



Specification document of MAX6605MXK

Component manufacturer	Maxim Integrated
Model number	MAX6605MXK
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	2
2. Component Software IF specification	3
3. File Structure and Definitions	5

License

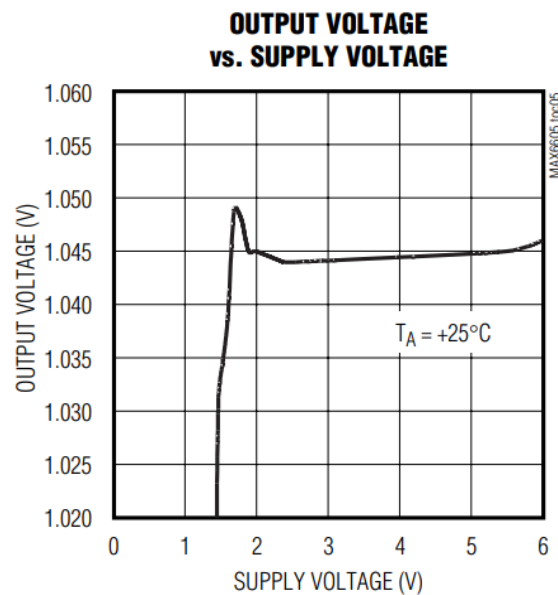
Open Source Software for Embedded Components ("OSS-EC") is open source software files and related documentation files for component products used in computer systems and other applications. OSS-EC is provided to those who accept the OSS-EC Terms of Use for the OSS-EC site; see https://oss-ec.com/license_agreement/ 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. We reserve the right to change these Terms of Use at any time and for any reason. We strongly recommend that you review the Terms of Use periodically.

1. Component datasheet

Temperature accuracy	$\pm 0.75^{\circ} \text{ C } (25^{\circ} \text{ C })$
Temperature range	$-55 \text{ to } +125^{\circ} \text{ C }$
Range of power supply voltage (Vdd)	2.7 to 5.5[V]
Output voltage (Vout)	Linear $11.9 \times \text{Vdd} / 3.3 \text{ [mV/}^{\circ} \text{ C] Typ.}$ Vdd = 3.3 [V] $0 [^{\circ} \text{ C}] \text{ } 0.744\text{[V] Typ.}$
Calculation	$\text{Vout} = 0.744\text{V} + (0.0119 \text{ V/}^{\circ} \text{ C} \times \text{Ta})$ $\text{Ta} = (\text{Vout} - 0.744\text{V}) / 0.0119 \text{ V/}^{\circ} \text{ C}$

More accurate temperature calculation

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



Applications

IoT etc

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

2. Component Software IF specification

The software interface specifications based on the MAX6605MXK 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

$$v_i = (a_i \times i_{ADC_vdd}) / 2^{i_{ADC_bit}} \quad [V]$$

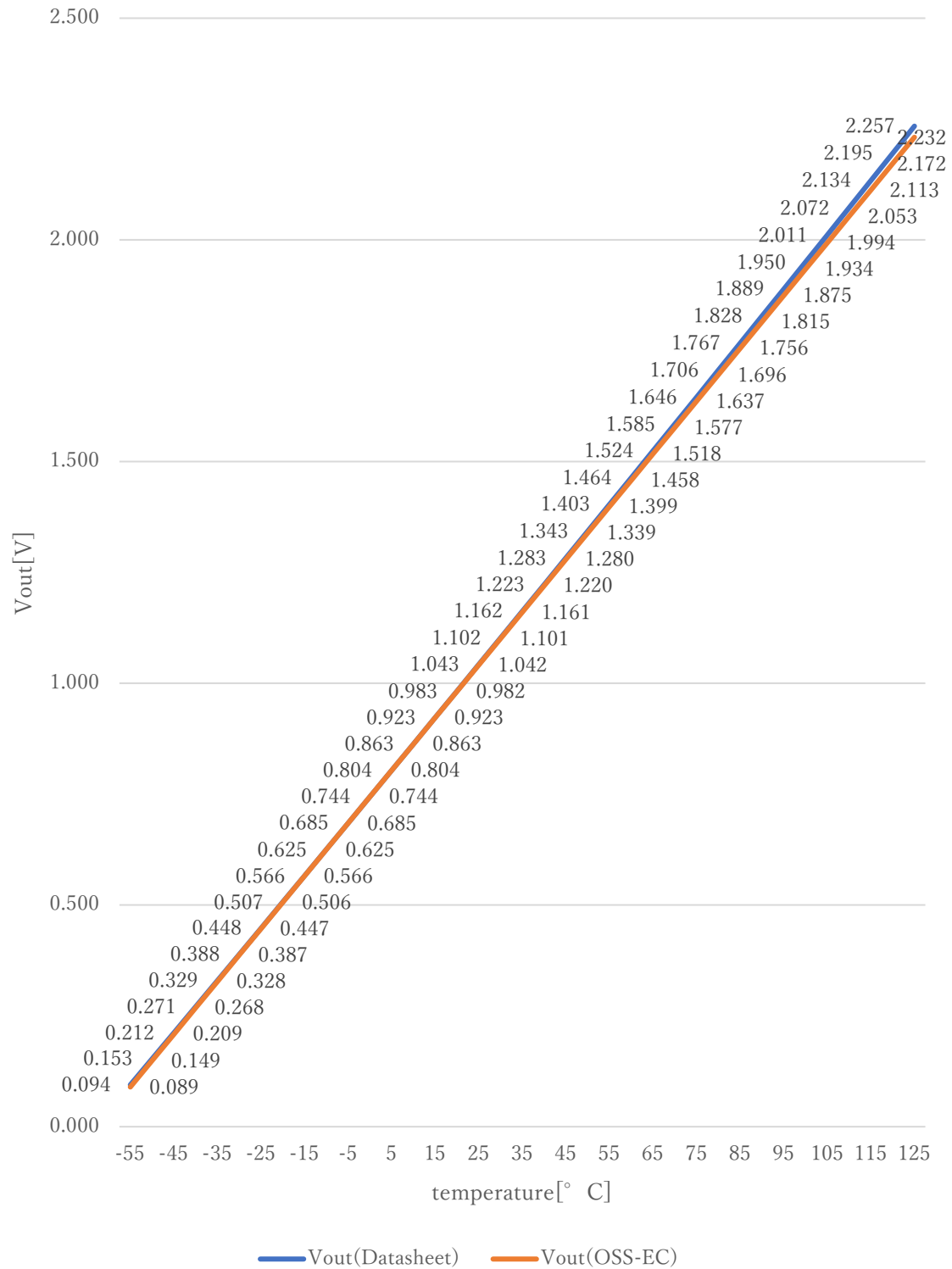
Voltage value to physical value conversion formula

