M471 ADC in Power Noise Environment

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code uses M471 ADC averaging mode to get stable ADC conversion data in a power noise environment.
BSP Version	M471 Series BSP CMSIS V3.00.000
Hardware	NuMaker-M471KI V1.1

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Overview

The ADC applications need stable power source for analog circuit, but the power source is from mains supply in home appliance. You may get a noise in voltage source from mains supply like 60 Hz ripple waveform. The noise will make the ADC conversion result unstable. This sample code uses M471 averaging mode to average the ADC data. The noise error also will be averaged by M471 averaging mode so the ADC can get a stable data with less noise error. The voltage source in the noise environment is shown in Figure 1-1, where channel-1 is the power supply waveform and channel-2 is the ADC input waveform.

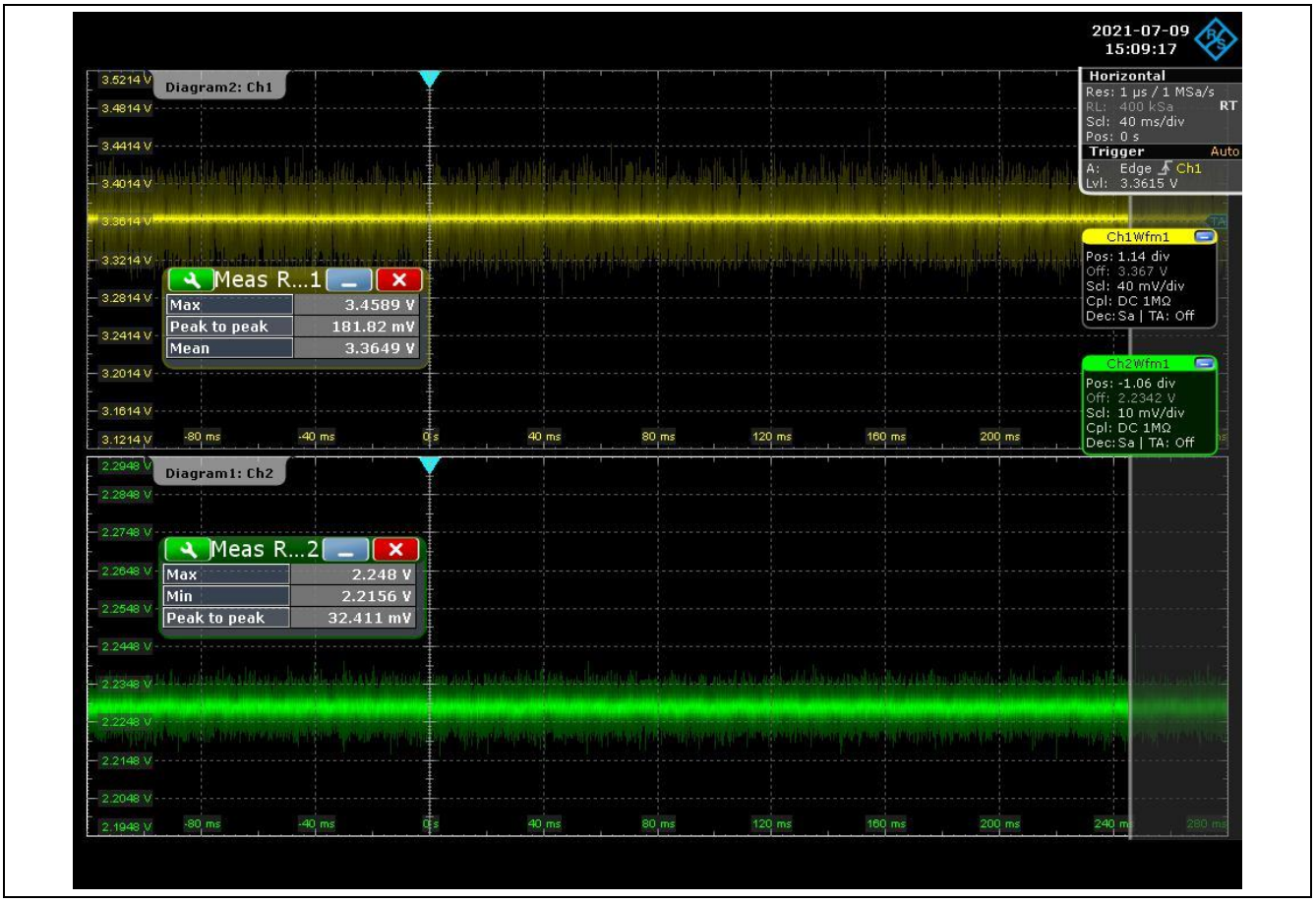

Figure 1-1 Voltage Source in Power Noise Environment

The ADC input waveform has a noise signal to affect the ADC conversion. The noise signal peak to peak is 32.4mV ((32.4mV/3.3V) * 4096 = 40 ADC data error). This will make the ADC data to float at the range 40 ADC error in the waveform amplitude. Using M471 averaging mode for getting more stable average ADC conversion results is better for ADC applications in a power noise environment.

