Specification document of MCP9700, MCP9700A

Component manufacturer	Microchip Technology		
Model number	MCP9700, MCP9700A		
Datasheets	Low-Power Linear Active Thermistor ICs (microchip.com)		
Specification Ver	01.00.00	Oct 13,2022	New release
	01.00.01	Oct 18,2022	Corrected license content
Documentation provided	Rui Long Lab Inc. https://rui-long-lab.com/		

1. Component datasheet

Temperature accuracy	±4.0° C (Max, 0 to +70° C) MCP9700
	±2.0° C (Max, 0 to +70° C) MCP9700A
Temperature range	-40 to +125° C
Range of power supply voltage (Vdd)	2.3 to 5.5[V]
Output voltage (Vout)	Linear 0.01 [mV/° C] Typ.
	0 [° C] 0.5 [V] Typ.
Calculation	Vout = 0.5V + (0.01 V/° C × Ta)
	Ta = (Vout – 0.5V) / (0.01 V/° C)

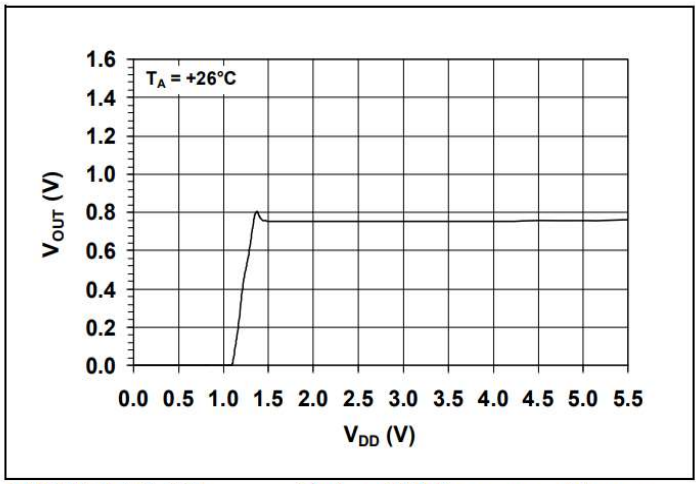


FIGURE 2-13: *Output Voltage vs. Power Supply.*

Applications

IoT etc

- Hard Disk Drives and Other PC Peripherals
- Entertainment Systems
- Home Appliance
- Office Equipment
- Battery Packs and Portable Equipment
- General Purpose Temperature Monitoring

2. Component Software IF specification

The software interface specifications based on the MCP9700, MCP9700A 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

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

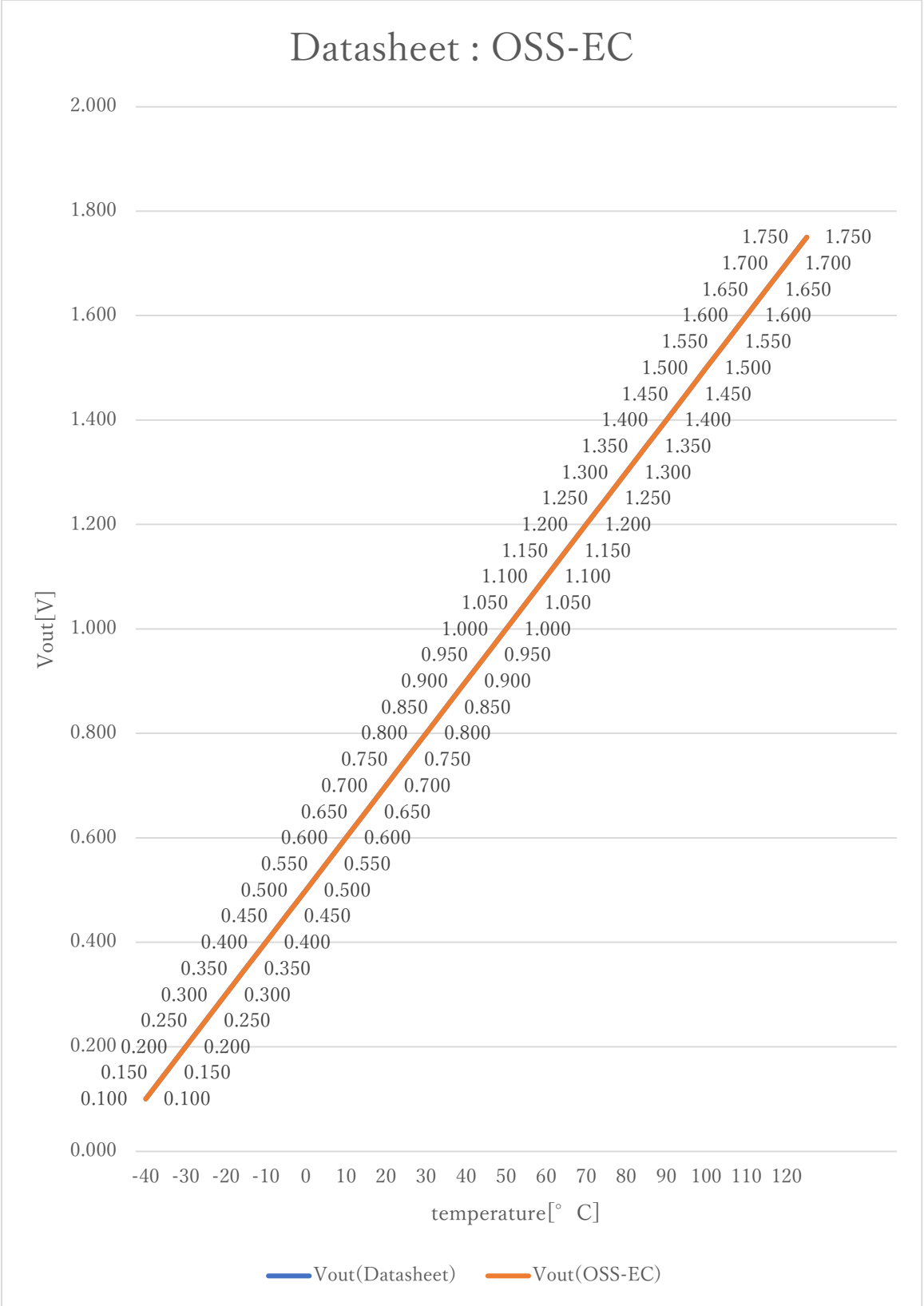
Voltage value to physical value conversion formula

