



Specification document of MAX6607IXK-T, MAX6608IUK-T

Component manufacturer	Maxim Integrated
Model number	MAX6607IXK-T, MAX6608IUK-T
Datasheets	MAX6607/08 DS (maximintegrated.com)
Specification Ver	01.00.00 Oct 09,2022 New release
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/

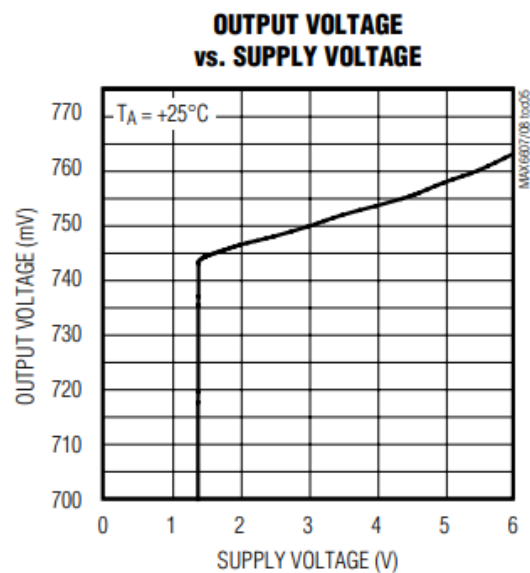
1. Component datasheet	2
2. Component Software IF specification	3
3. File Structure and Definitions	5

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	$\pm 2.0^{\circ} \text{ C}$ (Max, $+20$ to $+50^{\circ} \text{ C}$)
Temperature range	-20 to $+85^{\circ} \text{ C}$
Range of power supply voltage (Vdd)	1.8 to 3.6[V]
Output voltage (Vout)	Linear $10 \text{ [mV/^{\circ} C]}$ Typ. Vdd = 3.3 [V] $0 \text{ [^{\circ} C]} \text{ } 0.500\text{[V]}$ Typ.
Calculation	$V_{out} = 0.5V + (0.01 \text{ V/^{\circ} C} \times T_a)$ $T_a = (V_{out} - 0.5V) / 0.01 \text{ V/^{\circ} C}$



Applications

IoT etc

- Digital Cameras
- Battery Packs
- Portable Equipment
- GPS Equipmen

2. Component Software IF specification

The software interface specifications based on the MAX6607IXK-T, MAX6608IUK-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

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

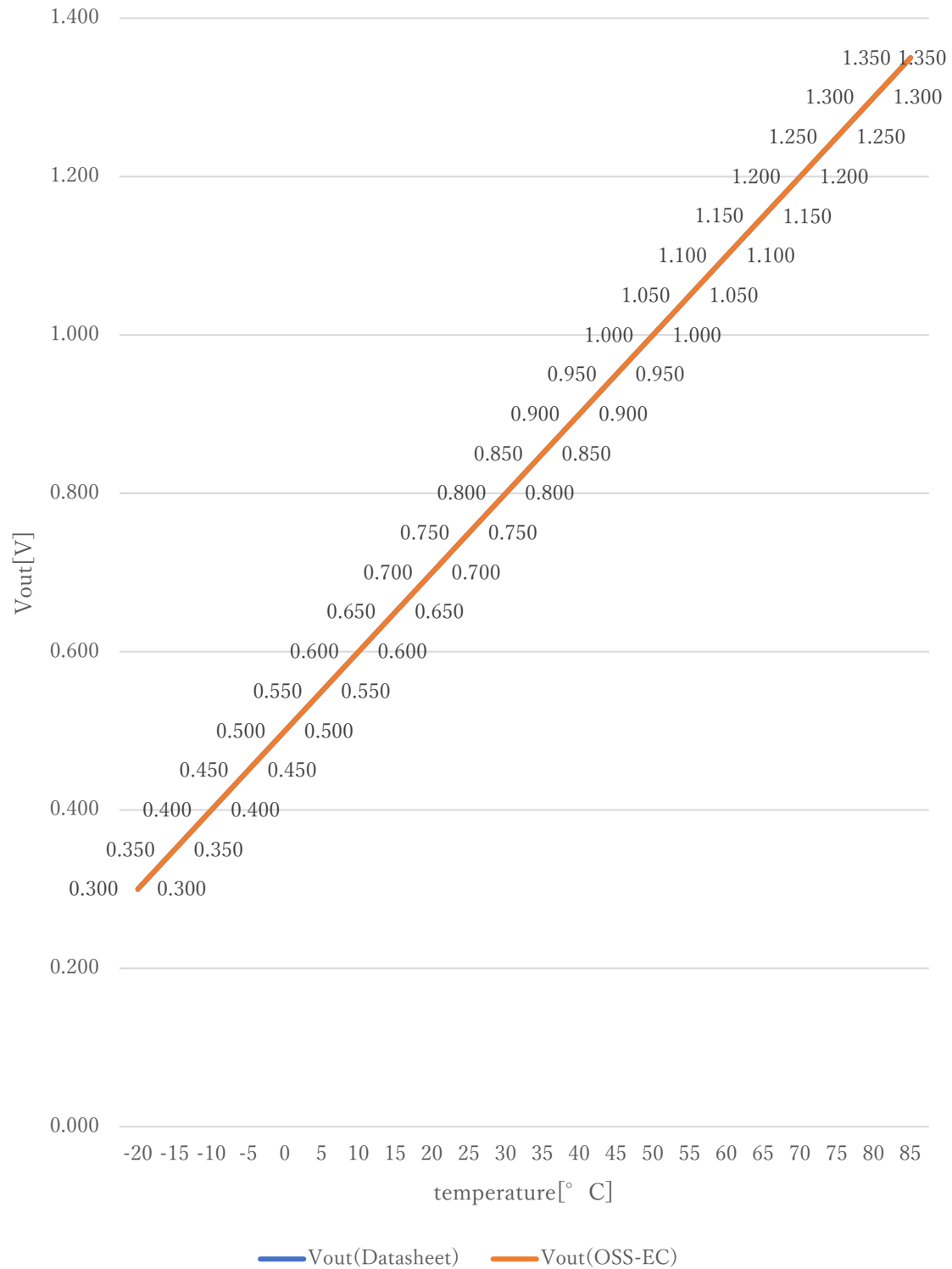
Voltage value to physical value conversion formula

