Specification document of CHS-MSS

Component manufacturer	TDK
Model number	CHS-MSS
Datasheets	sensor_humidity_chs_en.pdf (tdk.com)
Specification Ver	01.00.00 Oct 20,2022 New release
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/

1. Component datasheet

Humidity accuracy	±5%RH	Edc=5V, 25° C, 20 to 85%RH
Humidity range	20% to 85% RH	
Range of power supply voltage (Vdd)	4.75 to 5.25[V]	
Output voltage (Vout)	Linear 10 [mV/%RH]	Edc=5V, 25° C, 20 to 85%RH
Calculation	Vout = 0.0V + (0.01 V/%RH × H)	
	H = (Vout – 0.0V) / (0.01 V/%RH)	
Vdd vs Vout	Non-link	

Applications IoT etc

- Refrigerators (condensation prevention)
- Air conditioners (indoor humidity control)
- PPCs, LBPs (image quality control)
- Industrial electronic humidity sensors, air conditioners for plant factories, etc.

2. Component Software IF specification

The software interface specifications based on the CHS-MSS 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

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

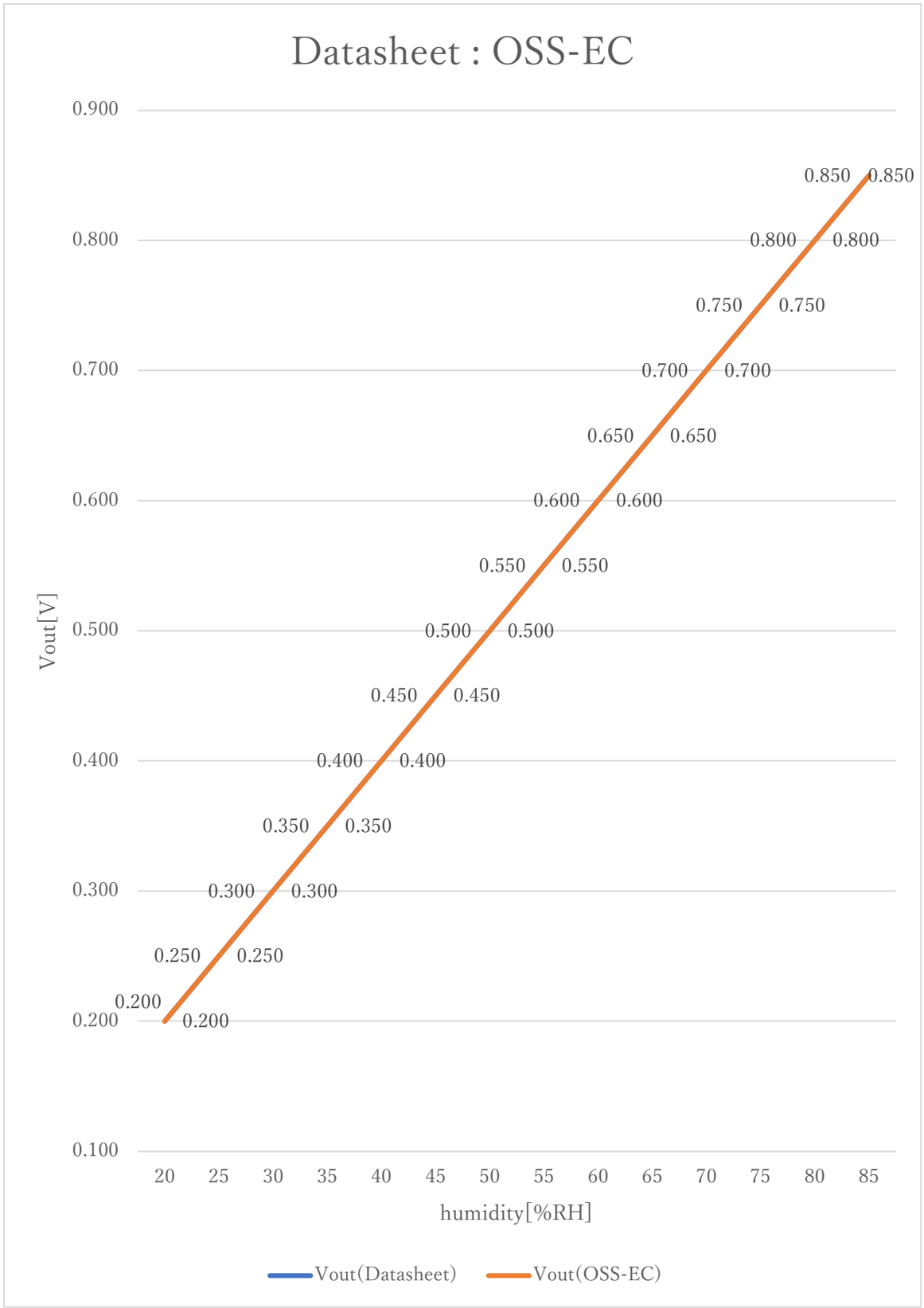
Voltage value to physical value conversion formula

