Adjusting PWM Duty Cycle Based on ADC Input Voltage

Example Code Introduction for 8-bit NuMicro® Family

Document Information

Application	This example code uses MS51 PWM to output 10 kHz waveform with 0~100% duty cycle according to 0~3V ADC input voltage.
BSP Version	MS51_Series_BSP_Keil_V1.00.004
Hardware	NuTiny-MS51DA 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

This example code uses MS51 PWM to output 10 kHz waveform with 0~100% duty cycle according to 0~3V ADC input voltage. The following sections show how to use ADC to sample an analog voltage and convert to 12-bit binary representation, and how to change PWM duty cycle through an ADC conversion result.

1.1 Principle

The MS51 series microcontroller (MCU) is embedded with a 12-bit ADC which can convert an analog input signal to 12-bit binary data. A 12-bit resolution means that an ADC conversion result will be in the range of 0~4095. The conversion result will correspond to PWM output signal duty cycle from 0%~100%. The formula below is used to express relationship between ADC conversion result and PWM duty cycle.

$$PWM\ duty\ cycle = \frac{\text{ADC conversion result}}{4095} \times 100$$

1.2 Demo Result

The execution results of this example code are shown as Figure 1-1 and Figure 1-2, which show the relationship between ADC input voltage and PWM duty cycle. The ADC input voltage is 1.02V and 1.5V for the PWM duty cycle of 34% and 50%, respectively.

Please note that channel 1 is used to measure the ADC input voltage and channel 2 is used to measure the PWM output signal.

