Application Note

NOVOSENSE

NSC9260X DLL-based Calibration

AN-12-0014

Author: Juan Yu, Feifei Sun

AN-12-0014

NSC9260X DLL-based Calibration

NOVOSENSE

ABSTRACT

The document introduces how to do customized calibration platform based on NSC9260X DLLs and hardware provided by NOVOSENSE, including calibration algorithm introduction, DLL description, calibration example, firmware command summary and DAC calibration based on NOVOSENSE calibration hardware.

INDEX

1.Calibration Algorithm Introduction

1.1.Introduction to the stepping principle

NSC9260X signal chain is illustrated in Figure 1.1. Sensor output signal will be amplified through analog front end and will be quantized by PADC into a 24-bit digital output. Internal temperature sensor or external temperature sensor output will be amplified through auxiliary temperature measurement channel and quantized by TADC into a 24-bit temperature digital output. These two digital outputs will be calibrated by a built-in MCU and output to DAC to generate final analog voltage or current outputs. Calibration algorithm is embedded in MCU, and the corresponding calibration coefficients are stored in one 64-byte EEPROM. Calibration algorithm can be divided into ADC calibration (ADC CAL), sensor calibration (SENSOR_-CAL), range ratio adjustment, and DAC calibration (DAC CAL). ADC calibration is to calibrate zero point and sensitivity of PADC output and TADC output. Sensor calibration is to calibrate sensor offset, sensitivity, non-linearity, and temperature drift of offset and sensitivity. Range ratio adjustment supports user to modify the product's working range after module calibration is completed. DAC calibration is to calibrate zero point and sensitivity deviation of the analog output.

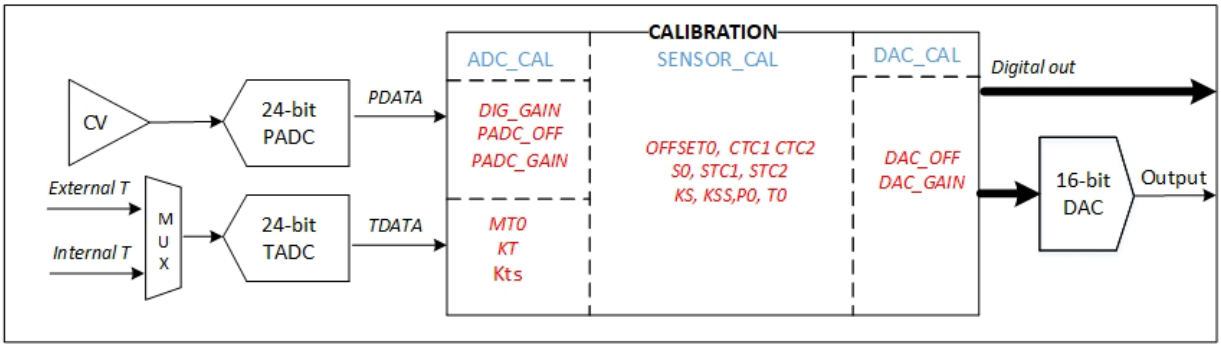


Figure 1.1 NSC9260X Signal Chain

NSC9260X can adjust zero and full range of the sensor. It can compensate up to 2nd order temperature coefficient for zero and full range respectively, and up to 3rd order nonlinear compensation according to following formulas.

$$OFFSET = OFFSET0 + CTC1 * (TDATA_{CAL} - T0) + CTC2 * (TDATA_{CAL} - T0)^2$$

$$S = S0 * (1 + STC1 * (TDATA_{CAL} - T0) + STC2 * (TDATA_{CAL} - T0)^2)$$

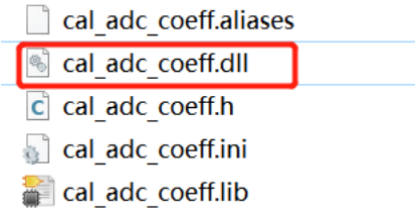
$$P_{NL} = (PDATA_{CAL1} - OFFSET) * S$$

