

# Specification document of MAX6605MXKV

Component manufacturer	Maxim Integrated
Model number	MAX6605MXKV

Datasheets MAX6605 DS (maximintegrated.com)

Specification Ver 01.00.00 Oct 04,2022 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  $\pm 0.75^{\circ}$  C (  $25^{\circ}$  C )

Temperature range  $-40 \text{ to } +85^{\circ} \text{ C}$ 

Range of power supply voltage (Vdd) 2.7 to 5.5[V]

Output voltage ( Vout ) Linear  $11.9 \times Vdd/3.3 \text{ [mV/}^{\circ} \text{ C] Typ.}$ 

Vdd = 3.3 [V]

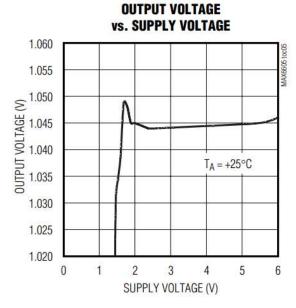
0 [° C] 0.744[V] Typ.

Calculation  $Vout = 0.744V + (0.0119 \text{ V/}^{\circ} \text{ C} \times \text{Ta})$ 

 $Ta = (Vout - 0.744V) / 0.0119 V/^{\circ} C$ 

More accurate temperature calculation

Vout =  $0.744V + (0.0119 \text{ V/}^{\circ} \text{ C} \times \text{Ta}) + (1.604 \times 10^{-6} \times \text{Ta}^{2})$ 



Applications

IoT etc

- Cellular Phones
- Battery Packs
- GPS Equipment
- Digital Cameras

Automotive



## 2. Component Software IF specification

The software interface specifications based on the MAX6605MXKV 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

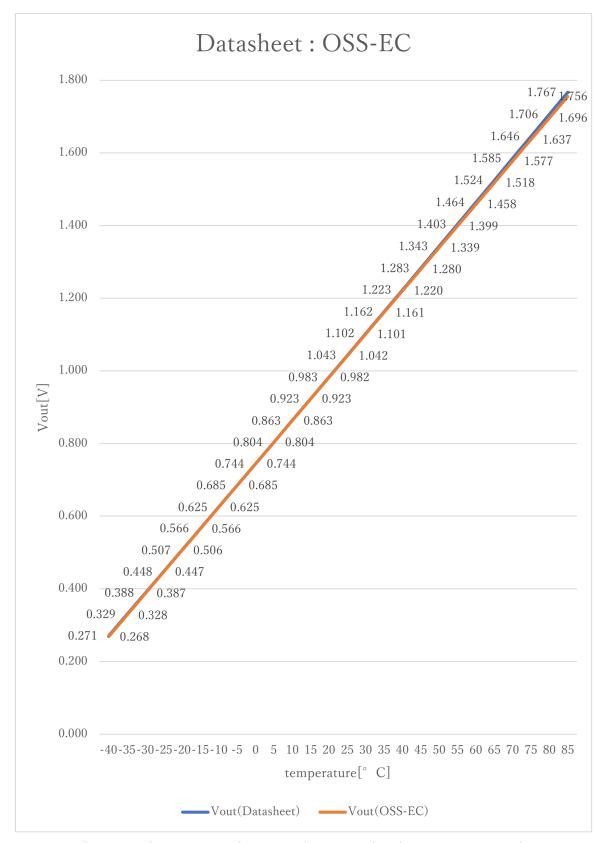
$$vi = (ai \times iADC_vdd) / 2^{iADC_bit}$$
 [V]

Voltage value to physical value conversion formula

```
y = (vi - iMAX6605MXKV\_xoff) / iMAX6605MXKV\_gain + iMAX6605MXKV\_yoff [°C] iMAX6605MXKV\_min <math>\leq y \leq iMAX6605MXKV\_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 iMAX6605MXKV_xoff
                                   0. 744F
#define iMAX6605MXKV_yoff
                                   <u>0. 0F</u>
                                                    // Y offset [°C]
                                                    // Gain [V/°C]
#define iMAX6605MXKV gain
                                   0.0119F
#define iMAX6605MXKV_max
                                   85. OF
                                                    // Temperature Max [°C]
#define iMAX6605MXKV_min
                                   -40. OF
                                                    // Temperature Min [°C]
```





