Adjust PWM Output by ADC Result

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Adjust PWM duty cycle by ADC conversion result.
BSP Version	NUC230/240 Series BSP v3.01.002
Hardware	NUC240VE3AN NuEdu Board V2.0 + NuEdu-Basic01 V2.0

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

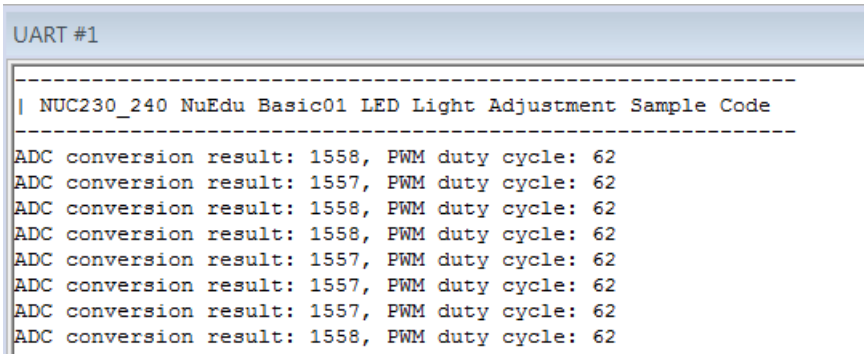
This sample demonstrates to change LED light while user adjusts the knob on NuEdu-Basic01 board.

1.2 Principle

When user turns the knob on NuEdu-Basic01 board, the resistor value is changed. When the resistor value is changed then the voltage of circuit is also changed. MCU uses ADC to sample the voltage, and using the conversion result to calculate PWM duty cycle. Then the LED light is also changed.

1.3 Demo Result

User can enter debug mode of Keil MDK and open UART#1 window. The execution result shown as below figure.



2 Code Description

The sample code is as follows.
Enable crystals. Enable ADC module and PWMA module clock, select its clock source. Configure the MFP function of used pins in SYS_Init().
Configure the MFP function of pins as follows:

- PA.0 as ADC channel 0
- PA.12 as PWMA channel 0

```
void SYS_Init(void)
{
    /*---------------------------------------------------------------------------------------------------------*/
    /* Init System Clock                                                                                       */
    /*---------------------------------------------------------------------------------------------------------*/

    /* Enable Internal RC clock */
    CLK_EnableXtalRC(CLK_PWRCON_OSC22M_EN_Msk);

    /* Waiting for IRC22M clock ready */
    CLK_WaitClockReady(CLK_CLKSTATUS_OSC22M_STB_Msk);

    /* Switch HCLK clock source to Internal RC and HCLK source divide 1 */
    CLK_SetHCLK(CLK_CLKSEL0_HCLK_S_HIRC, CLK_CLKDIV_HCLK(1));

    /* Enable ADC module clock */
    CLK_EnableModuleClock(ADC_MODULE);

    /* Enable PWM module clock */
    CLK_EnableModuleClock(PWM01_MODULE);

    /* ADC clock source is 22.1184MHz, set divider to 7, ADC clock is 22.1184/7 MHz */
    CLK_SetModuleClock(ADC_MODULE, CLK_CLKSEL1_ADC_S_HIRC, CLK_CLKDIV_ADC(7));

    /* Select PWM module clock source */
    CLK_SetModuleClock(PWM01_MODULE, CLK_CLKSEL1_PWM01_S_HIRC, 0);

    /* Reset PWMA channel0~channel3 */
    SYS_ResetModule(PWM03_RST);

    /* Update System Core Clock */
```

```
        SystemCoreClockUpdate();

        /*---------------------------------------------------------------------------------*/
        /* Init I/O Multi-function                                                         */
        /*---------------------------------------------------------------------------------*/
        /* Disable the GPA0 digital input path to avoid the leakage current. */
        GPIO_DISABLE_DIGITAL_PATH(PA, 0x01);

        /* Configure the GPA0 ADC analog input pins. */
        SYS->GPA_MFP = (SYS->GPA_MFP & ~(SYS_GPA_MFP_PA0_Msk)) | SYS_GPA_MFP_PA0_ADC0;
        SYS->ALT_MFP1 = (SYS->ALT_MFP1 & ~(SYS_ALT_MFP1_PA0_Msk)) | SYS_ALT_MFP1_PA0_ADC0;

        /* Set GPA multi-function pins for PWMA Channel 0 */
        SYS->GPA_MFP = (SYS->GPA_MFP & ~(SYS_GPA_MFP_PA12_Msk)) | SYS_GPA_MFP_PA12_PWM0;
        SYS->ALT_MFP1 = (SYS->ALT_MFP1 & ~(SYS_ALT_MFP1_PA12_Msk)) |SYS_ALT_MFP1_PA12_PWM0;
    }
```