$$PDATA_{CAL2} = P_{NL} + KS * P_{NL}^2 + KSS * P_{NL}^3 + P0$$

Table 1.1 Calibration Coefficient Description

Symbol	Descriptions	Range	Default	Data Format
PDATACAL2	Intermediate variable, calibrated linearity	(-2, 2)	---	---
OFFSET0	Sensor zero drift compensation	(-1, 1)	0	16-bit sign
CTC1	Zero 1st order temperature coefficient compensation	(-0.00781, 0.00781)	0	16-bit sign
CTC2	Zero 2nd order temperature coefficient compensation	(-6.1e-5, 6.1e-5)	0	16-bit sign
S0	Sensor sensitivity temperature coefficient	(0, 2)	0	16-bit unsigned
STC1	Sensitivity 1st order temperature coefficient compensation	(-0.00781, 0.00781)	0	16-bit sign
STC2	Sensitivity 2nd order temperature coefficient compensation	(-6.1e-5, 6.1e-5)	0	16-bit sign
KS	2nd order nonlinear coefficient of the sensor	(-1, 1)	0	16-bit sign
KSS	3rd order nonlinear coefficient of the sensor	(-0.5, 0.5)	0	16-bit sign
P0	Nonlinear calibration reference pressure value	(-1, 1)	0	8-bit sign

This document mainly introduces SENSOR_CAL and requires to use NSA2860_CAL_P.dll (contact NOVOSENSE for DLL) which based on method of least square.



Calibration mode refers to the following descriptions.

```
This program supports the following calibration modes
The general calibration formula is:
B'-B0 = (raw_P - (Off + tc1*(T-T0) + tc2*(T-T0)^2)) * s0 * (1 + ts1*(T-T0) + ts2*(T-T0)^2)
B-B0 = (B' - B0) + ks*(B'-B0)^2 + kss*(B'-B0)^3
1.  Off  s0                                                    //2P1T
2.  Off  s0  ks                                                //3P1T
3.  Off  s0  ks   kss                                          //4P1T
4.  Off  s0  tc1  ts1                                          //2P2T
5.  Off  s0  ks   tc1  ts1                                     //3P2T
6.  Off  s0  ks   kss  tc1  ts1                                //4P2T
7.  Off  s0  tc1  ts1  tc2  ts2                                //3P2T
8.  Off  s0  ks   tc1  ts1  tc2   ts2                          //3P3T
9.  Off  s0  ks   kss  tc1  ts1   tc2   ts2                    //4P3T
```

Figure 1.2 Calibration Modes

Collect all PDATA_raw (0xA5[0] = 1) for xPxT, then call NSA2860_CAL_P.dll and cal_adc_coeff.dll to calculate calibration coefficients and finally program them into EEPROM. After powering down and powering on again, coefficients in EEPROM will be loaded and sensor performance after calibration can be tested.

Table 1.2 Calibration Modes

Calibration Coefficients	T1				T2				T3			
OFFSET0, S0	P1	P2	--	--	--	--	--	--	--	--	--	--
OFFSET0, S0, KS	P1	P2	P3	--	--	--	--	--	--	--	--	--
OFFSET0, S0, KS, KSS	P1	P2	P3	P4	--	--	--	--	--	--	--	--
OFFSET0, S0, CTC1, STC1	P1	P2	--	--	P1	P2	--	--	--	--	--	--
OFFSET0, S0, CTC1, STC1, KS	P1	P2	P3	--	P1	P2	--	--	--	--	--	--
OFFSET0, S0, CTC1, STC1, KS, KSS	P1	P2	P3	P4	P1	P2	--	--	--	--	--	--
OFFSET0, S0, CTC1, CTC2, STC1, STC2	P1	P2	--	--	P1	P2	--	--	P1	P2	--	--
OFFSET0, S0, CTC1, CTC2, STC1, STC2, KS	P1	--	P3	--	P1	P2	P3	--	P1	--	P3	--
OFFSET0, S0, CTC1, CTC2, STC1, STC2, KS, KSS	P1	--	--	P4	P1	P2	P3	P4	P1	--	--	P4

2.DLL Library Description

Methods Called in DLL Library

调用方法：CalibrateP(Double[] PData, Double[] TData, Double[] RData, Double Tstand, Double Pstand, Int32 Psize, Int32 CalMode, Double ct1, Double ct2, Double st1, Double st2)
CalibrateP(Double[] PData, Double[] TData, Double[] RData, Double Tstand, Double Pstand, Int32 Psize, Int32 CalMode, Double ct1, Double ct2, Double st1, Double st2) (1D数组，内容为)

