

Component manufacturer

# Specification document of STLM20W87F

Model number	STLM20W87F
Datasheets	Illtra-low current 2.4 V precision analog temperature sens

STMicroelectronics

Datasneets Ottra-low current 2.4 v precision analog temperature sensor

(st.com)

Specification Ver 01.00.00 Oct 20,2022 New release Documentation provided Rui Long Lab Inc. <a href="https://rui-long-lab.com/">https://rui-long-lab.com/</a>

1.	Component datasheet	2
	Component Software IF specification	
۷.	Component Software in Specification	
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  $\pm 1.5$  ° C maximum temperature accuracy

at 25  $^{\circ}$  C ( $\pm 0.5 ^{\circ}$  C typical)

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

Range of power supply voltage (Vdd) 2.4 to 5.5[V]

Output voltage ( Vout ) Linear  $-11.79 \text{ [mV/}^{\circ} \text{ C]} \text{ (-55 to } +130^{\circ} \text{ C)}$ Calculation  $Vout = 1.8528V + (-0.01179 \text{ V/}^{\circ} \text{ C} \times \text{Ta})$ 

 $Ta = (Vout - 1.8528V) / (-0.01179 V/^{\circ} C)$ 

Vdd vs Vout Non-link

Applications IoT etc

· Smartphones

· Multimedia PDA devices

· GPS devices

· Portable medical instruments

· Voltage-controlled crystal oscillator temperature monitors

• RF power transistor monitor

2



## 2. Component Software IF specification

The software interface specifications based on the STLM20W87F 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

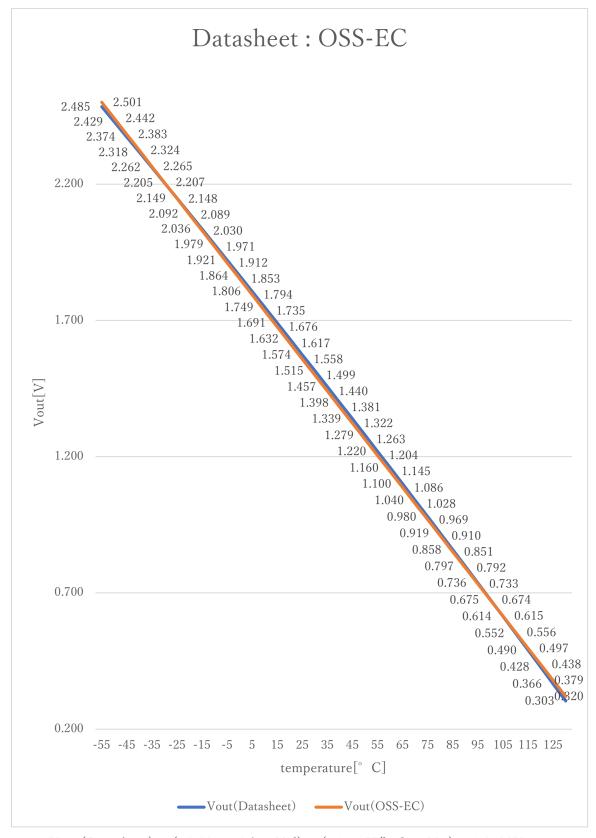
$$vi = (ai \times iADC_vdd) / 2^{iADC_bit}$$
 [V]

Voltage value to physical value conversion formula

```
y = (vi - iSTLM20W87F_xoff) / iSTLM20W87F_gain + iSTLM20W87F_yoff [°C] iSTLM20W87F_min \leq y \leq iSTLM20W87F_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]
#define iSTLM20W87F_xoff
                                                    // X offset [V]
                                   1.8528F
#define iSTLM20W87F_yoff
                                   <u>0. 0F</u>
                                                    // Y offset [°C]
                                                    // Gain [V/°C]
#define iSTLM20W87F_gain
                                   -0. 01179F
#define iSTLM20W87F_max
                                                    // Temperature Max [°C]
                                   130. OF
#define iSTLM20W87F_min
                                   -55. OF
                                                    // Temperature Min [°C]
```





