Specification document of MPXH6115A

Component manufacturer	NXP Semiconductors
Model number	MPXH6115A
Datasheets	[MPXxx6115A, 15 to 115 kPa, Absolute, Integrated Presure 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	15 to 115[kPa] 1.5% maximum error 0 to 85° C
Range of power supply voltage(Vdd)	4.75 to 5.25[V] 5.0[V]Typ.
Output voltage (Vout)	$Vout = Vdd \times (P \times 0.009 - 0.095) \pm Error$
	$Vdd = 5.0[V]$
	Temperature 0 to 85° C
	$P = ((Vout / Vdd) + 0.095) / 0.009$
Vdd vs Vout	link

Applications IoT etc

- Industrial controls
- Weather station and weather reporting device barometers

Automotive

- Engine control/manifold absolute pressure (MAP)

2. Component Software IF specification

The software interface specifications based on the MPXH6115A 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

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

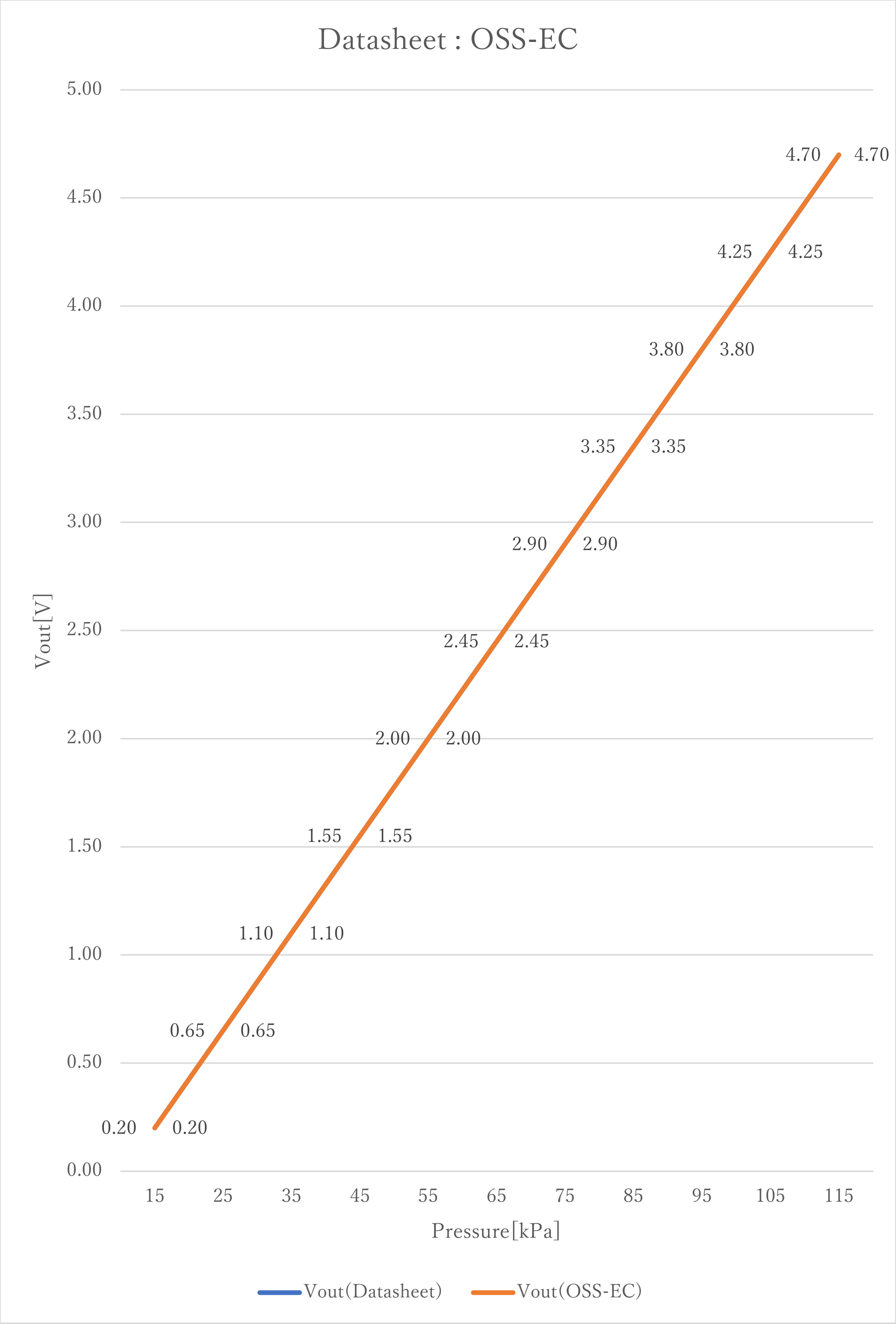
Voltage value to physical value conversion formula

