Using M251 QSPI with PDMA to Simulate PWM

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code uses M251 QSPI with PDMA to simulate PWM output signals.
BSP Version	M251_Series_BSP_CMSIS_V3.02.007
Hardware	NuMaker-M251SD-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 the M251 QSPI with PDMA to simulate PWM output signals.

Most MCUs (microcontrollers) come with built-in PWM interface, but if you need multiple PWM outputs, you may face a shortage of available channels. In such cases, if there are unused peripherals in your application, you can simulate PWM signals using those peripherals. This example code demonstrates the usage of the QSPI interface with PDMA to output PWM signals. PDMA is used in Scatter-Gather mode for continuous data transfer.

This example code does not require additional external hardware modules. If you need to observe the output signals, you can use devices such as an oscilloscope or logic analyzer to measure the output waveform.

1.1 Principle

To simulate PWM signals using the QSPI interface, you need to determine the smallest controllable unit to calculate the data size every QSPI transmission and the number of data required every transmission. In this example, the minimum controllable unit is 0.1% duty cycle, with a controllable range of 0.0% to 100.0% duty cycle. Based on this, each data transmitted via QSPI is 25 bits, and each transmission requires 40 data. A total of 1000 bits are used to simulate one PWM signal, so the PWM signal frequency is one-thousandth of the QSPI clock frequency. If the QSPI clock frequency is 1 MHz, then the PWM signal frequency is 1 kHz.

The transmitted data is stored in an array with 40 elements, where the values of the elements are calculated based on the duty cycle. Three variables are used to determine the elements in the array: the ten times value of the duty cycle, the quotient of the ten times value divided by 25, and the remainder of the ten times value divided by 25. The duty cycle is recorded as the 10 times value to avoid including decimal information since the data type is unsigned int. By performing division operations, the quotient determines which element in the array the signal transition occurs, and the remainder determines which bit within that element the signal transition occurs.

In a complete PWM signal cycle, the signal is high before the signal transition and low after the signal transition. Represented in the array, the elements before the signal transition element have a value of 1, and the elements after the signal transition element have a value of 0. Within the signal transition element, bits (25 - remainder) to (25 - remainder + 1) transition from 1 to 0. The QSPI array to be transmitted is constructed based on this rule.

For example, to simulate a PWM with a duty cycle of 63.5%, the signal transition occurs at the 15th to 16th bits of the 25th element (635/25 = 25...10). Using shifting and logical operators, the value of the 25th element in the array is determined as 0x1FFFC00. The values of the 1st to 24th elements are all 0x1FFFFFF to simulate a signal value of 1, and the values of the 26th to 40th elements are all 0 to simulate a signal value of 0.

Since 1000 bits are used to simulate a complete PWM signal cycle, the PWM signal frequency is one-thousandth of the QSPI bit clock frequency.

Only the MOSI pin needs to be declared as an output in the multi-pin function since the pins are only used for simulating the PWM signal. The CLK, MISO, and SS pins do not need to be declared or used.

The QSPI transmission is accelerated using PDMA. PDMA operates in Scatter-Gather mode and is configured using a DSCT table. The DSCT table contains the transmission settings for PDMA, the source and destination addresses, and the configuration for the next DSCT table after the transmission is complete.

1.2 Demo Result

The compilation and programming results of this example code are shown in Figure 1-1.

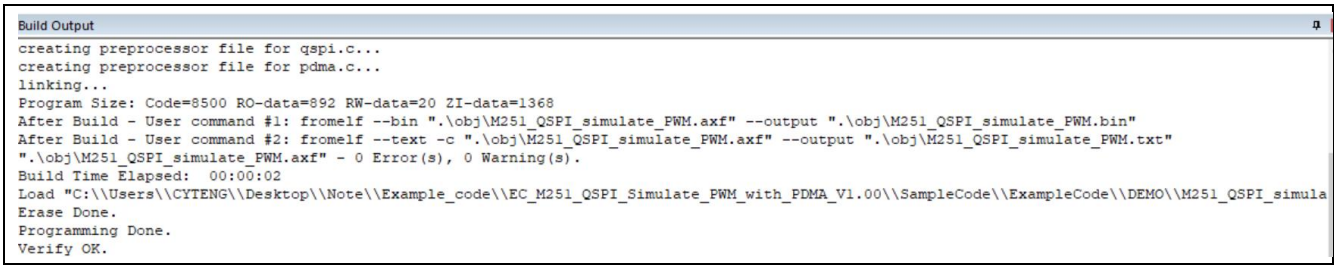

Figure 1-1 Compilation and Programming Results

The MOSI signal pin (PA0) output, representing the PWM signal, can be measured using a logic analyzer. The duty cycle of the PWM signal depends on the modification of the transmitted data. Figure 1-2 and Figure 1-3 show the output signals for duty cycles of 75% and 25% respectively.

