Move EADC Data by PDMA

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code moves the EADC conversion result to RAM buffer by the PDMA.
BSP Version	M451 Series BSP CMSIS V3.01.003
Hardware	NuTiny-EVB-M451-LQFP100 V1.3

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 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 Function Description

1.1 Introduction

This sample code moves the EADC conversion result to RAM buffer by the PDMA. It demonstrates how to configure and trigger the PDMA to move the EADC conversion result. It also demonstrates how to trigger the EADC conversion by software or the interrupt of end of EADC conversion.

1.2 Principle

There are multiple sources to trigger the EADC to begin ADC conversion that include software trigger and the end of EADC conversion trigger. This sample code demonstrates how to trigger the EADC module 4 by software trigger and how to trigger the EADC module 5, 6, and 7 by the interrupt of end of EADC module 4 conversion.

There are multiple sources to trigger the PDMA to begin to move data that include the end of EADC conversion. This sample code demonstrates how to configure and trigger the PDMA to begin to move the EADC module 4, 5, 6, and 7 conversion result to RAM buffer.

1.3 Demo Result

```
System clock rate: 72000000 Hz
+------------------------------------------------------+
|    M453 Series EADC and PDMA Transfer Sample Code    |
+------------------------------------------------------+
PDMA interrupt !!
Conversion result of channel 0: 0xCB9, PDMA transfer data :0xCB9
Conversion result of channel 1: 0x7D9, PDMA transfer data :0x7D9
Conversion result of channel 2: 0x000, PDMA transfer data :0x000
Conversion result of channel 3: 0xFFF, PDMA transfer data :0xFFF
Conversion result of channel 0: 0xBC3, PDMA transfer data :0xBC3
Conversion result of channel 1: 0xA9F, PDMA transfer data :0xA9F
Conversion result of channel 2: 0x000, PDMA transfer data :0x000
Conversion result of channel 3: 0xFFF, PDMA transfer data :0xFFF
Exit EADC sample code
```

The key point is that the conversion result of EADC channels must exact same to the PDMA transfer data.

2 Code Description

Configure the EADC module 4 trigger source to software trigger and analog input channel to channel 0. Configure the EADC module 5, 6, and 7 trigger source to the interrupt of end of module 4 conversion and analog input channels to channel 1, 2, and 3.

```
    /* Set the ADC internal sampling time, input mode as single-end and enable the A/D
converter */
    EADC_Open(EADC, EADC_CTL_DIFFEN_SINGLE_END);
    EADC_SetInternalSampleTime(EADC, 6);


    /* Configure the sample 4 module for analog input channel 0 and enable software trigger
source */
    EADC_ConfigSampleModule(EADC, 4, EADC_SOFTWARE_TRIGGER, 0);
    /* Configure the sample 5 module for analog input channel 1 and enable ADINT0 trigger
source */
    EADC_ConfigSampleModule(EADC, 5, EADC_ADINT0_TRIGGER, 1);
    /* Configure the sample 6 module for analog input channel 2 and enable ADINT0 trigger
source */
    EADC_ConfigSampleModule(EADC, 6, EADC_ADINT0_TRIGGER, 2);
    /* Configure the sample 7 module for analog input channel 3 and enable ADINT0 trigger
source */
    EADC_ConfigSampleModule(EADC, 7, EADC_ADINT0_TRIGGER, 3);


    /* Clear the A/D ADINT0 interrupt flag for safe */
    EADC_CLR_INT_FLAG(EADC, EADC_STATUS2_ADIF0_Msk);
    /* Enable the interrupt 0 */
    EADC_ENABLE_INT(EADC, BIT0);
    /* Enable sample module 4 to use interrupt 0 */
    EADC_ENABLE_SAMPLE_MODULE_INT(EADC, 0, (BIT0 << 4));


    /* Clear the A/D ADINT1 interrupt flag for safe */
    EADC_CLR_INT_FLAG(EADC, EADC_STATUS2_ADIF1_Msk);
    /* Enable the interrupt 1 */
    EADC_ENABLE_INT(EADC, BIT1);
    /* Enable sample module 7 to use interrupt 1 */
    EADC_ENABLE_SAMPLE_MODULE_INT(EADC, 1, (BIT0 << 7));


    NVIC_EnableIRQ(ADC01_IRQn);
```