After enabled ADC and PWMA module in main() function, MCU calls ADC_GET_CONVERSION_DATA() to get the sampled value by ADC channel 0, and then calls PWM_ConfigOutputChannel() to set frequency and duty cycle of PWMA channel 0.

```
/*---------------------------------------------------------------------------------------------*/
/*  Main Function                                                                              */
/*---------------------------------------------------------------------------------------------*/
int32_t main(void)
{
    uint32_t u32ADCRawData;

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, IP clock and multi-function I/O */
    SYS_Init();

    /* Lock protected registers */
    SYS_LockReg();

    printf("-----------------------------------------------------------\n");
    printf("| NUC230_240 NuEdu Basic01 LED Light Adjustment Sample Code\n");
    printf("-----------------------------------------------------------\n");
```

```
    /* Set the ADC operation mode as single, input mode as single-end and enable the
analog input channel 0 */
    ADC_Open(ADC, ADC_ADCR_DIFFEN_SINGLE_END, ADC_ADCR_ADMD_SINGLE, 0x1 << 0);

    /* Power on ADC module */
    ADC_POWER_ON(ADC);

    /* Set PWMA channel 0 output configuration */
    PWM_ConfigOutputChannel(PWMA, PWM_CH0, 100, 50);

    /* Enable PWM Output path for PWMA channel 0 */
    PWM_EnableOutput(PWMA, 0x01);

    /* Start PWM */
    PWM_Start(PWMA, 0x01);

    while(1)
    {
        /* LED4 toggle */
        LED4 ^= 1;

        /* Clear ADC convert complete interrupt flag */
        ADC_CLR_INT_FLAG(ADC, ADC_ADF_INT);

        /* ADC start conversion */
        ADC_START_CONV(ADC);

        /* Wait for ADC convert complete */
        while(ADC_GET_INT_FLAG(ADC,ADC_ADF_INT)!=ADC_ADF_INT);

        /* Get the conversion result */
        u32ADCRawData = ADC_GET_CONVERSION_DATA(ADC,0);

        /* Configure PWM duty cycle to adjust the light of RGB LED by ADC conversion
result */
        PWM_ConfigOutputChannel(PWMA, PWM_CH0, 100, (100 - (u32ADCRawData * 100 / 4096)));

        printf("ADC conversion result: %d, PWM duty cycle: %d\n", u32ADCRawData, (100 -
(u32ADCRawData * 100 / 4096)));

        /* Delay 30000 us */
        CLK_SysTickDelay(30000);
```

```
    }
}
```

3 Software and Hardware Environment

Software Environment

- BSP version
  - NUC230/240 Series BSP v3.01.002
- IDE version
  - Keil uVersion 5.26

Hardware Environment

- Circuit components
  - NUC240VE3AN NuEdu Board V2.0
  - NuEdu-Basic01 V2.0
- Diagram

NUC240 NuEdu Board V2.0		NuEdu-Basic01 V2.0
VCC	↔	VCC
ADC0 (PA.0)	↔	NU2_1(ADC_MEASURE)
PWM0 (PA.12)	↔	NU5_3(R_LED)
GND	↔	GND

4 Directory Information

EC_ NUC240_ADC_PWM_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 NUC240_ADC_PWM.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. 27, 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.