1.1 Demo Result

The demo result is shown below.

```
ADC Total  Conversion Number = 10000
ADC First  Conversion Data   = 2454
ADC Second Conversion Data   = 2454
Max/Min/Avg/Dev          = 2457, 2453, 2454, 4
```

Figure 1-2 Demo Result

2 Code Description

After setting system clock and UART function, the Main function will call EADC_FunctionTest() function to start the Enhance ADC test.

```
int32_t main(void)
{
    /* Init System, IP clock and multi-function I/O. */
    SYS_Init();

    /* Init UART0 for printf */
    UART0_Init();

    /* EADC function test */
    EADC_FunctionTest();

    while(1);
}
```

The ADC_READ_COUNTER defines the ADC total conversion numbers for storage in g_u32ADCresult variable. The average_times is the setting for ADC averaging mode. If using “ave_times_8” to set averaging mode, the MCU will average 8 ADC conversion data to get ADC result.

```
/*---------------------------------------------------------------------------*/
/* Define global variables and constants                                     */
/*---------------------------------------------------------------------------*/
#define PLL_CLOCK       72000000
#define ADC_READ_COUNTER 10000
uint32_t g_u32ADCresult[ADC_READ_COUNTER];

enum average_times {
    ave_times_1 = 0,
    ave_times_2,
    ave_times_4,
    ave_times_8,
    ave_times_16,
    ave_times_32,
    ave_times_64,
    ave_times_128,
    ave_times_256
};
```