Configure the PDMA to move 8 EADC conversion results. Each conversion result is 32 bites width:

```
    /* Reset PDMA module */
    SYS_ResetModule(PDMA_RST);
    /* Open PDMA channel */
    PDMA_Open(BIT0 << PDMA_CH);
    /* PDMA transfer width is one word (32 bit) and transfer count 8 */
    PDMA_SetTransferCnt(PDMA_CH, PDMA_WIDTH_32, 8);
    /* PDMA transfer data from ADC to buffer g_au32DestArray*/
    PDMA_SetTransferAddr(PDMA_CH, (uint32_t) & (EADC->CURDAT), PDMA_SAR_FIX,
(uint32_t)g_au32DestArray, PDMA_DAR_INC);
    PDMA_SetTransferMode(PDMA_CH, PDMA_ADC_RX, FALSE, 0);


    PDMA_SetBurstType(PDMA_CH, PDMA_REQ_SINGLE, PDMA_BURST_128);


    g_u32IsPDMATestOver = 0;
    PDMA_EnableInt(PDMA_CH, PDMA_INT_TRANS_DONE);
    NVIC_EnableIRQ(PDMA_IRQn);


    PDMA_Trigger(PDMA_CH);


    /* Enable PDMA transfer */
    EADC_ENABLE_PDMA(EADC);
```

Use the software trigger to trigger the EADC module 4 to begin to convert its input voltage. After the module 4 conversion completed, the interrupt of end of conversion will trigger the EADC module 5, 6, and 7 to begin to convert their input voltages.

```
/* Reset the ADC indicator and trigger sample module 4 to start A/D conversion */
    g_u32AdcIntFlag = 0;
    g_u32IsADCChannel7Over = 0;

    /* Start EADC module 4 conversion */
    EADC_START_CONV(EADC, (BIT0 << 4));

    /* Wait EADC interrupt (g_u32AdcIntFlag will be set at IRQ_Handler function) */
    while (g_u32IsADCChannel7Over == 0);

    /* Get the conversion result of the sample module 4 ~ 7 */
    for (u32ModuleCount = 4, u32SampleCount = 0;
            u32ModuleCount < 8;
            u32ModuleCount++, u32SampleCount++)
        g_au32ConversionData[u32SampleCount] = EADC_GET_CONV_DATA(EADC, u32ModuleCount);
```

We must trigger the EADC module 4 twice to generate 8 EADC conversion results because the PDMA had been configured to move 8 conversion results.

3 Software and Hardware Environment

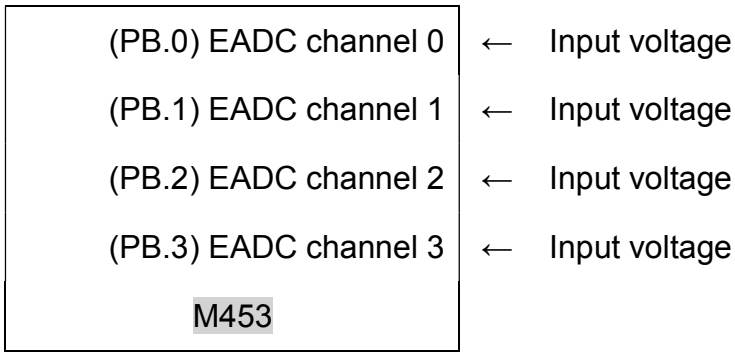
Software Environment

- BSP version
  - M451 Series BSP CMSIS V3.01.003
- IDE version
  - Keil uVersion 4.74

Hardware Environment

- Circuit components
  - NuTiny-EVB-M451-LQFP100 V1.3
- Diagram

M453 uses channel 0 as the input voltage of the EADC module 4, uses channel 1 as the input voltage of the EADC module 5, uses channel 2 as the input voltage of the EADC module 6, uses channel 3 as the input voltage of the EADC module 7. The connection between M453 and input voltages as below:



4 Directory Information

- EC_M453_Move_EADC_Data_by_PDMA_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click M453_Move_EADC_Data_by_PDMA.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
Sep. 16, 2019	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.