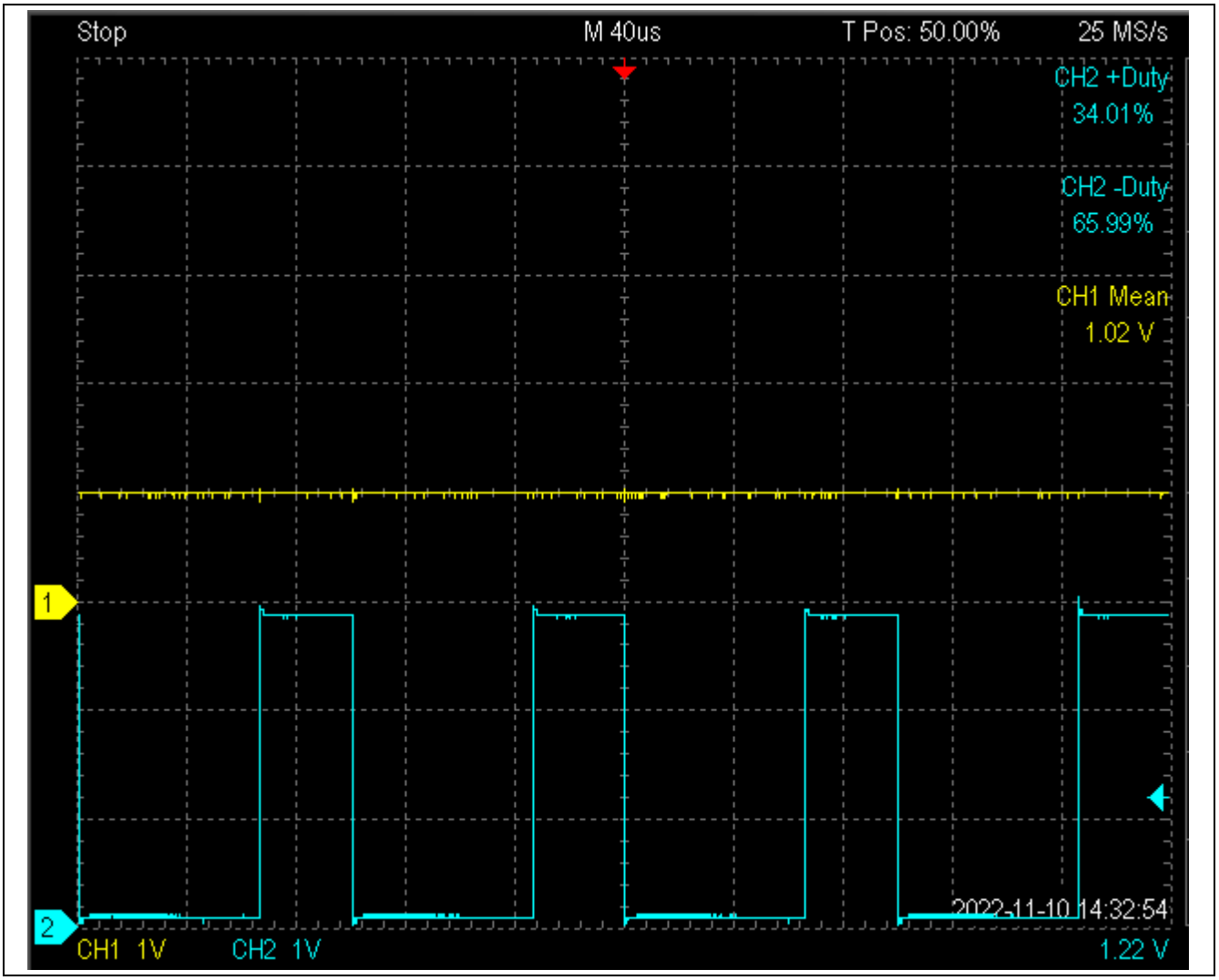

Figure 1-1 PWM Duty Cycle versus ADC Input Voltage is 1.02V

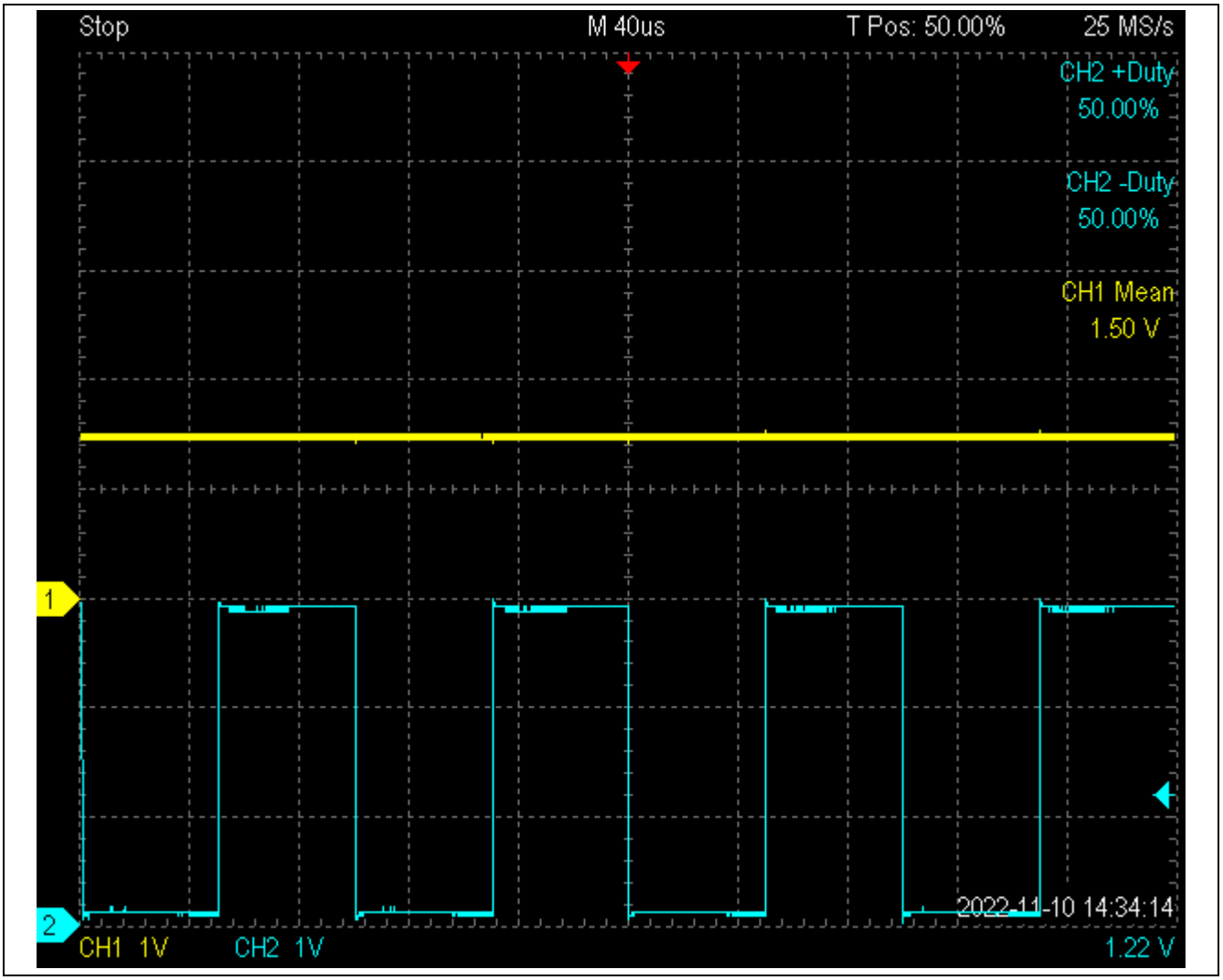

Figure 1-2 PWM Duty Cycle versus ADC Input Voltage is 1.5V

2. Code Description

The main() function is divided to three parts. The first part is the system initialization processing. It includes modify HIRC as 24 MHz and initialize related peripherals. Please refer to section 2.1 for PWM related functions and section 2.2 for ADC related functions, respectively. The second part is to start ADC conversion process in the IDLE_STATE state and get a result when ADC interrupt occurred. Lastly, calculate and update the PWM duty cycle according to the ADC conversion result in the ADC_UPDATE_STATE state.

```
void main(void)
{
    unsigned char state = IDLE_STATE;

    /* FSYS = 24MHz */
    MODIFY_HIRC(HIRC_24);
    PWM0_Init();
    ENABLE_GLOBAL_INTERRUPT;

    while (1)
    {
        switch (state)
        {
            case IDLE_STATE:
            default:

                ADC_Convert();
                state = ADC_UPDATE_STATE;
                break;


            case ADC_UPDATE_STATE:

                if (g_bADC_Flag == 1)
                {
                    g_bADC_Flag = 0;

                    i64_ADCdataConvert = u32_ADCdataAIN;
                    i64_ADCdataConvert = i64_ADCdataConvert * 100 / 4095;
                    /* Integer value */
                    i64_ADCdataConvert = (2 * i64_ADCdataConvert -
(int)(i64_ADCdataConvert));

                    PWM0_Reload((unsigned int)i64_ADCdataConvert);
