QSPI and I2S Use PDMA to Transfer

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	QSPI and I2S use PDMA to transfer data
BSP Version	M480 BSP CMSIS V3.03.001
Hardware	NuMaker-PFM-M487 Ver3.0

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

QSPI and I2S use PDMA to transfer data. QSPI0 demonstrates how to use PDMA Basic mode to transfer data. I2S0 demonstrates how to use PDMA Scatter-Gather mode for transmission.

1.2 Principle

Connect QSPI0 TX to QSPI0 RX and I2S0 TX and I20S RX to execute this example code. There are 64 data which is 32 bits in the code. The data is transferred to the QSPI0 TX through the PDMA Basic mode. QSPI0 transfers data from TX to RX. And then PDMA moves data from QSPI0 RX to MasterRxBuffer. I2S0 TX obtains data from MasterRxBuffer by PDMA Scatter-Gather Mode, and I2S TX pass it to RX. I2S0 RX data is transferred to SlaveRxBuffer through PDMA. Finally, print the data in the SlaveRxBuffer through UART, and verify the data is correct or not. (Since I2S0 TX would sent a dummy data at beginning, the size of SlaveRXBuffer need to declare 65.)

1.3 Demo Result

The transmission data of QSPI0 is as follows.

UART #1

```
+------------------------------------------------------------+
|                  QSPI + I2S + PDMA Sample Code             |
+------------------------------------------------------------+


QSPI0 to I2S0 Loop test with PDMA

QSPI0 TX                QSPI0 RX
0x55000001              0x55000001
0x55000002              0x55000002
0x55000003              0x55000003
0x55000004              0x55000004
0x55000005              0x55000005
0x55000006              0x55000006
0x55000007              0x55000007
0x55000008              0x55000008
0x55000009              0x55000009
0x5500000A              0x5500000A
0x5500000B              0x5500000B
0x5500000C              0x5500000C
0x5500000D              0x5500000D
0x5500000E              0x5500000E
0x5500000F              0x5500000F
0x55000010              0x55000010
0x55000011              0x55000011
0x55000012              0x55000012
0x55000013              0x55000013
0x55000014              0x55000014
0x55000015              0x55000015
0x55000016              0x55000016
0x55000017              0x55000017
0x55000018              0x55000018
0x55000019              0x55000019
0x5500001A              0x5500001A
0x5500001B              0x5500001B
0x5500001C              0x5500001C
0x5500001D              0x5500001D
0x5500001E              0x5500001E
0x5500001F              0x5500001F
0x55000020              0x55000020
0x55000021              0x55000021
0x55000022              0x55000022
0x55000023              0x55000023
0x55000024              0x55000024
0x55000025              0x55000025
0x55000026              0x55000026
0x55000027              0x55000027
0x55000028              0x55000028
0x55000029              0x55000029
0x5500002A              0x5500002A
0x5500002B              0x5500002B
0x5500002C              0x5500002C
0x5500002D              0x5500002D
0x5500002E              0x5500002E
0x5500002F              0x5500002F
0x55000030              0x55000030
0x55000031              0x55000031
0x55000032              0x55000032
0x55000033              0x55000033
0x55000034              0x55000034
0x55000035              0x55000035
0x55000036              0x55000036
0x55000037              0x55000037
```

```
0x55000038              0x55000038
0x55000039              0x55000039
0x5500003A              0x5500003A
0x5500003B              0x5500003B
0x5500003C              0x5500003C
0x5500003D              0x5500003D
0x5500003E              0x5500003E
0x5500003F              0x5500003F
0x55000040              0x55000040


Exit QSPI .
```

The transmission data of I2S0 is as follows.