This part of EADC_FunctionTest() function is for setting the ADC and test the ADC averaging mode. The MCU will trigger the ADC 10000 times to sample voltage and calculate the maximum and the minimum value to show the ADC result is stable.

```
void EADC_FunctionTest()
{
    uint32_t u32Count,u32Max,u32Min,u32Dev,u32Sum,u32Avg;

    /* The Maximum EADC clock frequency is 36 MHz.
```

```
     * Hence, we set PLL to 72 MHz, HCLK 72 MHz and PCLK to 36 MHz.
     * EADC clock source is from PCLK = 36 MHz.
     */
    SYS_UnlockReg();
    /* Set input mode as single-end and enable the A/D converter */
    EADC_Open(EADC, EADC_CTL_DIFFEN_SINGLE_END);
    EADC_SetVRef(EADC, EADC_VREF_REFSEL_VREFP);

    // EADC Calibration
    EADC->CALSR = EADC_CALSR_CALIF_Msk;     // clear Calibration Finish interrupt flag
    EADC->CALCTL |= EADC_CALCTL_CAL_Msk;    // enable Calibration function
    EADC_START_CONV(EADC, BIT0 << 0);
    while((EADC->CALSR & EADC_CALSR_CALIF_Msk) == 0);   // wait Calibration Finish
    EADC->CALSR = EADC_CALSR_CALIF_Msk;    // clear Calibration Finish interrupt flag

    /* Configure sample module 7 for input channel 3 and enable ADINT0 trigger source */
    EADC_ConfigSampleModule(EADC, 7, EADC_ADINT0_TRIGGER, 3);
    EADC_ENABLE_SAMPLE_MODULE_INT(EADC, 0, BIT7);//Enable sample module 0 interrupt

     /* Average the ADC conversion data by averaging mode */
    EADC->MCTL1[7] = (ave_times_8 << EADC_MCTL1_ACU_Pos)| EADC_MCTL1_AVG_Msk;

    EADC_START_CONV(EADC, BIT7);
    for(u32Count=0; u32Count<ADC_READ_COUNTER; u32Count++) {
        while((EADC->STATUS2 & (BIT0<<0)) == 0);
        EADC_CLR_INT_FLAG(EADC, BIT0<<0);
        g_u32ADCresult[u32Count] = (EADC->DAT[7] & EADC_DAT_RESULT_Msk);
    }
    /* Disable EOC trigger */
    EADC_ConfigSampleModule(EADC, 7, EADC_SOFTWARE_TRIGGER, 3);

    for(u32Count=1; u32Count<ADC_READ_COUNTER; u32Count++) {
        printf("%d\n", g_u32ADCresult[u32Count]);
        u32Sum += g_u32ADCresult[u32Count];
    }

    u32Max = u32Min = g_u32ADCresult[1];
    for(u32Count=1; u32Count<ADC_READ_COUNTER; u32Count++) {
        if(g_u32ADCresult[u32Count] < u32Min)
            u32Min = g_u32ADCresult[u32Count];
        if(g_u32ADCresult[u32Count] > u32Max)
            u32Max = g_u32ADCresult[u32Count];
    }
    u32Dev = (u32Max-u32Min);
    u32Avg = u32Sum/(ADC_READ_COUNTER-1);

    printf("ADC Total  Conversion Number = %d\n",ADC_READ_COUNTER);
    printf("ADC First  Conversion Data   = %d\n",g_u32ADCresult[0]);
    printf("ADC Second Conversion Data   = %d\n",g_u32ADCresult[1]);
    printf("Max/Min/Avg/Dev          = %d,%d,%d,%d\n",u32Max,u32Min,u32Avg,u32Dev);
}
```

3 Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M471 Series BSP CMSIS V3.00.000
- IDE version
  - Keil uVersion 5.26

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M471KI V1.1

- Pin Connection

Use printf function to output message and the UART0 TX & RX pins are PB.13 and PB.12 respectively. Connect the ADC channel PB.3 with a voltage source to get ADC data.