$$y = (vi - iMCP9700\_xoff) / iMCP9700\_gain + iMCP9700\_yoff \quad [℃]$$

$$iMCP9700\_min \leqq y \leqq iMCP9700\_max$$

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	Temperature value [℃]

```
#define iMCP9700_xoff          0.5F          // X offset [V]
#define iMCP9700_yoff          0.0F          // Y offset [℃]
#define iMCP9700_gain          0.01F         // Gain [V/℃]
#define iMCP9700_max           125.0F        // Temperature Max [℃]
#define iMCP9700_min           -40.0F        // Temperature Min [℃]
```


Datasheet : OSS-EC
Vout[V]
2.000
1.800
1.600
1.400
1.200
1.000
0.800
0.600
0.400
0.200
0.000
-40 -30 -20 -10 0 10 20 30 40 50 60 70 80 90 100 110 120
temperature[° C]
Vout(Datasheet)
Vout(OSS-EC)

3. File Structure and Definitions

MCP9700.h

```
#include "user_define.h"

// Components number
#define iMCP9700            114U                    // Microchip Technology MCP9700, MCP9700A

// MCP9700 System Parts definitions
#define iMCP9700_xoff       0.5F                    // X offset [V]
#define iMCP9700_yoff       0.0F                    // Y offset [℃]
#define iMCP9700_gain       0.01F                   // Gain [V/℃]
#define iMCP9700_max        125.0F                  // Temperature Max [℃]
#define iMCP9700_min        -40.0F                  // Temperature Min [℃]

extern const tbl_adc_t tbl_MCP9700;
```

MCP9700.cpp

```
#include        "MCP9700.h"
#if     iMCP9700_ma == iSMA                             // Simple moving average filter
static float32 MCP9700_sma_buf[iMCP9700_SMA_num];
static const sma_f32_t MCP9700_Phy_SMA =
{
        iInitial ,                                      // Initial state
        iMCP9700_SMA_num ,                          // Simple moving average number & buf size
        0U ,                                            // buffer position
        0.0F ,                                          // sum
        &MCP9700_sma_buf[0]                             // buffer
};
#elif   iMCP9700_ma == iEMA                             // Exponential moving average filter
static const ema_f32_t MCP9700_Phy_EMA =
{
        iInitial ,                                      // Initial state
        0.0F ,                                          // Xn-1
        iMCP9700_EMA_K                                  // Exponential smoothing factor
};
#elif   iMCP9700_ma == iWMA                             // Weighted moving average filter
static float32 MCP9700_wma_buf[iMCP9700_WMA_num];
static const wma_f32_t MCP9700_Phy_WMA =
{
        iInitial ,                                      // Initial state
        iMCP9700_WMA_num ,                        // Weighted moving average number & buf size
        0U ,                                            // buffer poition
        iMCP9700_WMA_num * (iMCP9700_WMA_num + 1)/2 ,   // kn sum
        &MCP9700_wma_buf[0]                             // Xn buffer
};
#else                                                   // Non-moving average filter
#endif

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

```
const tbl_adc_t tbl_MCP9700 =
{
        iMCP9700                ,
        iMCP9700_pin            ,
        iMCP9700_xoff           ,
        iMCP9700_yoff           ,
        iMCP9700_gain           ,
        iMCP9700_max            ,
        iMCP9700_min            ,
        iMCP9700_ma             ,

#if     iMCP9700_ma == iSMA                     // Simple moving average filter
        &MCP9700_Phy_SMA        ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#elif   iMCP9700_ma == iEMA                     // Exponential moving average filter
        (sma_f32_t*)iDummy_adr ,
        &MCP9700_Phy_EMA        ,
        (wma_f32_t*)iDummy_adr
#elif   iMCP9700_ma == iWMA                     // Weighted moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        &MCP9700_Phy_WMA
#else                                           // Non-moving average filter
        (sma_f32_t*)iDummy_adr ,
        (ema_f32_t*)iDummy_adr ,
        (wma_f32_t*)iDummy_adr
#endif

};
```