                    DISABLE_ADC;
                    state = IDLE_STATE;
                }

                break;
        }
    }
}
```

2.1 PWM Related Functions

The *PWM0_Init()* function is used to initialize the PWM module with specified configurations. It includes clock source, specified channel, frequency, PWM duty cycle and so on.

The *PWM0_Reload()* function is used to calculate and update the PWM duty cycle according to an argument.

```
/**
* @brief Initialize the PWM module with specified configurations. It could be clock
source, PWM channel and frequency.
*        The default duty cycle is 50%.
*
* @param[in] None
*
* @return None
*
* @details None
*/
void PWM0_Init(void)
{
    /* Define PWM0 clock source and divider. */
    PWM0_ClockSource(PWM_FSYS, 1);

    /* Setting PWM0 channel 2 */
    ENABLE_PWM0_CH2_P10_OUTPUT;
    P10_PUSHPULL_MODE;
    PWM0_ConfigOutputChannel(2, Independent, EdgeAligned, PWM_FREQ, 50);

    PWM0_RUN();
}

/**
* @brief Update PWM duty cycle.
*
* @param[in] u16PWM0Duty Set PWM default duty cycle. The range is 0%~100%
*
* @return None
*
* @details The formula is:
*          PWM duty cycle = PWM duty register /  (PWM period register + 1)
*          -> PWM duty register = PWM duty cycle * (PWM period register + 1)
*/
void PWM0_Reload(unsigned int u16PWM0Duty)
{
    /* Reload PWM duty */
    unsigned int u16PWM0DutyCycleTemp = 0;
    u16PWM0DutyCycleTemp = ((PWM_FREQ + 1) / 100) * u16PWM0Duty;

    PWM2H = u16PWM0DutyCycleTemp >> 8;
    PWM2L = u16PWM0DutyCycleTemp;

    set_PWMCON0_LOAD;
}
```