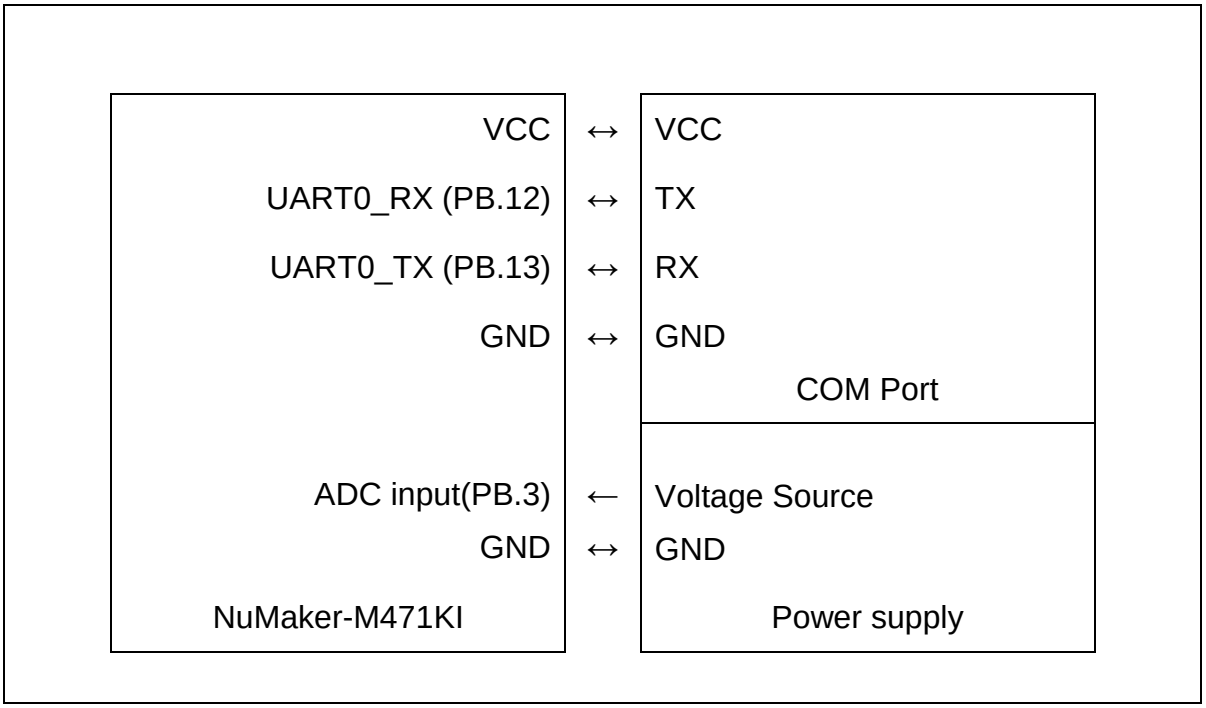

Figure 3-1 Pin Connection

4 Directory Information

The directory structure is shown below.

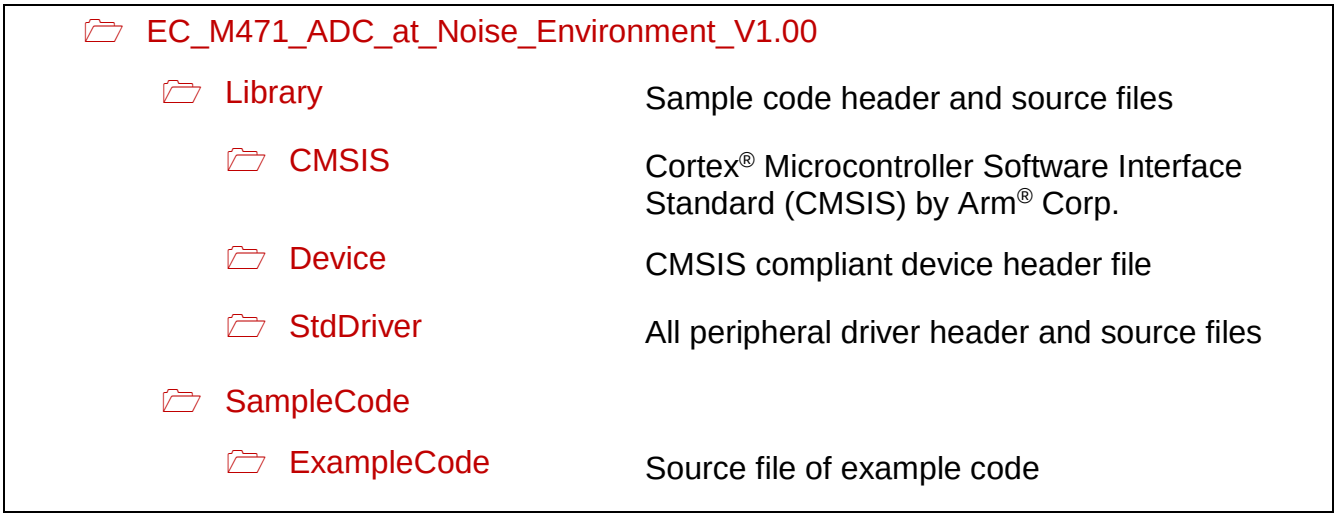

Figure 4-1 Directory Structure

5 Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *M471_ADC_in_Noise_Environment.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6 Revision History

Date	Revision	Description
2021.08.31	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.