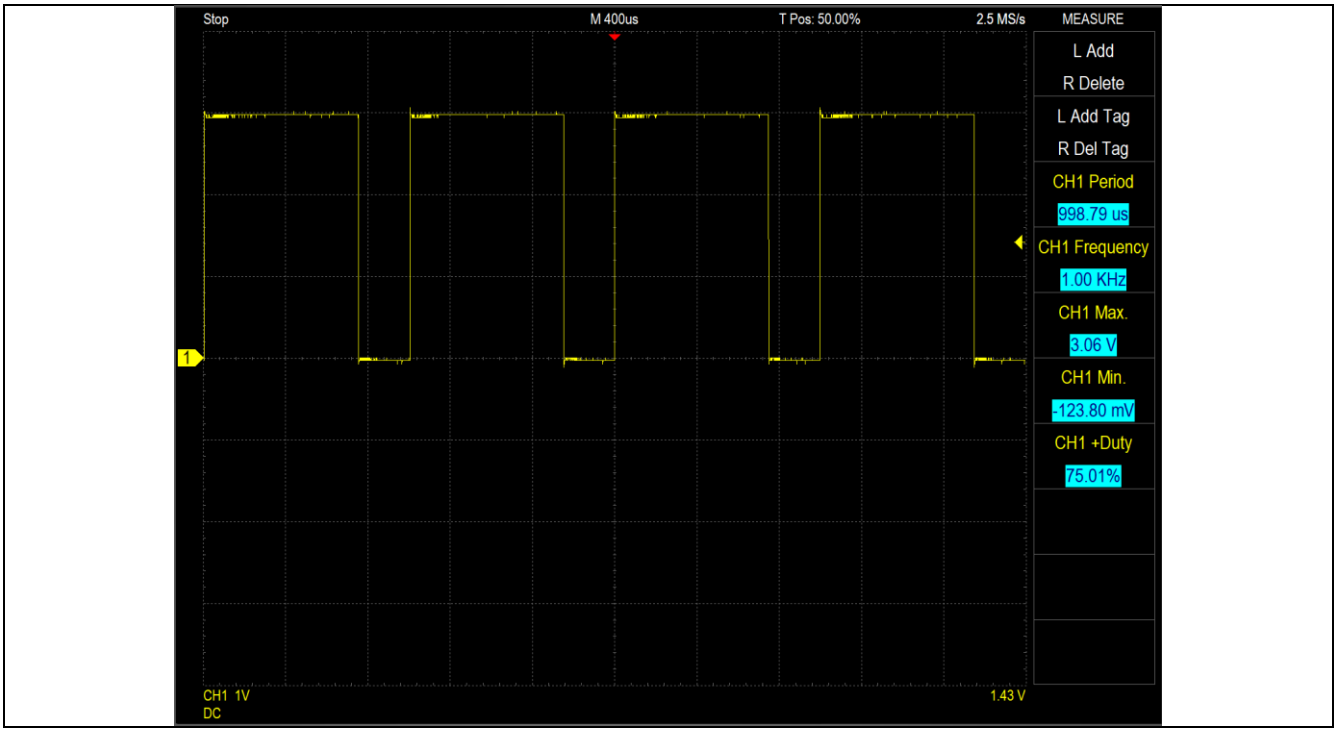

Figure 1-2 Output Signals for Duty Cycles of 75%

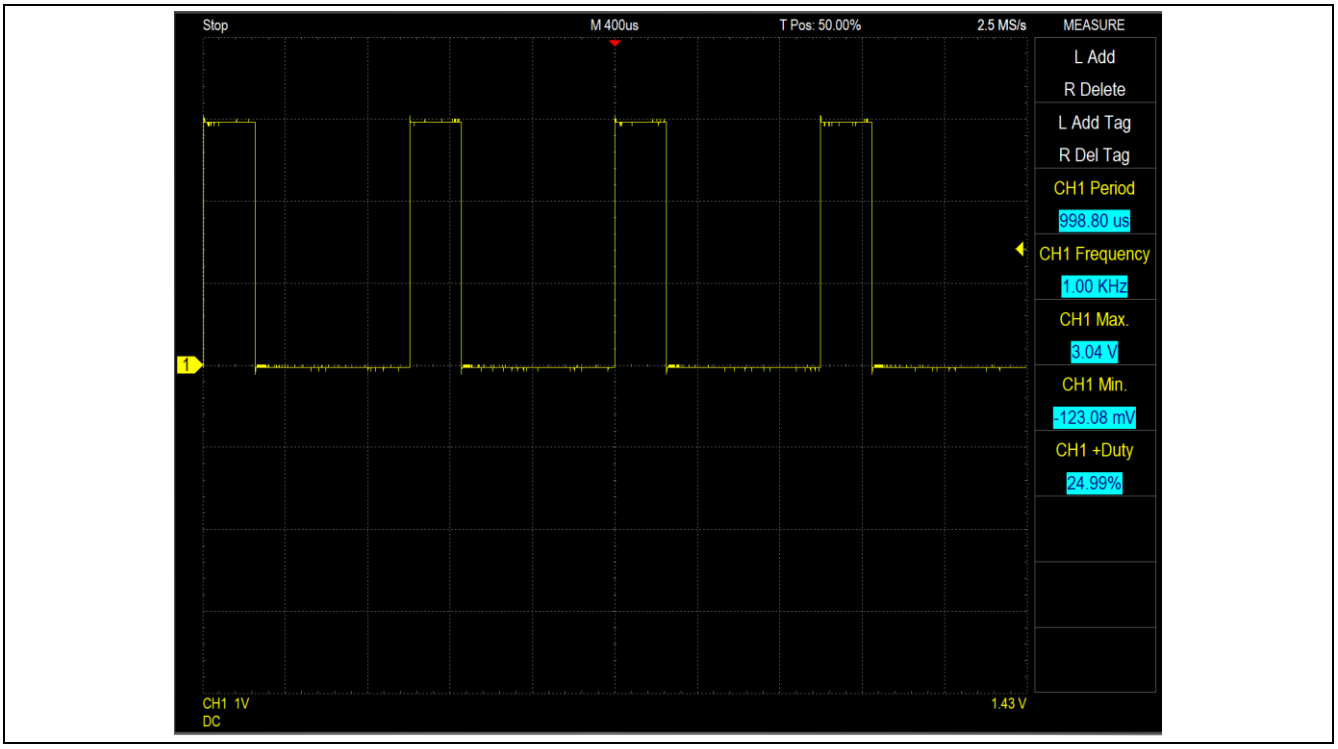

Figure 1-3 Output Signals for Duty Cycles of 75%

2. Code Description

After initializing the system, create an array for the QSPI transmission data by calculating the quotient and remainder from the variable Duty_right_now. The code is located in main.c.

```
int32_t main(void)
{
    /* Unlock protected registers */
    SYS_UnlockReg();

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

    /* Lock protected registers */
    SYS_LockReg();

    /* Init UART */
    UART0_Init();

    /* Calculate Duty array */
    Calculate_Duty_array(Duty_right_now);
```

The function Calculate_Duty_array is used to calculate the contents of an array.

```
void Calculate_Duty_array(uint16_t Duty_Value)
{
    uint8_t Duty_remainder, Duty_Quotient;

    /* Calculate Duty parameter about quotient */
    Duty_Quotient = Duty_Value / 25;

    /* Calculate Duty parameter about remainder */
    Duty_remainder = Duty_Value % 25;

    /* Build transmission data array */
    for (uint8_t i = 0; i < 40; i++)
    {

        /*High signal before signal transition*/
        if (i <  Duty_Quotient)
        {
            Duty_array[i] = 0x1FFFFFF;
        }

        /*Signal transition located*/
        else if (i ==  Duty_Quotient)
        {
            Duty_array[i] = 0x1ffffff & (~(0x1ffffff+(1 << (25 - Duty_remainder))));
        }

        /*Low signal after signal transition*/
        else if (i >  Duty_Quotient)
        {
            Duty_array[i] = 0;
        }
    }

}
```

Open PDMA and QSPI, and set the interval time between QSPI data transmissions.

```
    /* Open Channel 1, 2 */
```

```
    PDMA_Open(PDMA, 1 | 2);

    /* Enable QSPI functions */
    /* Bit clock rate 1 MHz. */
    /* Master mode, 25-bit word width, mode 0. */
    QSPI_Open(QSPI0, SPI_MASTER, SPI_MODE_0, 25, 1000000);

    /* Set Suspend cycle */
    QSPI_SET_SUSPEND_CYCLE(QSPI0, 0);
```