Table 2.1 DLL Input Descriptions

Parameters	Descriptions
Double [] PData	Enter normalized target pressure value (Ptarget/5V)
Double [] TData	Enter normalized target temperature value 1. Read TDATA register: 0x09, 0x0a, 0x0b (combined as 24-bit ADC_raw data, 2's complement) 2. Transfer TADC_raw [23: 0] to true code: If bit 23 is 1'b1, [true code = 0x1000000-TADC_raw]; if bit 23 is 1'b0, [true code = TADC_raw] 3. TData = (true code)/2^23*128
Double RData	Enter normalized pressure data after PDATA_cal 1. Read PDATA register: 0x06, 0x07, 0x08 (combined as 24-bit ADC_raw data, 2's complement) 2. Transfer PADC_raw [23:0] to true code: If bit 23 is 1'b1, [true code = 0x1000000-PADC_raw]; if bit 23 is 1'b0, [true code = PADC_raw] 3. RData = PDATACAL1 = (true code[23:0]/2^23 − PADC_OFF) ☒ (1 + PADC_GAIN) ☒ DIG_GAIN PADC_OFF/PADC_GAIN/DIG_GAIN is calculated by cal_adc_coeff.dll (refer to Table 2.3 and Figure 2.1)
Double Tstand	= 0
Double Pstand	= 0.5
Int32 Psize	Length of RData
Int32 CalMode	Different collection methods require different calibration modes. 1. Off s0 //2P1T 2. Off s0 ks //3P1T 3. Off s0 ks kss //4P1T 4. Off s0 tc1 ts1 //2P2T 5. Off s0 ks tc1 ts1 //3P2T 6. Off s0 ks kss tc1 ts1 //4P2T 7. Off s0 tc1 ts1 tc2 ts2 //2P3T 8. Off s0 ks tc1 ts1 tc2 ts2 //3P3T 9. Off s0 ks kss tc1 ts1 tc2 ts2 //4P3T
Double ct1	(1+PADC_GAIN) *DIG_GAIN*ct1
Double c2	(1+PADC_GAIN) *DIG_GAIN*c2
Double st1	(1+PADC_GAIN) *DIG_GAIN*st1
Double st2	(1+PADC_GAIN) *DIG_GAIN*st2

Table 2.2 PDATA Calibration Coefficients

Symbol	Descriptions	Range	Default	Data Format
PDATACAL1	Intermediate variable, output of PADC calibration	(-2, 2)	---	---
PADC_OFF	Offset calibration coefficient of P channel	(-1, 1)	0	24-bit sign
PADC_GAIN	Full-scale calibration coefficient of P channel	(-0.5, 0.5)	0	16-bit sign
DIG_GAIN	Digital gain of P channel	1, 2, 4, 8	1	2-bit

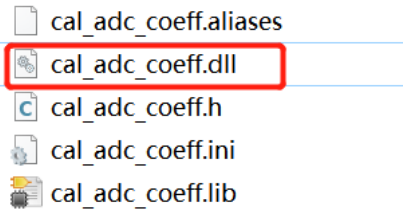


Figure 2.1 Cal_adc_coeff.dll

```
int32_t __cdecl CalADCCoeff(double targetData[], double rawData[],
    int32_t *DIG_GAIN, double *ADC_OFF, double *ADC_GAIN, int32_t len,
    int32_t len2);
```

Table 2.3 Cal_adc_coeff.dll Input Descriptions

Parameters	Descriptions
Double targetData []	Enter normalized target pressure value ((Ptarget/5V-0.5) *1.2) Normalize the target value to between [-0.6, +0.6]
Double rawData []	Enter normalized pressure raw data PDATA [23: 0]/2^23
Int32_t *DIG_GAIN	Output DIG_GAIN
Double *ADC_OFF	Output PADC_OFF
Double *ADC_GAIN	Output PADC_GAIN
Len	Input the length of targetData
Len2	Input the length of rawData

Note: It's better to do at least 3 iterations to obtain calibration coefficients. If convergence result cannot be obtained, iteration cycles can be added.