$$y = (vi - iCHS\_MSS\_xoff) / iCHS\_MSS\_gain + iCHS\_MSS\_yoff \quad [\%RH]$$

$$iCHS\_MSS\_min \leqq y \leqq iCHS\_MSS\_max$$

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	Humidity value [%RH]

```
#define iCHS_MSS_xoff          0.0F          // X offset [V]
#define iCHS_MSS_yoff          0.0F          // Y offset [%RH]
#define iCHS_MSS_gain          0.01F         // Gain [V/%RH]
#define iCHS_MSS_max           85.0F         // Humidity Max [%RH]
#define iCHS_MSS_min           20.0F         // Humidity Min [%RH]
```


Datasheet : OSS-EC
0.900
0.800
0.700
0.600
0.500
0.400
0.300
0.200
0.100
Vout[V]
0.850 0.850
0.800 0.800
0.750 0.750
0.700 0.700
0.650 0.650
0.600 0.600
0.550 0.550
0.500 0.500
0.450 0.450
0.400 0.400
0.350 0.350
0.300 0.300
0.250 0.250
0.200 0.200
20 25 30 35 40 45 50 55 60 65 70 75 80 85
humidity[%RH]
Vout(Datasheet)
Vout(OSS-EC)

3. File Structure and Definitions

CHS_MSS.h

```
#include "user_define.h"

// Components number
#define iCHS_MSS            101U                    // TDK CHS-MSS

// CHS_MSS System Parts definitions
#define iCHS_MSS_xoff       0.0F                    // X offset [V]
#define iCHS_MSS_yoff       0.0F                    // Y offset [%RH]
#define iCHS_MSS_gain       0.01F                   // Gain [V/%RH]
#define iCHS_MSS_max        85.0F                   // Humidity Max [%RH]
#define iCHS_MSS_min        20.0F                   // Humidity Min [%RH]

extern const tbl_adc_t tbl_CHS_MSS;
```

CHS_MSS.cpp

```
#include        "CHS_MSS.h"
#if     iCHS_MSS_ma == iSMA                             // Simple moving average filter
static float32 CHS_MSS_sma_buf[iCHS_MSS_SMA_num];
static const sma_f32_t CHS_MSS_Phy_SMA =
{
        iInitial ,                                      // Initial state
        iCHS_MSS_SMA_num ,                          // Simple moving average number & buf size
        0U ,                                            // buffer position
        0.0F ,                                          // sum
        &CHS_MSS_sma_buf[0]                             // buffer
};
#elif   iCHS_MSS_ma == iEMA                             // Exponential moving average filter
static const ema_f32_t CHS_MSS_Phy_EMA =
{
        iInitial ,                                      // Initial state
        0.0F ,                                          // Xn-1
        iCHS_MSS_EMA_K                                  // Exponential smoothing factor
};
#elif   iCHS_MSS_ma == iWMA                             // Weighted moving average filter
static float32 CHS_MSS_wma_buf[iCHS_MSS_WMA_num];
static const wma_f32_t CHS_MSS_Phy_WMA =
{
        iInitial ,                                      // Initial state
        iCHS_MSS_WMA_num ,                        // Weighted moving average number & buf size
        0U ,                                            // buffer poition
        iCHS_MSS_WMA_num * (iCHS_MSS_WMA_num + 1)/2 ,   // kn sum
        &CHS_MSS_wma_buf[0]                             // Xn buffer
};
#else                                                   // Non-moving average filter
#endif

#define iDummy_adr      0xffffffff                      // Dummy address
```

```
const tbl_adc_t tbl_CHS_MSS =
{
        iCHS_MSS              ,
        iCHS_MSS_pin          ,
        iCHS_MSS_xoff         ,
        iCHS_MSS_yoff         ,
        iCHS_MSS_gain         ,
        iCHS_MSS_max          ,
        iCHS_MSS_min          ,
        iCHS_MSS_ma           ,

#if     iCHS_MSS_ma == iSMA                     // Simple moving average filter
        &CHS_MSS_Phy_SMA      ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#elif   iCHS_MSS_ma == iEMA                     // Exponential moving average filter
        (sma_f32_t*)iDummy_adr ,
        &CHS_MSS_Phy_EMA      ,
        (wma_f32_t*)iDummy_adr
#elif   iCHS_MSS_ma == iWMA                     // Weighted moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        &CHS_MSS_Phy_WMA
#else                                           // Non-moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#endif

};
```