Vout(Datasheet) =  $0.744V + (0.0119 \text{ V/}^{\circ} \text{ C} \times \text{Ta}) + (1.604 \times 10^{-6} \times \text{Ta}^{2})$ 



## 3. File Structure and Definitions

#### MAX6605MXKV.h

```
#include "user_define.h"
// Components number
#define iMAX6605MXKV
                             111U
                                                         // Maxim Integrated MAX6605MXKV
// MAX6605MXKV System Parts definitions
#define iMAX6605MXKV_xoff
                             <u>0. 744F</u>
                                                         // X offset [V]
#define iMAX6605MXKV_yoff
                             0. OF
                                                         // Y offset [°C]
                             0.0119F
                                                         // Gain [V/°C]
#define iMAX6605MXKV_gain
#define iMAX6605MXKV_max
                             85. OF
                                                         // Temperature Max [°C]
#define iMAX6605MXKV_min
                             <u>-40. 0F</u>
                                                          // Temperature Min [°C]
extern const tbl_adc_t tbl_MAX6605MXKV;
```



## MAX6605MXKV.cpp

```
#include
                "MAX6605MXKV. h"
#if
                                                         // Simple moving average filter
        iMAX6605MXKV_ma == iSMA
static float32 MAX6605MXKV_sma_buf[iMAX6605MXKV_SMA_num];
static const sma_f32_t MAX6605MXKV_Phy_SMA =
        iInitial ,
                                                        // Initial state
        iMAX6605MXKV_SMA_num ,
                                                       // Simple moving average number & buf size
        OU ,
                                                         // buffer position
        0.0F,
                                                         // sum
        &MAX6605MXKV_sma_buf[0]
                                                         // buffer
};
#elif
        iMAX6605MXKV_ma == iEMA
                                                         // Exponential moving average filter
static const ema_f32_t MAX6605MXKV_Phy_EMA =
{
        iInitial ,
                                                         // Initial state
        0.0F,
                                                         // Xn-1
        iMAX6605MXKV_EMA_K
                                                         // Exponential smoothing factor
};
#elif
        iMAX6605MXKV_ma == iWMA
                                                         // Weighted moving average filter
static float32 MAX6605MXKV_wma_buf[iMAX6605MXKV_WMA_num];
static const wma_f32_t MAX6605MXKV_Phy_WMA =
{
        iInitial ,
                                                         // Initial state
        iMAX6605MXKV_WMA_num ,
                                                     // Weighted moving average number & buf size
        OU ,
                                                         // buffer poition
        iMAX6605MXKV\_WMA\_num * (iMAX6605MXKV\_WMA\_num + 1)/2 , // kn sum
        &MAX6605MXKV_wma_buf[0]
                                                         // Xn buffer
};
#else
                                                         // Non-moving average filter
#endif
#define iDummy_adr
                         0xffffffff
                                                         // Dummy address
```



```
const tbl_adc_t tbl_MAX6605MXKV =
        iMAX6605MXKV
        iMAX6605MXKV_pin
        iMAX6605MXKV\_xoff
        iMAX6605MXKV\_yoff
        iMAX6605MXKV_gain
        iMAX6605MXKV_max
        iMAX6605MXKV_min
        iMAX6605MXKV_ma
#if
        iMAX6605MXKV_ma == iSMA
                                                         // Simple moving average filter
        &MAX6605MXKV_Phy_SMA
        (ema_f32_t*) iDummy_adr ,
        (wma_f32_t*) iDummy_adr
#elif
        iMAX6605MXKV_ma == iEMA
                                                         // Exponential moving average filter
        (sma_f32_t*) iDummy_adr ,
        &MAX6605MXKV_Phy_EMA
        (wma_f32_t*) iDummy_adr
#elif
        iMAX6605MXKV_ma == iWMA
                                                          // Weighted moving average filter
        (sma_f32_t*) iDummy_adr
        (ema_f32_t*) iDummy_adr,
        &MAX6605MXKV_Phy_WMA
#else
                                                          // Non-moving average filter
        (sma_f32_t*) iDummy_adr ,
        (ema_f32_t*) iDummy_adr
        (wma_f32_t*) iDummy_adr
#endif
};
```