3.Specific Application Description

LabVIEW GUI Called NSA2860_CAL_P.dll Based on 3P1T Cal Mode at 25℃

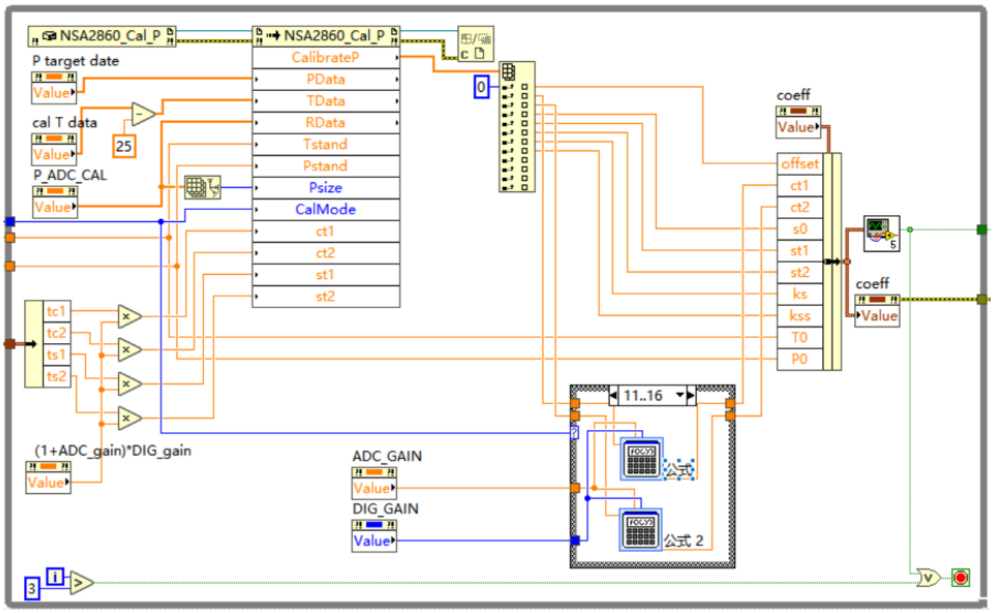


Figure 3.1 DLL Call VI Illustration

Collect raw data (pressure and temperature) under all pressure and temperature conditions (0xA5[0] = 1). For example, 6 sets of 24-bit data for 3P2T cal mode, 3 sets of 24-bit data for 3P1T cal mode (pressure P1_raw/P2_raw/P3_raw and temperature T1_raw/T2_raw/T3_raw/PADC_ OFF/PADC_ GAIN/DIG_GAIN). The calculation can refer to Figure 3.2.

$$(1 + \text{ADC\_GAIN}) * \text{DIG\_GAIN} = 2.11745$$

$$\text{Px\_cal} = (\text{Px\_raw} - \text{PADC\_OFF}) * (1 + \text{ADC\_GAIN}) * \text{DIG\_GAIN} = (\text{Px\_raw} - 0.226695) * 2.11745$$