UART #1

I2S TX Buffer	I2S RX Buffer
0x55000001	0x55000001
0x55000002	0x55000002
0x55000003	0x55000003
0x55000004	0x55000004
0x55000005	0x55000005
0x55000006	0x55000006
0x55000007	0x55000007
0x55000008	0x55000008
0x55000009	0x55000009
0x5500000A	0x5500000A
0x5500000B	0x5500000B
0x5500000C	0x5500000C
0x5500000D	0x5500000D
0x5500000E	0x5500000E
0x5500000F	0x5500000F
0x55000010	0x55000010
0x55000011	0x55000011
0x55000012	0x55000012
0x55000013	0x55000013
0x55000014	0x55000014
0x55000015	0x55000015
0x55000016	0x55000016
0x55000017	0x55000017
0x55000018	0x55000018
0x55000019	0x55000019
0x5500001A	0x5500001A
0x5500001B	0x5500001B
0x5500001C	0x5500001C
0x5500001D	0x5500001D
0x5500001E	0x5500001E
0x5500001F	0x5500001F
0x55000020	0x55000020
0x55000021	0x55000021
0x55000022	0x55000022
0x55000023	0x55000023
0x55000024	0x55000024
0x55000025	0x55000025
0x55000026	0x55000026
0x55000027	0x55000027
0x55000028	0x55000028
0x55000029	0x55000029
0x5500002A	0x5500002A
0x5500002B	0x5500002B
0x5500002C	0x5500002C
0x5500002D	0x5500002D
0x5500002E	0x5500002E
0x5500002F	0x5500002F
0x55000030	0x55000030
0x55000031	0x55000031
0x55000032	0x55000032
0x55000033	0x55000033
0x55000034	0x55000034
0x55000035	0x55000035
0x55000036	0x55000036
0x55000037	0x55000037
0x55000038	0x55000038
0x55000039	0x55000039
0x5500003A	0x5500003A
0x5500003B	0x5500003B
0x5500003C	0x5500003C
0x5500003D	0x5500003D
0x5500003E	0x5500003E

```
0x5500003F              0x5500003F
0x55000040              0x55000040


Exit I2S .
```

2 Code Description

QSPI0 uses PDMA Basic mode to transfer：

```
/* Enable PDMA channels */
    PDMA_Open(PDMA, (1 << QSPI_MASTER_TX_DMA_CH) | (1 << QSPI_MASTER_RX_DMA_CH) );

    /* Set transfer width (32 bits) and transfer count */
    PDMA_SetTransferCnt(PDMA, QSPI_MASTER_TX_DMA_CH, PDMA_WIDTH_32, TEST_COUNT);

    /* Set source/destination address and attributes */
    PDMA_SetTransferAddr(PDMA, QSPI_MASTER_TX_DMA_CH,
    (uint32_t)g_au32MasterToSlaveTestPattern, PDMA_SAR_INC, (uint32_t)&QSPI0->TX,
    PDMA_DAR_FIX);

    /* Set request source; set basic mode. */
    PDMA_SetTransferMode(PDMA, QSPI_MASTER_TX_DMA_CH, PDMA_QSPI0_TX, FALSE, 0);

    /* Single request type. SPI only support PDMA single request type. */
    PDMA_SetBurstType(PDMA, QSPI_MASTER_TX_DMA_CH, PDMA_REQ_SINGLE, 0);

    /* Disable table interrupt */
    PDMA->DSCT[QSPI_MASTER_TX_DMA_CH].CTL |= PDMA_DSCT_CTL_TBINTDIS_Msk;

    /* Set transfer width (32 bits) and transfer count */
    PDMA_SetTransferCnt(PDMA, QSPI_MASTER_RX_DMA_CH, PDMA_WIDTH_32, TEST_COUNT);

    /* Set source/destination address and attributes */
    PDMA_SetTransferAddr(PDMA, QSPI_MASTER_RX_DMA_CH, (uint32_t)&QSPI0->RX, PDMA_SAR_FIX,
    (uint32_t)g_au32MasterRxBuffer, PDMA_DAR_INC);

    /* Set request source; set basic mode. */
    PDMA_SetTransferMode(PDMA, QSPI_MASTER_RX_DMA_CH, PDMA_QSPI0_RX, FALSE, 0);

    /* Single request type. SPI only support PDMA single request type. */
    PDMA_SetBurstType(PDMA, QSPI_MASTER_RX_DMA_CH, PDMA_REQ_SINGLE, 0);

    /* Disable table interrupt */
    PDMA->DSCT[QSPI_MASTER_RX_DMA_CH].CTL |= PDMA_DSCT_CTL_TBINTDIS_Msk;

    /* Enable SPI master DMA function */
    QSPI_TRIGGER_TX_PDMA(QSPI0);
    QSPI_TRIGGER_RX_PDMA(QSPI0);
```

