoff) / iMAX6613\_gain + iMAX6613\_yoff [°C] iMAX6613\_min 
$$\leq$$
 y  $\leq$  iMAX6613\_max

```
A/D conversion value
ai
٧i
                 Sensor output voltage value [V]
i ADC vdd
                 Sensor supply voltage value [V]
iADC_bit
                 A/D conversion bit length
                 Temperature value [°C]
                                                     // X offset [V]
#define iMAX6613_xoff
                                   1.8455F
#define iMAX6613_yoff
                                   <u>0. 0F</u>
                                                     // Y offset [°C]
                                                     // Gain [V/°C]
#define iMAX6613 gain
                                   -0. 01123F
#define iMAX6613_max
                                                     // Temperature Max [°C]
                                   130. OF
#define iMAX6613_min
                                   -55. OF
                                                     // Temperature Min [°C]
```





 $Vout(Datasheet) = 1.8455V \text{ - ( } 0.01105 \text{ V/}^{\circ} \text{ C} \times \text{Ta ) - (} 2.25 \times 10^{\text{-}6} \times \text{ Ta}^{\text{2}}\text{)}$ 



## 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
                            1.8455F
#define iMAX6613_xoff
                                                         // X offset [V]
#define iMAX6613_yoff
                            0. 0F
                                                         // Y offset [°C]
                            -0. 01123F
                                                         // Gain [V/°C]
#define iMAX6613_gain
#define iMAX6613_max
                            130. OF
                                                         // Temperature Max [°C]
#define iMAX6613_min
                            <u>-55. 0F</u>
                                                         // Temperature Min [°C]
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
        OU ,
                                                         // 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
        OU ,
                                                         // 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

// Dummy address

0xffffffff



```
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
};
```