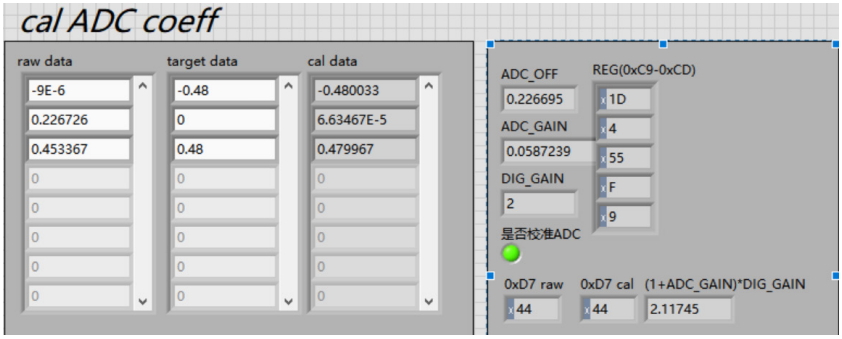


Figure 3.2 CALl cal ADC Coeff VI Illustration

Table 3.1 3P1T Related Data Descriptions

Parameters	Descriptions
Pressure1 (P1_target)	0.1 for target1: 0.5V (T1)
Pressure2 (P2_target)	0.5 for target2: 2.5V (T2)
Pressure3 (P3_target)	0.9 for target3: 4.5V (T3)
T1P1_P_raw	-9E-6 (P1_raw/2^23)
T1P2_P_raw	0.226726 (P2_raw/2^23)
T1P3_P_raw	0.453367 (P2_raw/2^23)
T1P1_T	0.002441 (T1_raw/2^23*128)
T1P2_T	0.002537 (T2_raw/2^23*128)
T1P3_T	0.002893 (T3_raw/2^23*128)
T1P1_P_cal	-0.480033 (P1_cal)
T1P2_P_cal	6.63467E-5 (P1_cal)
T1P3_P_cal	0.479967 (P1_cal)
cal_T1_data	T1P1_T*128+25 = 25.3124
cal_T2_data	T1P2_T*128+25 = 25.3247
cal_T3_data	T1P3_T*128+25 = 25.3703
Offset	6.55552E-5
S0	0.833347
Ks	0.000494655

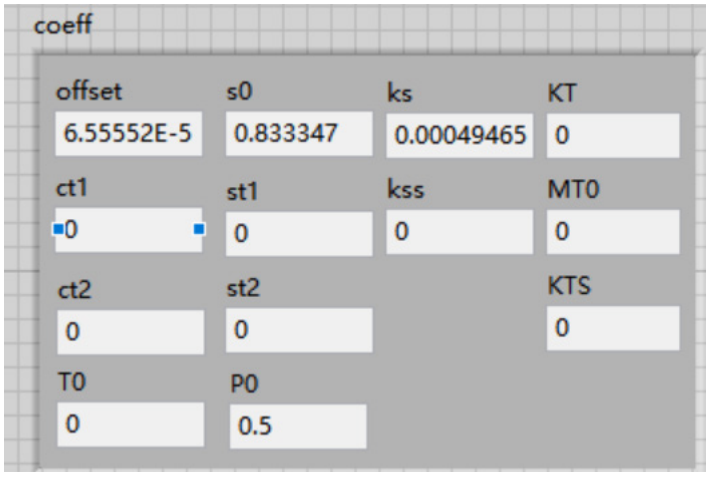


Figure 3.3 Calibration Coefficient VI

4.ASIC Only DAC Calibration

1.1.Introduction to the stepping principle

DAC_OFF/DAC_GAIN are used to calibrate offset, full scale of DAC and output driver. Calibration formula is shown as below. DAC_DATA (positive value) is 16-bit input of DAC. Error of analog output can be reduced after DAC calibration so that analog output can be calculated correctly based on PDATACAL. With internal low temperature drift reference voltage, DAC calibration can be done only at room temperature.

$$DAC\_DATA = (PDATACAL - DAC\_OFF) * (1 + DAC\_GAIN)$$

Table 4.1 DAC Calibration Coefficients

Symbol	Descriptions	Range	Default	Data Format
DAC_DATA	Calibrated input of DAC	(0, 1)	0	16-bit unsigned
DAC_OFF	Offset calibration coefficient of DAC	(-1, 1)	0	16-bit sign
DAC_GAIN	Full scale calibration coefficient of DAC	(-0.5, 0.5)	0	16-bit sign

