Specification document of MPXH6400A

Component manufacturer	NXP Semiconductors
Model number	MPXH6400A
Datasheets	MPXH6400A, 20 to 400 kPa, Absolute, Integrated Pressure Sensor (nxp.com)
Specification Ver	01.00.00 Oct 18,2022 New release
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/

1. Component datasheet

Pressure range	20 to 400[kPa] 1.5% maximum error 0 to 85° C
Range of power supply voltage(Vdd)	4.64 to 5.36[V] 5.0[V]Typ.
Output voltage (Vout)	Vout = Vdd × (P × 0.002421 - 0.00842) ± Error
	Vdd =5.0[V]
	Temperature 0 to 85° C
	P = ((Vout / Vdd) + 0.00842) / 0.002421
Vdd vs Vout	link

Applications

IoT etc

- Industrial controls

Automotive

- Fuel injected car engines
- Vehicles powered by green gases (for example LPG and CNG)
- Small engines

2. Component Software IF specification

The software interface specifications based on the MPXH6400A 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

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

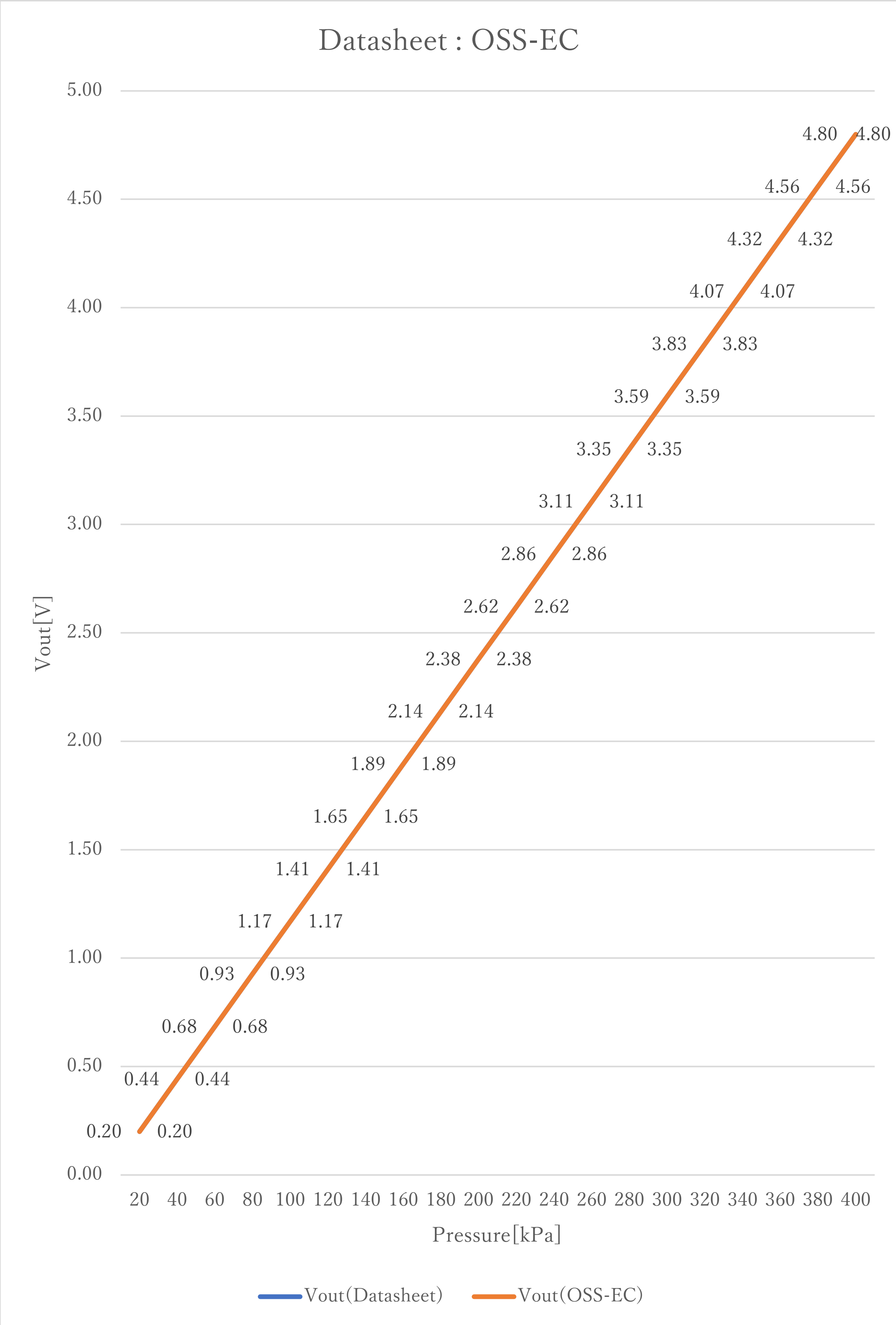
Voltage value to physical value conversion formula