$$y = (v_i - i_{MAX6607_xoff}) / i_{MAX6607_gain} + i_{MAX6607_yoff} \quad [^{\circ}C]$$

$$i_{MAX6607_min} \leq y \leq i_{MAX6607_max}$$

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 iMAX6607_xoff</code>	<u>0.5F</u>	// X offset [V]
<code>#define iMAX6607_yoff</code>	<u>0.0F</u>	// Y offset [$^{\circ}C$]
<code>#define iMAX6607_gain</code>	<u>0.01F</u>	// Gain [V/ $^{\circ}C$]
<code>#define iMAX6607_max</code>	<u>85.0F</u>	// Temperature Max [$^{\circ}C$]
<code>#define iMAX6607_min</code>	<u>-20.0F</u>	// Temperature Min [$^{\circ}C$]

Datasheet : OSS-EC



3. File Structure and Definitions

MAX6607.h

```
#include "user_define.h"

// Components number
#define iMAX6607      112U           // Maxim Integrated MAX6607IXK/MAX6608IUK

// MAX6607 System Parts definitions
#define iMAX6607_xoff    0.5F           // X offset [V]
#define iMAX6607_yoff    0.0F           // Y offset [°C]
#define iMAX6607_gain     0.01F          // Gain [V/°C]
#define iMAX6607_max      85.0F          // Temperature Max [°C]
#define iMAX6607_min      -20.0F         // Temperature Min [°C]

extern const tbl_adc_t tbl_MAX6607;
```

MAX6607.cpp

```
#include "MAX6607.h"

#if iMAX6607_ma == iSMA // Simple moving average filter
static float32 MAX6607_sma_buf[iMAX6607_SMA_num];
static const sma_f32_t MAX6607_Phy_SMA =
{
    iInitial , // Initial state
    iMAX6607_SMA_num , // Simple moving average number & buf size
    OU , // buffer position
    0.0F , // sum
    &MAX6607_sma_buf[0] // buffer
};

#elif iMAX6607_ma == iEMA // Exponential moving average filter
static const ema_f32_t MAX6607_Phy_EMA =
{
    iInitial , // Initial state
    0.0F , // Xn-1
    iMAX6607_EMA_K // Exponential smoothing factor
};

#elif iMAX6607_ma == iWMA // Weighted moving average filter
static float32 MAX6607_wma_buf[iMAX6607_WMA_num];
static const wma_f32_t MAX6607_Phy_WMA =
{
    iInitial , // Initial state
    iMAX6607_WMA_num , // Weighted moving average number & buf size
    OU , // buffer position
    iMAX6607_WMA_num * (iMAX6607_WMA_num + 1)/2 , // kn sum
    &MAX6607_wma_buf[0] // Xn buffer
};

#else // Non-moving average filter
#endif

#define iDummy_adr 0xffffffff // Dummy address

const tbl_adc_t tbl_MAX6607 =
{
```

```

        iMAX6607          ,
        iMAX6607_pin      ,
        iMAX6607_xoff     ,
        iMAX6607_yoff     ,
        iMAX6607_gain     ,
        iMAX6607_max      ,
        iMAX6607_min      ,
        iMAX6607_ma       ,

    #if    iMAX6607_ma == iSMA                                // Simple moving average filter
        &MAX6607_Phy_SMA      ,
        (ema_f32_t*) iDummy_adr ,
        (wma_f32_t*) iDummy_adr

    #elif  iMAX6607_ma == iEMA                                // Exponential moving average filter
        (sma_f32_t*) iDummy_adr ,
        &MAX6607_Phy_EMA      ,
        (wma_f32_t*) iDummy_adr

    #elif  iMAX6607_ma == iWMA                                // Weighted moving average filter
        (sma_f32_t*) iDummy_adr ,
        (ema_f32_t*) iDummy_adr ,
        &MAX6607_Phy_WMA

    #else                                           // Non-moving average filter
        (sma_f32_t*) iDummy_adr ,
        (ema_f32_t*) iDummy_adr ,
        (wma_f32_t*) iDummy_adr

    #endif

};

```