5.Sensor Calibration Process

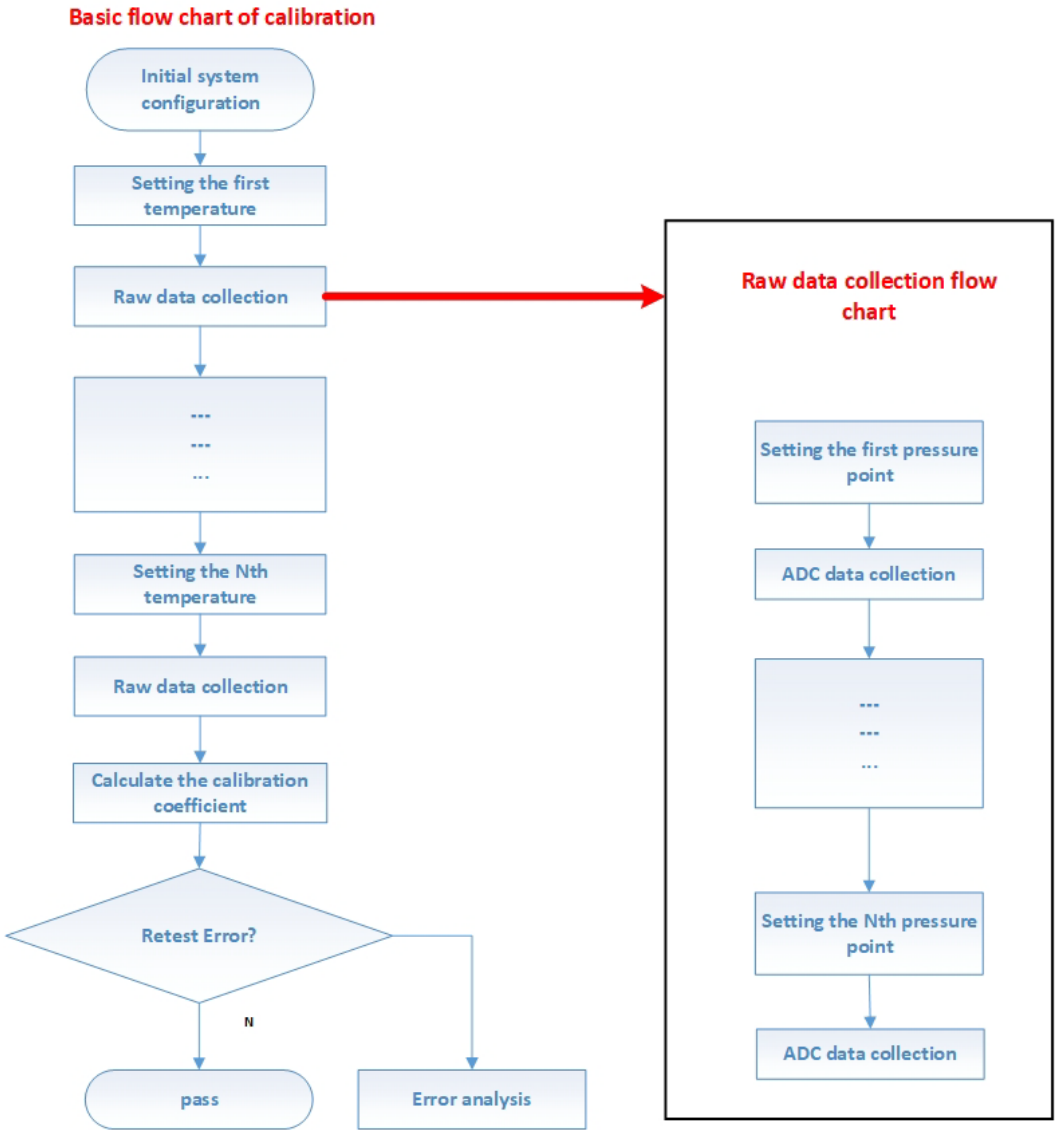


Figure 5.1 Sensor Calibration Process

6.Firmware Command List

1)Based on NOVOSENSE EVM Board

Command Format address byte 0xE1 is verified using Mark and subsequent data bytes are verified using Space											Command Descriptions
Transfer Direction	Command type	Byte 1	Byte 2 Byte length	Byte 3 Function Code	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Baud: 250kbps, stop bit: 1 bit, no flow control
Master -> Slave	Read Version	Dev-addr 0xE1	0x04	0x01	1	CRC (F6)					Return Frame E1 06 0100 087E
	Write/Read Register		Byte Length	0x16	Reg Addr	Reg Number	Data1	…	…	CRC	OWI Write
				0x17			0/1	CRC			OWI Read (0: 0~5V; 1: 4~20mA)
	IO operation		Byte Length	0x20	0x00/ 0x01						Power off then power on again
				0x21	0x0/0 x1						Byte4: 0x1: OWI on IO3; 0x0: Analog output on IO3
				0x22	0x00/ 0x01						0x01: Absolute output; 0x00: Proportional output
				0x23	CRC (0x18)						Switch IO to OWI, power off then power on and write KEY (0xB5A6C9) in corresponded OWI window to enter OWI communication
				0x24	0x00/ 0x01						Send command to read one DAC, 0x00 selects channel 1, 0x01 selects channel 2
				0x25	CRC						Read EVM board AD7799 data
	Digital quantity reading		Byte Length	0x32	CRC						OWI read continuously conversion data
				0x33							Enter OWI without power down