I2S0 uses PDMA Scatter-Gather mode to transfer：

```
     /* Enable PDMA channels */
    PDMA_Open(PDMA, (1 << I2S0_TX_DMA_CH) | (1 << I2S0_RX_DMA_CH));

    /* Tx description */
    g_asDescTable_TX[0].CTL = ((BUFF_LEN - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_32
    | PDMA_SAR_INC | PDMA_DAR_FIX | PDMA_REQ_SINGLE | PDMA_OP_SCATTER;
    g_asDescTable_TX[0].SA = (uint32_t)&g_au32MasterRxBuffer;
    g_asDescTable_TX[0].DA = (uint32_t)&I2S0->TXFIFO;
    g_asDescTable_TX[0].FIRST = (uint32_t)&g_asDescTable_TX[1] - (PDMA->SCATBA);

    g_asDescTable_TX[1].CTL = ((BUFF_LEN - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_32
    | PDMA_SAR_INC | PDMA_DAR_FIX | PDMA_REQ_SINGLE | PDMA_OP_BASIC;
    g_asDescTable_TX[1].SA = (uint32_t)&g_au32MasterRxBuffer[BUFF_LEN];
    g_asDescTable_TX[1].DA = (uint32_t)&I2S0->TXFIFO;
    g_asDescTable_TX[1].FIRST = (uint32_t)&g_asDescTable_TX[0] - (PDMA->SCATBA);
```

```
    /* Rx description */
    g_asDescTable_RX[0].CTL = ((BUFF_LEN - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_32
    | PDMA_SAR_FIX | PDMA_DAR_INC | PDMA_REQ_SINGLE | PDMA_OP_SCATTER;
    g_asDescTable_RX[0].SA = (uint32_t)&I2S0->RXFIFO;
    g_asDescTable_RX[0].DA = (uint32_t)&g_au32SlaveRxBuffer;
    g_asDescTable_RX[0].FIRST = (uint32_t)&g_asDescTable_RX[1] - (PDMA->SCATBA);

    g_asDescTable_RX[1].CTL = (((BUFF_LEN+1) - 1) << PDMA_DSCT_CTL_TXCNT_Pos) |
    PDMA_WIDTH_32 | PDMA_SAR_FIX | PDMA_DAR_INC | PDMA_REQ_SINGLE | PDMA_OP_BASIC;
    g_asDescTable_RX[1].SA = (uint32_t)&I2S0->RXFIFO;
    g_asDescTable_RX[1].DA = (uint32_t)&g_au32SlaveRxBuffer[BUFF_LEN];
    g_asDescTable_RX[1].FIRST = (uint32_t)&g_asDescTable_RX[0] - (PDMA->SCATBA);

    /* Set PDMA Transfer Mode */
    PDMA_SetTransferMode(PDMA, I2S0_TX_DMA_CH, PDMA_I2S0_TX, 1,
    (uint32_t)&g_asDescTable_TX[0]);
    PDMA_SetTransferMode(PDMA, I2S0_RX_DMA_CH, PDMA_I2S0_RX, 1,
    (uint32_t)&g_asDescTable_RX[0]);
```

3 Software and Hardware Environment

Software Environment

- BSP version
  - M480 Series BSP CMSIS v3.03.000
- IDE version
  - Keil uVersion 5.26

Hardware Environment

- Circuit components
  - NuMaker-PFM-M487 Ver3.0

4 Directory Information

EC_M480_QSPI_PDMA_I2S_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 M480_QSPI_PDMA_I2S.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. 10, 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.