Configure the DSCT table used for Scatter-Gather mode in PDMA by setting the registers PDMA_DSCTn_CTL, PDMA_DSCTn_SA, PDMA_DSCTn_DA, and PDMA_DSCTn_NEXT in the form of a structure. Additionally, enable QSPI and PDMA.

```
    /* Define DSCT structure */
    DMA_DESC_T DMA_DESC[2];

    /*DSCT0 Description*/
    DMA_DESC[0].u32Ctl =
        ((40 - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | /* Transfer count is 40 */ \
        PDMA_WIDTH_32 |     /* Transfer width is 32 bits(one word) */ \
        PDMA_SAR_INC |      /* Source increment size is 32 bits(one word) */ \
        PDMA_DAR_FIX |      /* Destination increment size is 32 bits(one word) */ \
        PDMA_REQ_SINGLE |   /* Transfer type is single transfer type */ \
        PDMA_BURST_128 |    /* No effect in single transfer type */ \
        PDMA_TBINTDIS_DISABLE |    /* Disable transfer done and table empty interrupt */ \
        PDMA_OP_SCATTER;    /* Operation mode is scatter-gather mode */

    /* Configure source address */
    DMA_DESC[0].u32Src = (uint32_t)Duty_array;

    /* Configure destination address */
    DMA_DESC[0].u32Dest = (uint32_t)&QSPI0->TX;

    /* Configure next descriptor table address */
    DMA_DESC[0].u32Offset = (uint32_t)&DMA_DESC[1] - (PDMA->SCATBA); /* next descriptor
    table is table 2 */

    DMA_DESC[1].u32Ctl =
        ((40 - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | /* Transfer count is 40 */ \
        PDMA_WIDTH_32 |     /* Transfer width is 32 bits(one word) */ \
        PDMA_SAR_INC |      /* Source increment size is 32 bits(one word) */ \
        PDMA_DAR_FIX |      /* Destination increment size is 32 bits(one word) */ \
        PDMA_REQ_SINGLE |   /* Transfer type is single transfer type */ \
        PDMA_BURST_128 |    /* No effect in single transfer type */ \
        PDMA_TBINTDIS_DISABLE |    /* Disable transfer done and table empty interrupt */ \
        PDMA_OP_SCATTER;    /* Operation mode is scatter-gather mode */

    /* Configure source address */
    DMA_DESC[1].u32Src = (uint32_t)Duty_array;

    /* Configure destination address */
    DMA_DESC[1].u32Dest = (uint32_t)&QSPI0->TX;

    /* Configure next descriptor table address */
    DMA_DESC[1].u32Offset = (uint32_t)&DMA_DESC[0] - (PDMA->SCATBA); /* next descriptor
    table is table 1 */
```

Set the transfer mode of PDMA and start the transmission of QSPI with PDMA.

```
     /* Set transfer mode*/
    PDMA_SetTransferMode(PDMA, 0, PDMA_QSPI0_TX, TRUE, (uint32_t)&DMA_DESC[0]);
    PDMA_SetTransferMode(PDMA, 1, PDMA_QSPI0_TX, TRUE, (uint32_t)&DMA_DESC[1]);

    /* Start QSPI with PDMA */
    QSPI_TRIGGER_TX_PDMA(QSPI0);
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M251_Series_BSP_CMSIS_V3.02.007
- IDE version
  - Keil uVision 5.39

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M251SD V1.1
- Pin Connect
  - Connect the UART0 TX (PB.13) pin to the PC UART RX and connect the UART0 RX (PB.12) pin to the PC UART TX to show the execution results of this example code. To measure the PWM output signal, you can connect a device to the QSPI0 MOSI (PA.0) pin.

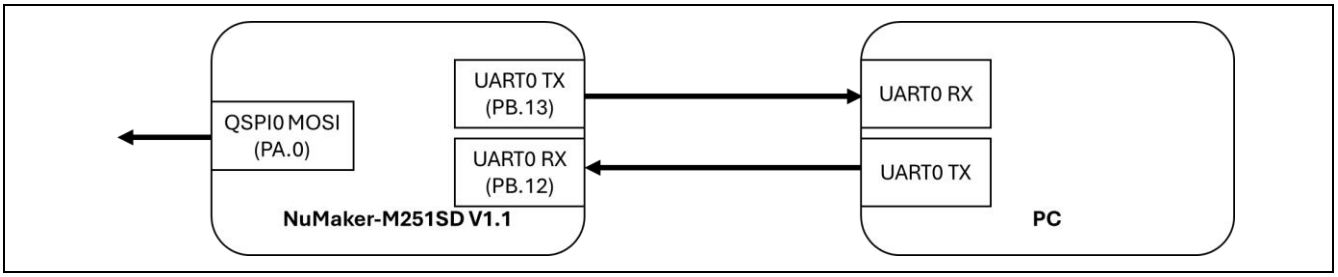

Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

EC_M251_QSPI_with_PDMA_to_Simulate_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	
Project	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 *M251_QSPI_with_PDMA_to_Simulate_PWM.uvprojx*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2025.01.15	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.