Specification document of MAX6605MXKV

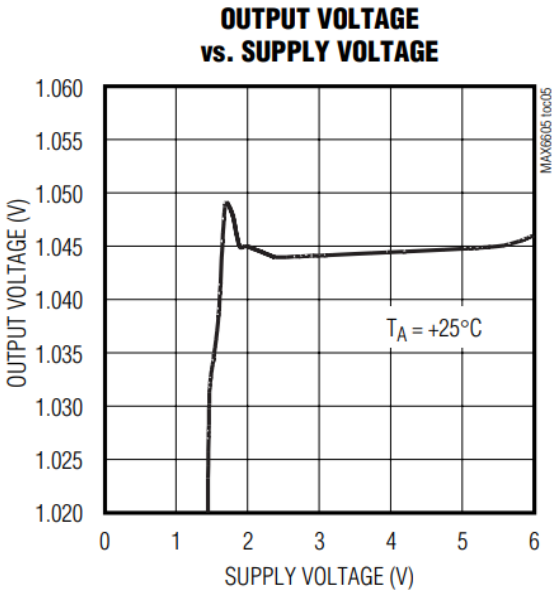
Component manufacturer	Maxim Integrated		
Model number	MAX6605MXKV		
Datasheets	MAX6605 DS (maximintegrated.com)		
Specification Ver	01.00.00	Oct 04,2022	New release
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/		

1. Component datasheet

Temperature accuracy	±0.75° C (25° C)
Temperature range	-40 to +85° C
Range of power supply voltage (Vdd)	2.7 to 5.5[V]
Output voltage (Vout)	Linear 11.9×Vdd/3.3 [mV/° C] Typ. Vdd = 3.3 [V] 0 [° C] 0.744[V] Typ.
Calculation	Vout = 0.744V + (0.0119 V/° C × Ta) Ta = (Vout – 0.744V) / 0.0119 V/° C

More accurate temperature calculation

Vout = 0.744V + (0.0119 V/° C × Ta) + (1.604 ×10^{-6} × 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

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

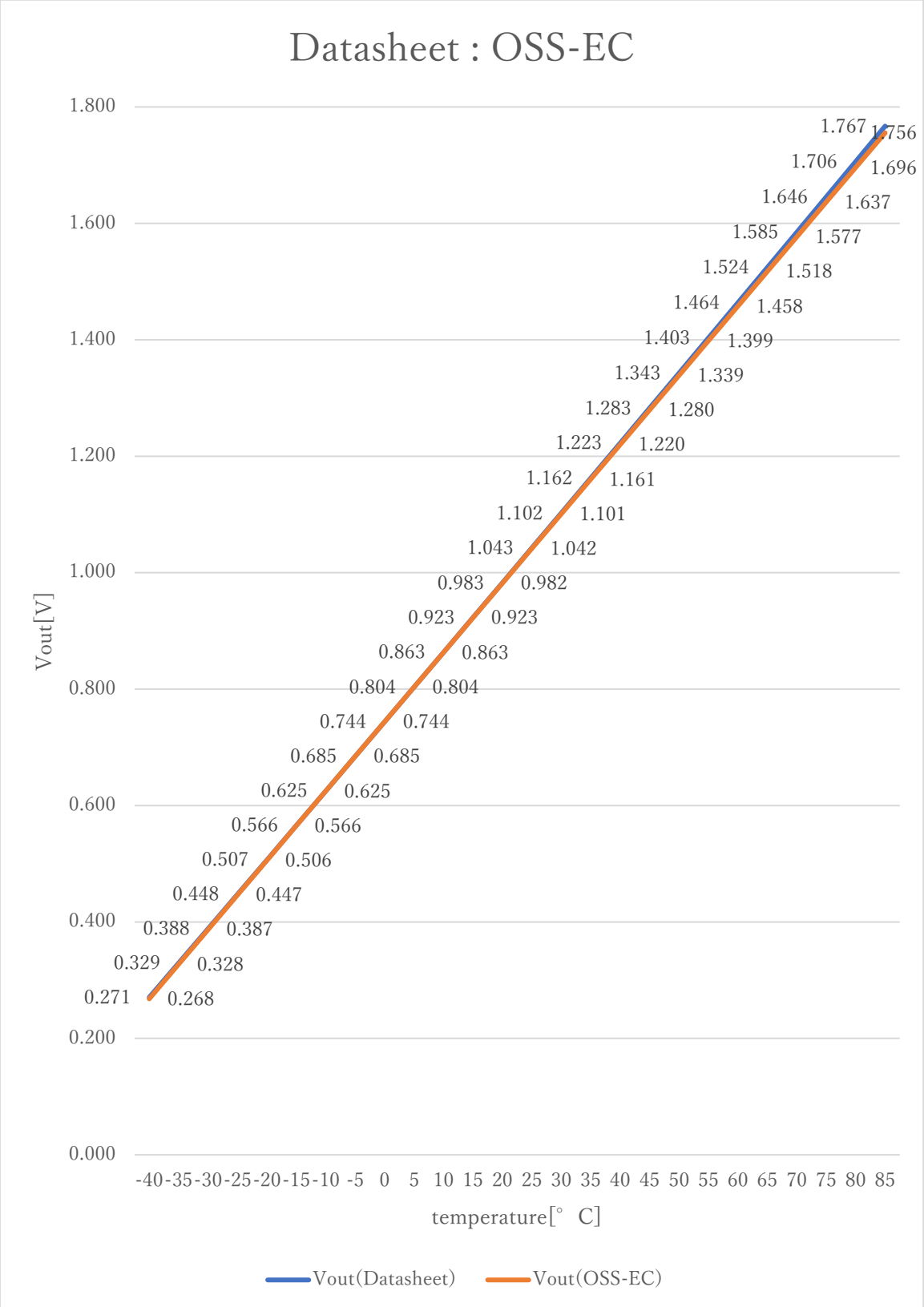
Voltage value to physical value conversion formula

$$y = (vi - iMAX6605MXKV\_xoff) / iMAX6605MXKV\_gain + iMAX6605MXKV\_yoff \quad [℃]$$

$$iMAX6605MXKV\_min \leqq y \leqq iMAX6605MXKV\_max$$

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 iMAX6605MXKV_xoff       0.744F          // X offset [V]
#define iMAX6605MXKV_yoff       0.0F            // Y offset [℃]
#define iMAX6605MXKV_gain       0.0119F         // Gain [V/℃]
#define iMAX6605MXKV_max        85.0F           // Temperature Max [℃]
#define iMAX6605MXKV_min        -40.0F          // Temperature Min [℃]
```



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

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   0.744F                  // X offset [V]
#define iMAX6605MXKV_yoff   0.0F                    // Y offset [℃]
#define iMAX6605MXKV_gain   0.0119F                 // Gain [V/℃]
#define iMAX6605MXKV_max    85.0F                   // Temperature Max [℃]
#define iMAX6605MXKV_min    -40.0F                  // Temperature Min [℃]

extern const tbl_adc_t tbl_MAX6605MXKV;
```

MAX6605MXKV.cpp

```
#include        "MAX6605MXKV.h"
#if     iMAX6605MXKV_ma == iSMA                         // Simple moving average filter
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
        0U ,                                            // 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
        0U ,                                            // 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

};
```