$$y = (vi - iMPXH6400A\_xoff) / iMPXH6400A\_gain + iMPXH6400A\_yoff \quad [kPa]$$

$$iMPXH6400A\_min \leqq y \leqq iMPXH6400A\_max$$

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	Pressure value [kPa]

```
#define iMPXH6400A_xoff     ( -0.00842F*iADC_vdd )    // X offset [V]
#define iMPXH6400A_yoff     0.0F                      // Y offset [kPa]
#define iMPXH6400A_gain     ( 0.002421F*iADC_vdd )    // Gain [V/kPa]
#define iMPXH6400A_max      400.0F                    // Pressure Max [kPa]
#define iMPXH6400A_min      20.0F                     // Pressure Min [kPa]
```


Datasheet : OSS-EC
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
Vout[V]
0.20 0.20
0.44 0.44
0.68 0.68
0.93 0.93
1.17 1.17
1.41 1.41
1.65 1.65
1.89 1.89
2.14 2.14
2.38 2.38
2.62 2.62
2.86 2.86
3.11 3.11
3.35 3.35
3.59 3.59
3.83 3.83
4.07 4.07
4.32 4.32
4.56 4.56
4.80 4.80
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300 320 340 360 380 400
Pressure[kPa]
Vout(Datasheet)
Vout(OSS-EC)

3. File Structure and Definitions

MPXH6400A.h

```
#include "user_define.h"

// Components number
#define iMPXH6400A          121U                            // NXP MPXH6400A

// MPXH6400A System Parts definitions
#define iMPXH6400A_xoff     ( -0.00842F*iADC_vdd )          // X offset [V]
#define iMPXH6400A_yoff     0.0F                            // Y offset [kPa]
#define iMPXH6400A_gain     ( 0.002421F*iADC_vdd )          // Gain [V/kPa]
#define iMPXH6400A_max      400.0F                          // Pressure Max [kPa]
#define iMPXH6400A_min      20.0F                           // Pressure Min [kPa]

extern const tbl_adc_t tbl_MPXH6400A;
```

MPXH6400A.cpp

```
#include        "MPXH6400A.h"
#if     iMPXH6400A_ma == iSMA                          // Simple moving average filter
static float32 MPXH6400A_sma_buf[iMPXH6400A_SMA_num];
static const sma_f32_t MPXH6400A_Phy_SMA =
{
        iInitial ,                                     // Initial state
        iMPXH6400A_SMA_num ,                          // Simple moving average number & buf size
        0U ,                                           // buffer position
        0.0F ,                                         // sum
        &MMPXH6400A_sma_buf[0]                         // buffer
};
#elif   iMPXH6400A_ma == iEMA                          // Exponential moving average filter
static const ema_f32_t MPXH6400A_Phy_EMA =
{
        iInitial ,                                     // Initial state
        0.0F ,                                         // Xn-1
        iMPXH6400A_EMA_K                               // Exponential smoothing factor
};
#elif   iMPXH6400A_ma == iWMA                          // Weighted moving average filter
static float32 MPXH6400A_wma_buf[iMPXH6400A_WMA_num];
static const wma_f32_t MPXH6400A_Phy_WMA =
{
        iInitial ,                                     // Initial state
        iMPXH6400A_WMA_num ,                          // Weighted moving average number & buf size
        0U ,                                           // buffer poition
        iMPXH6400A_WMA_num * (iMPXH6400A_WMA_num + 1)/2 ,  // kn sum
        &MPXH6400A_wma_buf[0]                          // Xn buffer
};
#else                                                  // Non-moving average filter
#endif

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

```
const tbl_adc_t tbl_MPXH6400A =
{
        iMPXH6400A              ,
        iMPXH6400A_pin          ,
        iMPXH6400A_xoff         ,
        iMPXH6400A_yoff         ,
        iMPXH6400A_gain         ,
        iMPXH6400A_max          ,
        iMPXH6400A_min          ,
        iMPXH6400A_ma           ,

#if     iMPXH6400A_ma == iSMA                       // Simple moving average filter
        &MPXH6400A_Phy_SMA      ,
        (ema_f32_t*)iDummy_adr  ,
        (wma_f32_t*)iDummy_adr
#elif   iMPXH6400A_ma == iEMA                       // Exponential moving average filter
        (sma_f32_t*)iDummy_adr  ,
        &MPXH6400A_Phy_EMA      ,
        (wma_f32_t*)iDummy_adr
#elif   iMPXH6400A_ma == iWMA                       // Weighted moving average filter
        (sma_f32_t*)iDummy_adr  ,
        (ema_f32_t*)iDummy_adr  ,
        &MPXH6400A_Phy_WMA
#else                                               // Non-moving average filter
        (sma_f32_t*)iDummy_adr  ,
        (ema_f32_t*)iDummy_adr  ,
        (wma_f32_t*)iDummy_adr
#endif

};
```