2.2 ADC Related Functions

The *ADC_Convert()* function is used to initialize the ADC module with specified configurations. It includes clock divider, specified channel and acquisition time for calculating ADC sampling time. After a conversion is completed, user can get an ADC conversion result in the *ADC_ISR()* function.

```
/**
* @brief ADC interrupt service routine
*
* @param[in] None
*
* @return None
*
* @details Get ADC conversion result.
*/
void ADC_ISR(void) interrupt 11
{
    _push_(SFRS);

    clr_ADCCON0_ADCF;//Clear ADC interrupt flag

    u32_ADCdataAIN = ADCRH << 4;
    u32_ADCdataAIN |= ADCRL;
    g_bADC_Flag = 1;
    _pop_(SFRS);
}

/**
* @brief Initialize the ADC module with specified configurations and start ADC convert.
*
* @param[in] None
*
* @return None
*
* @details None
*/
void ADC_Convert(void)
{
    /*Enable channel 5 */
    ENABLE_ADC_CH5;
    ADCCON1 |= 0x30;          /* clock divider */
    ADCCON2 |= 0x0E;          /* AQT time */
    AUXR1 |= SET_BIT4;        /* ADC clock low speed */
    clr_ADCCON0_ADCF;
    set_ADCCON0_ADCS;

    /* Enable ADC interrupt */
    set_IE_EADC;
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - MS51_Series_BSP_Keil_V1.00.004
- IDE version
  - Keil uVersion 5.26

3.2 Hardware Requirements

- Circuit components
  - NuTiny-MS51DA V1.1
- Pin Connect
  - Connect the ADC_CH5(P0.4) to the external power supply to get a different conversion result.
  - Connect the PWM0_CH2(P1.0) to an oscilloscope to monitor the PWM output signal.

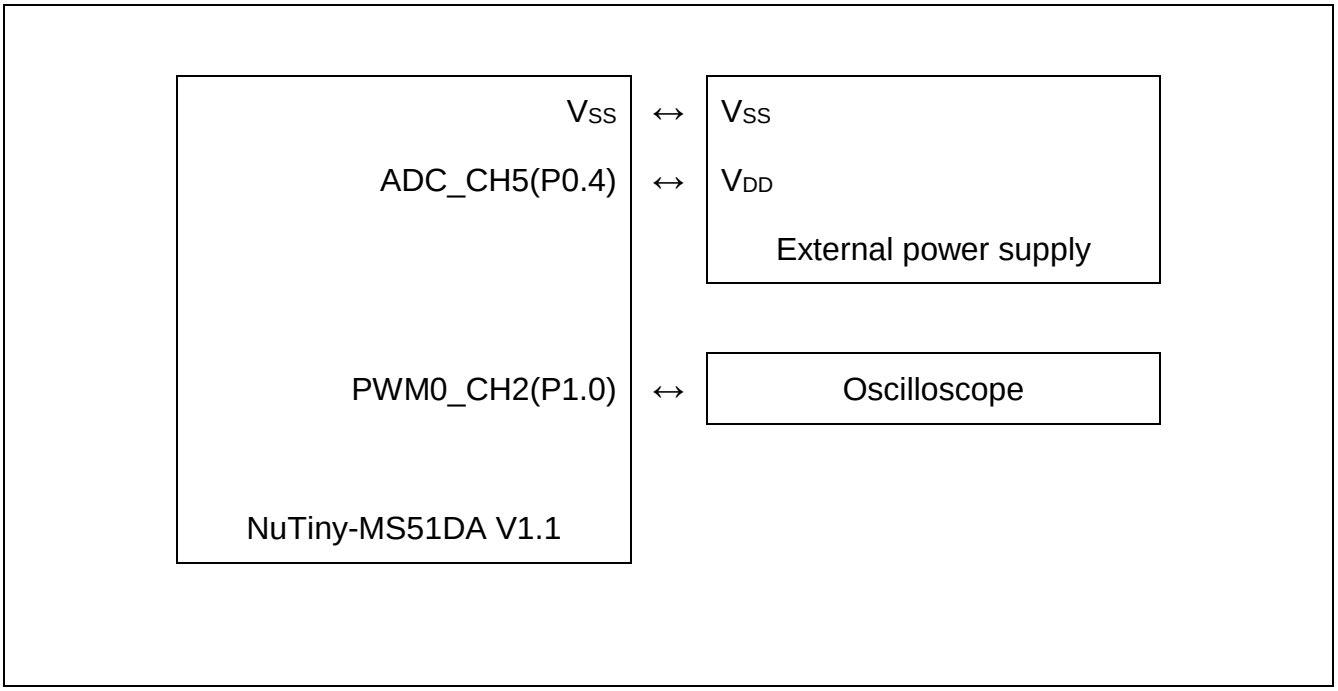

Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

EC_MS51_ADC_PWM_V1.00	
Library	Sample code header and source files
Device	MS51 series flash device header file
Startup	A51 startup file and executable file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code

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 *MS51_ADC_PWM.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2022.11.10	1.00	Initial version.

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.