 $Vout(Datasheet) = (-3.88 \times 10^{-6} \times Ta^2) + (-1.15 \text{ V/}^{\circ} \text{ C} \times Ta) + 1.8639 \text{V}$ 



#### 3. File Structure and Definitions

#### STLM20W87F.h

```
#include "user_define.h"
// Components number
#define iSTLM20W87F
                                                             // STMicroelectronics STLM20W87F
                              124U
// \ {\tt STLM20W87FSystem} \ {\tt Parts} \ {\tt definitions}
#define iSTLM20W87F_xoff
                              1.8528F
                                                             // X offset [V]
#define iSTLM20W87F_yoff
                              <u>0. 0F</u>
                                                             // Y offset [°C]
#define iSTLM20W87F_gain
                              <u>-0. 01179F</u>
                                                             // Gain [V/°C]
#define iSTLM20W87F_max
                              130.0F
                                                             // Temperature Max [°C]
#define iSTLM20W87F_min
                              <u>-55. 0F</u>
                                                             // Temperature Min [°C]
extern const tbl_adc_t tbl_STLM20W87F;
```



### STLM20DD9F.cpp

```
#include
                "STLM20W87F.h"
#if
        iSTLM20W87F_ma == iSMA
                                                        // Simple moving average filter
static float32 STLM20W87F_sma_buf[iSTLM20W87F_SMA_num];
static const sma_f32_t STLM20W87F_Phy_SMA =
        iInitial ,
                                                        // Initial state
        iSTLM20W87F_SMA_num ,
                                                      // Simple moving average number & buf size
        OU ,
                                                        // buffer position
        0.0F,
                                                        // sum
        &STLM20W87F_sma_buf[0]
                                                        // buffer
};
#elif
        iSTLM20W87F_ma == iEMA
                                                        // Exponential moving average filter
static const ema_f32_t STLM20W87F_Phy_EMA =
{
        iInitial ,
                                                        // Initial state
        0.0F,
                                                        // Xn-1
        iSTLM20W87F_EMA_K
                                                        // Exponential smoothing factor
};
#elif iSTLM20W87F_ma == iWMA
                                                        // Weighted moving average filter
static float32 STLM20W87F_wma_buf[iSTLM20W87F_WMA_num];
static const wma_f32_t STLM20W87F_Phy_WMA =
{
        iInitial ,
                                                        // Initial state
        iSTLM20W87F_WMA_num ,
                                                    // Weighted moving average number & buf size
                                                        // buffer poition
        OU ,
        iSTLM20W87F\_WMA\_num * (iSTLM20W87F\_WMA\_num + 1)/2,
                                                                 // kn sum
        &STLM20W87F_wma_buf[0]
                                                        // Xn buffer
};
#else
                                                        // Non-moving average filter
#endif
#define iDummy_adr
                         0xffffffff
                                                        // Dummy address
```



```
const tbl_adc_t tbl_STLM20W87F =
        iSTLM20W87F
        iSTLM20W87F_pin
        iSTLM20W87F_xoff
        iSTLM20W87F_yoff
        iSTLM20W87F_gain
        iSTLM20W87F_max
        iSTLM20W87F_min
        iSTLM20W87F_ma
#if
        iSTLM20W87F_ma == iSMA
                                                         // Simple moving average filter
        &STLM20W87F_Phy_SMA
        (ema_f32_t*) iDummy_adr,
        (wma_f32_t*) iDummy_adr
#elif
        iSTLM20W87F_ma == iEMA
                                                         // Exponential moving average filter
        (sma_f32_t*) iDummy_adr,
        &STLM20W87F_Phy_EMA
        (wma_f32_t*) iDummy_adr
#elif
        iSTLM20W87F_ma == iWMA
                                                          // Weighted moving average filter
        (sma_f32_t*) iDummy_adr ,
        (ema_f32_t*) iDummy_adr ,
        &STLM20W87F_Phy_WMA
#else
                                                          // Non-moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*) iDummy_adr ,
        (wma_f32_t*) iDummy_adr
#endif
};
```