Command Format address byte 0xE1 is verified using Mark and subsequent data bytes are verified using Space											Command Descriptions
	Individual IO operations			0x41~0x4E	0x0/0x1	CRC					Control IO1~IOD to high (0x1)/low (0x0) levels
Slave -> Master	Return Frame Format	Dev-addr 0xE1	Length	0x01~0x4E Corre-sponding to send instruc-tions	0xF0/0xF1/0xFF/0x00	…	…	…	…	CRC	0xF0: CRC fail; 0xF1: FUNC fail; 0xFF: unpredict-able; 0x00: Normal operation

2)Based on NOVOSENSE 32CAL board

Command format (0x)										Command Descriptions
Transfor Direction	Byte 1 MCU-addr	Byte 2 Frame length	Byte 3 Function Code	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	…	Baud Rate is 250kbps, 8-bit data bit, no Parity bit, 1-bit stop bit, no flow control
PC -> MCU	10000000~11110000 Deter-mined by 4 MCU IO pin levels	0~255	0x00	CRC (0x5F)						Read MCU version
			0x01	n (1~32)	CRC					Choose channel n
			0x02	0x0/0x1/0x2	n (1~32)	CRC				Enter OWI for channel n only 0x0: 0~5V 0x1: 4~20mA 0x2: 0~10V
			0x03	0x0/0x1/0x2	CRC					Enter OWI for all 32 channels 0x0: 0~5V 0x1: 4~20mA 0x2: 0~10V
			0x04	n (1~32)	0x00/0x01	CRC				Measure analog output for channel n 0x00: Measure voltage 0x01: Measure current

Command format (0x)										Command Descriptions
			0x05	0x00/0x01						Measure analog outputs for all 32 channels 0x00: Measure voltage 0x01: Measure current
			0x06	0x00/0x01	CRC					0x00: analog output 0x01: digital CM
			0x07	0x00/0x01						0x00: ratio metric voltage output 0x01: absolute voltage output
			0x0b	Reg Addr	NO.	Data1	…	CRC		Write in OWI
			0x0c			0x0/ 0x1/ 0x2	CRC			Read in OWI 0x00: 0~5V 0x01: 4~20mA 0x02: 0~10V
			0x0f	Reg Addr	Data	CRC				Write same register for all 32 channels
			0x10	Reg Addr	0x0/ 0x1/ 0x2	CRC				Read same register for all 32 channels 0x00: 0~5V 0x01: 4~20mA 0x02: 0~10V
			0x11	0x00/0x01	CRC					Set different modes 0x00: 5V 0x01: 3.3V
			0x12	CRC						Read hardware version
			0x13							Check VDDE and GND shortage for all 32 channels
			0x31	0x00/0x01						Trim CAPOFF for all 32 channels, write final trim value to EEPROM (not program) 0x00: 0~5V 0x01: 4~20mA 0x02: 0~10V
MCU -> PC	10000000~ 11110000	(0~255)	0x00~0x31 Corresponding to sent instructions	0xF0/0xF1/ 0x00	Data1	Data2	Data3	Data4	CRC	0xF0: CRC fail; 0xF1: FUNC fail; 0x00: Normal Operation

Note: CRC calculation is done by users. Code example is as follows.

```
 unsigned char Cal_CRC (unsigned char *data, int len)
{
    unsigned char CRC = 0;
    unsigned char genPoly = 0x07;     //0x107 ,  x^8+x^2+x+1
    int i, j;
    for (I = 0; i<len; i++)
    {
     CRC^ = data[i];
     For (j = 0; j<8; j++)
  {
    If (CRC & 0x80) CRC = (CRC<<1) ^genPoly;
    else CRC<<=1;
  }
 }
    //CRC &=0xff;
    return CRC;
}
```

example for test:

input:

  data pointer: E1 04 23 (hex) len: 3

return:

  CRC: (1Byte): 0x18

7.Revision History

Revision	Description	Author	Date
1.0	Initial version	Juan Yu, Feifei Sun	2023/08/05

Sales Contact: sales@novosns.com; Further Information: www.novosns.com