$$y = (v_i - i_{MAX6605MXK_xoff}) / i_{MAX6605MXK_gain} + i_{MAX6605MXK_yoff} \quad [^{\circ}C]$$

$$i_{MAX6605MXK_min} \leq y \leq i_{MAX6605MXK_max}$$

a_i	A/D conversion value	
v_i	Sensor output voltage value [V]	
i_{ADC_vdd}	Sensor supply voltage value [V]	
i_{ADC_bit}	A/D conversion bit length	
y	Temperature value [$^{\circ}C$]	
<code>#define iMAX6605MXK_xoff</code>	<u>0.744F</u>	// X offset [V]
<code>#define iMAX6605MXK_yoff</code>	<u>0.0F</u>	// Y offset [$^{\circ}C$]
<code>#define iMAX6605MXK_gain</code>	<u>0.0119F</u>	// Gain [V/ $^{\circ}C$]
<code>#define iMAX6605MXK_max</code>	<u>125.0F</u>	// Temperature Max [$^{\circ}C$]
<code>#define iMAX6605MXK_min</code>	<u>-55.0F</u>	// Temperature Min [$^{\circ}C$]

Datasheet : OSS-EC



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

3. File Structure and Definitions

MAX6605MXK.h

```
#include "user_define.h"

// Components number
#define iMAX6605MXK      110U           // Maxim Integrated MAX6605MXK

// MAX6605MXK System Parts definitions
#define iMAX6605MXK_xoff  0.744F           // X offset [V]
#define iMAX6605MXK_yoff  0.0F             // Y offset [°C]
#define iMAX6605MXK_gain  0.0119F          // Gain [V/°C]
#define iMAX6605MXK_max   125.0F           // Temperature Max [°C]
#define iMAX6605MXK_min   -55.0F          // Temperature Min [°C]

extern const tbl_adc_t tbl_MAX6605MXK;
```

MAX6605MXK.cpp

```
#include "MAX6605MXK.h"

#if iMAX6605MXK_ma == iSMA // Simple moving average filter
static float32 MAX6605MXK_sma_buf[iMAX6605MXK_SMA_num];
static const sma_f32_t MAX6605MXK_Phy_SMA =
{
    iInitial , // Initial state
    iMAX6605MXK_SMA_num , // Simple moving average number & buf size
    0U , // buffer position
    0.0F , // sum
    &MAX6605MXK_sma_buf[0] // buffer
};

#elif iMAX6605MXK_ma == iEMA // Exponential moving average filter
static const ema_f32_t MAX6605MXK_Phy_EMA =
{
    iInitial , // Initial state
    0.0F , // Xn-1
    iMAX6605MXK_EMA_K // Exponential smoothing factor
};

#elif iMAX6605MXK_ma == iWMA // Weighted moving average filter
static float32 MAX6605MXK_wma_buf[iMAX6605MXK_WMA_num];
static const wma_f32_t MAX6605MXK_Phy_WMA =
{
    iInitial , // Initial state
    iMAX6605MXK_WMA_num , // Weighted moving average number & buf size
    0U , // buffer position
    iMAX6605MXK_WMA_num * (iMAX6605MXK_WMA_num + 1)/2 , // kn sum
    &MAX6605MXK_wma_buf[0] // Xn buffer
};

#else // Non-moving average filter
#endif

#define iDummy_adr 0xffffffff // Dummy address

const tbl_adc_t tbl_MAX6605MXK =
{
```

```

iMAX6605MXK
iMAX6605MXK_pin
iMAX6605MXK_xoff
iMAX6605MXK_yoff
iMAX6605MXK_gain
iMAX6605MXK_max
iMAX6605MXK_min
iMAX6605MXK_ma

#if iMAX6605MXK_ma == iSMA // Simple moving average filter
    &MAX6605MXK_Phy_SMA
    (ema_f32_t*) iDummy_adr
    (wma_f32_t*) iDummy_adr
#elif iMAX6605MXK_ma == iEMA // Exponential moving average filter
    (sma_f32_t*) iDummy_adr
    &MAX6605MXK_Phy_EMA
    (wma_f32_t*) iDummy_adr
#elif iMAX6605MXK_ma == iWMA // Weighted moving average filter
    (sma_f32_t*) iDummy_adr
    (ema_f32_t*) iDummy_adr
    &MAX6605MXK_Phy_WMA
#else // Non-moving average filter
    (sma_f32_t*) iDummy_adr
    (ema_f32_t*) iDummy_adr
    (wma_f32_t*) iDummy_adr
#endif

};

```