Specification document of MPXA4250A

Component manufacturer	NXP Semiconductors		
Model number	MPXA4250A		
Datasheets	https://www.nxp.com/docs/en/data-sheet/MPX4250A.pdf		
Specification Ver	01.00.00	Aug 31,2022	New release
	01.00.01	Sep 10,2022	Corrected license URL
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/		

License

1. Component Software IF specification

The software interface specifications based on the MPXA4250A 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

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

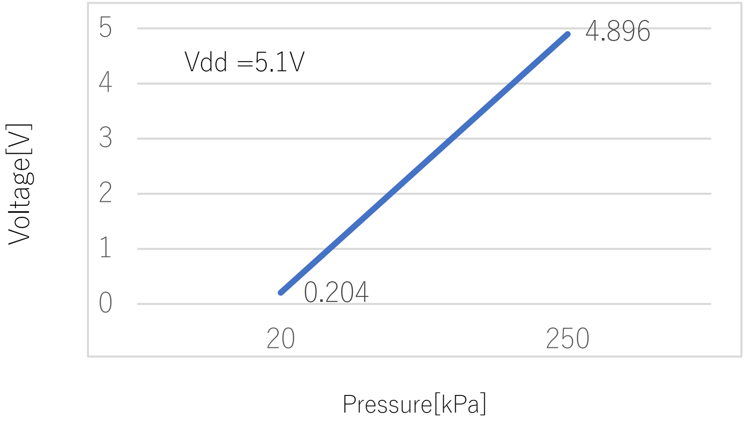
Voltage value to physical value conversion formula

$$y = (vi - iMPXA4250A\_xoff) / iMPXA4250A\_gain + iMPXA4250A\_yoff \quad [kPa]$$

$$iMPXA4250A\_min \leqq y \leqq iMPXA4250A\_max$$

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 iMPXA4250A_xoff        ( -0.04F*iADC_vdd )      // X offset [V]
#define iMPXA4250A_yoff        0.0F                     // Y offset [kPa]
#define iMPXA4250A_gain        ( 0.004F*iADC_vdd )      // Gain [V/kPa]
#define iMPXA4250A_max         250.0F                   // Pressure Max [kPa]
#define iMPXA4250A_min         20.0F                    // Pressure Min [kPa]
```



2. File Structure and Definitions

MPXA4250A.h

```
#include ″user_define.h″

// Components number
#define iMPXA4250A              100U                    // NXP MPXA4250A

// MPXA4250A System Parts definitions
#define iMPXA4250A_xoff         ( -0.04F*iADC_vdd )     // X offset [V]
#define iMPXA4250A_yoff         0.0F                    // Y offset [kPa]
#define iMPXA4250A_gain         ( 0.004F*iADC_vdd )     // Gain [V/kPa]
#define iMPXA4250A_max          250.0F                  // Pressure Max [kPa]
#define iMPXA4250A_min          20.0F                   // Pressure Min [kPa]

extern const tbl_adc_t tbl_MPXA4250A;
```

MPXA4250A.cpp

```
#include        "MPXA4250A.h"
#if     iMPXA4250A_ma == iSMA                          // Simple moving average filter
static float32 MPXA4250A_sma_buf[iMPXA4250A_SMA_num];
static const sma_f32_t MPXA4250A_Phy_SMA =
{
        iInitial ,                                     // Initial state
        iMPXA4250A_SMA_num ,                             // Simple moving average number & buf
size
        0U ,                                           // buffer position
        0.0F ,                                         // sum
        &MPXA4250A_sma_buf[0]                          // buffer
};
#elif   iMPXA4250A_ma == iEMA                          // Exponential moving average filter
static const ema_f32_t MPXA4250A_Phy_EMA =
{
        iInitial ,                                     // Initial state
        0.0F ,                                         // Xn-1
        iMPXA4250A_EMA_K                               // Exponential smoothing factor
};
#elif   iMPXA4250A_ma == iWMA                          // Weighted moving average filter
static float32 MPXA4250A_wma_buf[iMPXA4250A_WMA_num];
static const wma_f32_t MPXA4250A_Phy_WMA =
{
        iInitial ,                                     // Initial state
        iMPXA4250A_WMA_num ,                            // Weighted moving average number & buf
size
        0U ,                                           // buffer poition
        iMPXA4250A_WMA_num * (iMPXA4250A_WMA_num + 1)/2 , // kn sum
        &MPXA4250A_wma_buf[0]                          // Xn buffer
};
#else                                                  // Non-moving average filter
#endif

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

```
const tbl_adc_t tbl_MPXA4250A =
{
        iMPXA4250A              ,
        iMPXA4250A_pin          ,
        iMPXA4250A_xoff         ,
        iMPXA4250A_yoff         ,
        iMPXA4250A_gain         ,
        iMPXA4250A_max          ,
        iMPXA4250A_min          ,
        iMPXA4250A_ma           ,

#if     iMPXA4250A_ma == iSMA                           // Simple moving average filter
        &MPXA4250A_Phy_SMA      ,
        iDummy_adr              ,
        iDummy_adr
#elif   iMPXA4250A_ma == iEMA                           // Exponential moving average filter
        iDummy_adr              ,
        &MPXA4250A_Phy_EMA      ,
        iDummy_adr
#elif   iMPXA4250A_ma == iWMA                           // Weighted moving average filter
        iDummy_adr              ,
        iDummy_adr              ,
        &MPXA4250A_Phy_WMA
#else                                                   // Non-moving average filter
        iDummy_adr              ,
        iDummy_adr              ,
        iDummy_adr
#endif

};
```