$$y = (vi - iMPXH6115A\_xoff) / iMPXH6115A\_gain + iMPXH6115A\_yoff \quad [kPa]$$

$$iMPXH6115A\_min \leqq y \leqq iMPXH6115A\_max$$

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 iMPXH6115A_xoff          ( -0.095F*iADC_vdd )     // X offset [V]
#define iMPXH6115A_yoff          0.0F                     // Y offset [kPa]
#define iMPXH6115A_gain          ( 0.009F*iADC_vdd )      // Gain [V/kPa]
#define iMPXH6115A_max           115.0F                   // Pressure Max [kPa]
#define iMPXH6115A_min           15.0F                    // Pressure Min [kPa]
```


Datasheet : OSS-EC
Vout[V]
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
15
25
35
45
55
65
75
85
95
105
115
Pressure[kPa]
0.20
0.65
1.10
1.55
2.00
2.45
2.90
3.35
3.80
4.25
4.70
Vout(Datasheet)
Vout(OSS-EC)

3. File Structure and Definitions

MPXH6115A.h

```
#include "user_define.h"

// Components number
#define iMPXH6115A            119U                        // NXP MPXH6115A

// MPXH6115A System Parts definitions
#define iMPXH6115A_xoff       ( -0.095F*iADC_vdd )        // X offset [V]
#define iMPXH6115A_yoff       0.0F                        // Y offset [kPa]
#define iMPXH6115A_gain       ( 0.009F*iADC_vdd )         // Gain [V/kPa]
#define iMPXH6115A_max        115.0F                      // Pressure Max [kPa]
#define iMPXH6115A_min        15.0F                       // Pressure Min [kPa]

extern const tbl_adc_t tbl_MPXH6115A;
```

MPXH6115A.cpp

```
#include        "MPXH6115A.h"
#if     iMPXH6115A_ma == iSMA                           // Simple moving average filter
static float32 MPXH6115A_sma_buf[iMPXH6115A_SMA_num];
static const sma_f32_t MPXH6115A_Phy_SMA =
{
        iInitial ,                                      // Initial state
        iMPXH6115A_SMA_num ,                           // Simple moving average number & buf size
        0U ,                                            // buffer position
        0.0F ,                                          // sum
        &MPXH6115A_sma_buf[0]                           // buffer
};
#elif   iMPXH6115A_ma == iEMA                           // Exponential moving average filter
static const ema_f32_t MPXH6115A_Phy_EMA =
{
        iInitial ,                                      // Initial state
        0.0F ,                                          // Xn-1
        iMPXH6115A_EMA_K                                // Exponential smoothing factor
};
#elif   iMPXH6115A_ma == iWMA                           // Weighted moving average filter
static float32 MPXH6115A_wma_buf[iMPXH6115A_WMA_num];
static const wma_f32_t MPXH6115A_Phy_WMA =
{
        iInitial ,                                      // Initial state
        iMPXH6115A_WMA_num ,                         // Weighted moving average number & buf size
        0U ,                                            // buffer poition
        iMPXH6115A_WMA_num * (iMPXH6115A_WMA_num + 1)/2 , // kn sum
        &MPXH6115A_wma_buf[0]                           // Xn buffer
};
#else                                                   // Non-moving average filter
#endif

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

```
const tbl_adc_t tbl_MPXH6115A =
{
        iMPXH6115A              ,
        iMPXH6115A_pin          ,
        iMPXH6115A_xoff         ,
        iMPXH6115A_yoff         ,
        iMPXH6115A_gain         ,
        iMPXH6115A_max          ,
        iMPXH6115A_min          ,
        iMPXH6115A_ma           ,

#if     iMPXH6115A_ma == iSMA                           // Simple moving average filter
        &MPXH6115A_Phy_SMA      ,
        (ema_f32_t*)iDummy_adr  ,
        (wma_f32_t*)iDummy_adr
#elif   iMPXH6115A_ma == iEMA                           // Exponential moving average filter
        (sma_f32_t*)iDummy_adr  ,
        &MPXH6115A_Phy_EMA      ,
        (wma_f32_t*)iDummy_adr
#elif   iMPXH6115A_ma == iWMA                           // Weighted moving average filter
        (sma_f32_t*)iDummy_adr  ,
        (ema_f32_t*)iDummy_adr  ,
        &MPXH6115A_Phy_WMA
#else                                                   // Non-moving average filter
        (sma_f32_t*)iDummy_adr  ,
        (ema_f32_t*)iDummy_adr  ,
        (wma_f32_t*)iDummy_adr
#endif

};
```