M480 ADC to UART BY PDMA

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	M480 ADC to UART BY PDMA Sample Code.
BSP Version	M480 Series BSP CMSIS V3.04.000
Hardware	NuMaker-ETM-M487

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

Transmit ADC data to UART by PDMA without CPU.

1.2 Principle

This example code can check the voltage data by UART without CPU intervention.

1.3 Demo Result

After execute the program, UART will output the ADC voltage.

2 Code Description

Get ADC data and send data out by PWM and DAC

```
void IDLE_UARTFunction(void)
{

    /*---------------------------------------------------------------------------------*/
    /* Init UART1                                                                      */
    /*---------------------------------------------------------------------------------*/
    /* Reset UART module */
    SYS_ResetModule(UART1_RST);
    UART_Open(UART1, 115200);
    UART1->INTEN |= UART_INTEN_TXPDMAEN_Msk;

    /*---------------------------------------------------------------------------------*/
    /* Init EPWM0                                                                      */
    /*---------------------------------------------------------------------------------*/
    /* Reset EPWM0 module */
    SYS_ResetModule(EPWM0_RST);

    /* Set EPWM0 timer clock prescaler */
    EPWM_SET_PRESCALER(EPWM0, 0, 0);

    /* Set up counter type */
    EPWM0->CTL1 &= ~EPWM_CTL1_CNTTYPE0_Msk;

    // EPWM0 channel 0 frequency is 10Hz, duty 30%,
    EPWM_ConfigOutputChannel(EPWM0, 0, 10, 30);

    /* EPWM period point trigger ADC enable */
    EPWM_EnableADCTrigger(EPWM0, 0, EPWM_TRG_ADC_EVEN_PERIOD);

    /* Set output level at zero, compare up, period(center) and compare down of specified
channel */
    EPWM_SET_OUTPUT_LEVEL(EPWM0, BIT0, EPWM_OUTPUT_HIGH, EPWM_OUTPUT_LOW,
EPWM_OUTPUT_NOTHING, EPWM_OUTPUT_NOTHING);

    /* Enable output of EPWM0 channel 0 */
    EPWM_EnableOutput(EPWM0, BIT0);

    /*---------------------------------------------------------------------------------*/
```

```
    /* Init EADC                                                              */
    /*------------------------------------------------------------------------*/
    /* Set input mode as single-end and enable the A/D converter */
    EADC_Open(EADC, EADC_CTL_DIFFEN_SINGLE_END);

    /* Configure the sample module 0 for analog input channel 2 and enable EPWM0 trigger
source */
    EADC_ConfigSampleModule(EADC, 0, EADC_PWM0TG0_TRIGGER, 2);

    EADC->CTL &= 0xFFFFF3F;
    EADC->CTL |= 1<<6;

    EADC_ENABLE_PDMA(EADC);

    /* Enable EPWM0 channel 0 counter */
    EPWM_Start(EPWM0, BIT0); /* EPWM0 channel 0 counter start running. */

    /*------------------------------------------------------------------------*/
    /* Init PDMA                                                              */
    /*------------------------------------------------------------------------*/
    /* Open Channel 1, 4 */
    PDMA_Open(PDMA, 1 << 1);

    /* Enable Scatter Gather mode, assign the first scatter-gather descriptor table is
table 1,
       and set transfer mode as memory to peripheral */
    PDMA_SetTransferMode(PDMA, 1, PDMA_EADC0_RX, TRUE, (uint32_t)&DMA_DESC[0]);


    /* Scatter-Gather descriptor table configuration in SRAM */
    g_u32DMAConfig = \
                     (0 << PDMA_DSCT_CTL_TXCNT_Pos) | /* Transfer count is 1 */ \
                     PDMA_WIDTH_32 |  /* Transfer width is 32 bits(one word) */ \
                     PDMA_SAR_FIX |   /* Source increment size is fixed(no increment) */ \
                     PDMA_DAR_FIX |   /* Destination increment size is fixed(no increment)
*/ \
                     PDMA_REQ_SINGLE | /* Transfer type is burst transfer type */ \
                     PDMA_BURST_4 |   /* Burst size is 128. No effect in single transfer
type */ \
                     PDMA_OP_SCATTER; /* Operation mode is scatter-gather mode */

    /*-----------------------------------------------------------
                          Scatter Gather mode table 1
```

```
        ------------------------------------------------------------------------*/

    DMA_DESC[0].ctl = g_u32DMAConfig;
    /* Configure source address */
    DMA_DESC[0].src = EADC_BASE; /* Ping-Pong buffer 1 */
    /* Configure destination address */
    DMA_DESC[0].dest = ((uint32_t) (&g_u8Tx_Buffer[0]));
    /* Configure next descriptor table address */
    DMA_DESC[0].offset = (uint32_t)&DMA_DESC[1] - (PDMA->SCATBA); /* next operation table
is table 2 */

    /*------------------------------------------------------------------------
                          Scatter Gather mode table 2
        ------------------------------------------------------------------------*/

    DMA_DESC[1].ctl = g_u32DMAConfig;
    /* Configure source address */
    DMA_DESC[1].src = ((uint32_t) (&g_u8Tx_Buffer[0])); /* Ping-Pong buffer 1 */
    /* Configure destination address */
    DMA_DESC[1].dest = ((uint32_t) (&g_u8Rx_Buffer[0]));
    /* Configure next descriptor table address */
    DMA_DESC[1].offset = (uint32_t)&DMA_DESC[2] - (PDMA->SCATBA); /* next operation table
is table 2 */

    /*------------------------------------------------------------------------
                          Scatter Gather mode table 3
        ------------------------------------------------------------------------*/

    DMA_DESC[2].ctl = g_u32DMAConfig;
    /* Configure source address */
    DMA_DESC[2].src = ((uint32_t) (&g_u8Rx_Buffer[0])); /* Ping-Pong buffer 1 */
    /* Configure destination address */
    DMA_DESC[2].dest = UART1_BASE;
    /* Configure next descriptor table address */
    DMA_DESC[2].offset = 0; /* next operation table is table 2 */

    /* Enable transfer done interrupt */
    PDMA_EnableInt(PDMA, 1, PDMA_INT_TRANS_DONE);
    NVIC_EnableIRQ(PDMA_IRQn);

    printf("\n\nCPU @ %dHz\n", SystemCoreClock);
```

```
    printf("+------------------------------------------------------------------------+
\n");
    printf("|    M480 PDMA Driver Ping-Pong Buffer Sample Code (Scatter-gather)     |
\n");
    printf("+------------------------------------------------------------------------+
\n");



}
```

3 Hardware and Software environment

- **Software Environment**
  - BSP version
    - M480 Series BSP CMSIS V3.04.000
  - IDE version
    - Keil uVersion 5.26
- **Hardware Environment**
  - Circuit components
    - NuMaker-ETM-M487
  - Diagram
    -

4 Directory Information

- EC_M480_ADC_to_UART_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 a sample code

1. Enter Keil compile mode
    a. Build
    b. Download
    c. Start/Stop debug session
2. Enter debug mode
    a. Run

6 Revision History

Date	Revision	Description
Nov 14, 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.