AN0009
Application Note

DMA with Flexible Mapping

Introduction

This application note describes how to use DMA with flexible mapping function to make DMA request configuration more free and flexible. This function is only available for some AT32 MCUs.

Applicable products:

Part number	AT32Fxx

Contents

List of Tables

List of Figures

1 Overview

The DMA with flexible mapping feature is available in some of Artery MCUs (AT32F413\ AT32F415\ AT32F403A\ AT32F407). Such function makes DMA channel configuration more flexible by transferring the DMA request channel of a certain peripheral to any one of the 14 channels in DMA1 or DMA2 (for example, assign the DMA request of SPI1 RX data to the channel 7 of DMA1).

This application note describes how to use DMA with flexible mapping request to make DMA transfers more flexible.

2 DMA configuration and usage

2.1 Regular DMA configuration (fixed DMA mapping mode)

Regular DMA configuration is as follows: the DMA channel of a peripheral is fixed and unchanged, and it is ready to use as long as it is configured and enabled. This means that if the user wants to enable the DMA of a certain peripheral, the channel cannot be changed. For example, to use the DMA function of SPI1 RX, the user has to check the corresponding reference manual, as shown in Table 1 below.

Table 1. DMA with fixed mapping request

Peripheral	Channel **1**	Channel **2**	Channel **3**	Channel **4**	...	Channel **7**
ADC1	ADC1				...	
SPI/I^2S		SPI1/I2S1_RX	SPI1/I2S1_TX	SPI2/I2S2_RX	...	
...	...	...	...	...	...	...
TMR4	TMR4_CH1			TMR4_CH2	...	TMR4_OVERFLOW

From Table 1, we can see that the channel 2 of DMA1 should be enabled.

2.2 How to use DMA with flexible mapping function

Flexible mapping request function means that the DMA channel of a peripheral is not fixed, and the user can select any one of the 14 channels in DMA1 or DMA2.

To use this feature, the following procedures should be respected:

1) Enable flexible DMA mapping request

Write 1 to the 24th bit (DMA_FLEX_EN) of the DMA channel source register 1 (DMA_SRC_SEL1).

Write the relevant hardware ID to the register corresponding to the channel configuration.

The DMA request of each peripheral is assigned with a hardware ID, which needs to be written to the channel source register. Such ID can be found in the Table 2 below. For example, AT32F403A is as follows:

Table 2. Summary of DMA flexible mapping requests of each channel for AT32F403A

CHx_SRC	Request source	CHx_SRC	DMA source	CHx_SRC	Request source	CHx_SRC	Request source
0	No select	1	ADC1	2	reserved	3	ADC3
4	reserved	5	DAC1	6	DAC2	7	reserved
8	reserved	9	SPI1_RX	10	SPI1_TX	11	SPI2_RX
12	SPI2_TX	13	SPI3_RX	14	SPI3_TX	15	SPI4_RX
16	SPI4_TX	17	I2S2EXT_RX	18	I2S2EXT_TX	19	I2S3EXT_RX
20	I2S3EXT_TX	21	reserved	22	reserved	23	reserved
24	reserved	25	USART1_RX	26	USART1_TX	27	USART2_RX

CHx_SRC	Request source	CHx_SRC	DMA source	CHx_SRC	Request source	CHx_SRC	Request source
28	USART2_TX	29	USART3_RX	30	USART3_TX	31	UART4_RX
32	UART4_TX	33	UART5_RX	34	UART5_TX	35	USART6_RX
36	USART6_TX	37	UART7_RX	38	UART7_TX	39	UART8_RX
40	UART8_TX	41	I2C1_RX	42	I2C1_TX	43	I2C2_RX
44	I2C2_TX	45	I2C3_RX	46	I2C3_TX	47	reserved
48	reserved	49	SDIO1	50	SDIO2	51	reserved
52	reserved	53	TMR1_TRIG	54	TMR1_HALL	55	TMR1_UP
56	TMR1_CH1	57	TMR1_CH2	58	TMR1_CH3	59	TMR1_CH4
60	reserved	61	TMR2_TRIG	62	reserved	63	TMR2_UP
64	TMR2_CH1	65	TMR2_CH2	66	TMR2_CH3	67	TMR2_CH4
68	reserved	69	TMR3_TRIG	70	reserved	71	TMR3_UP
72	TMR3_CH1	73	TMR3_CH2	74	TMR3_CH3	75	TMR3_CH4
76	reserved	77	TMR4_TRIG	78	reserved	79	TMR4_UP
80	TMR4_CH1	81	TMR4_CH2	82	TMR4_CH3	83	TMR4_CH4
84	reserved	85	TMR5_TRIG	86	reserved	87	TMR5_UP
88	TMR5_CH1	89	TMR5_CH2	90	TMR5_CH3	91	TMR5_CH4
92	reserved	93	reserved	94	reserved	95	TMR6_UP
96	reserved	97	reserved	98	reserved	99	reserved
100	reserved	101	reserved	102	reserved	103	TMR7_UP
104	reserved	105	reserved	106	reserved	107	reserved
108	reserved	109	TMR8_TRIG	110	TMR8_HALL	111	TMR8_UP
112	TMR8_CH1	113	TMR8_CH2	114	TMR8_CH3	115	TMR8_CH4
116	reserved	117	reserved	118	reserved	119	reserved
120	reserved	121	reserved	122	reserved	123	reserved
124	reserved	125	reserved	126	reserved	127	reserved
128	reserved	129	reserved	130	reserved	131	reserved
132	reserved	133	reserved	134	reserved	135	reserved
136	reserved	137	reserved	138	reserved	139	reserved
140	reserved	141	reserved	142	reserved	143	reserved
144	reserved	145	reserved	146	reserved	147	reserved
148	reserved	149	reserved	150	reserved	151	reserved
152	reserved	153	reserved	154	reserved	155	reserved
156	reserved	157	reserved	158	reserved	159	reserved
160	reserved	161	reserved	162	reserved	163	reserved

CHx_SRC	Request source	CHx_SRC	DMA source	CHx_SRC	Request source	CHx_SRC	Request source
164	reserved	165	reserved	166	reserved	167	reserved
168	reserved	169	reserved	170	reserved	171	reserved
172	reserved	173	reserved	174	reserved	175	reserved

The CHx_SRC setting value in the above table is the hardware ID, which should be written to the corresponding channel bit in the channel source register. For example, to map the DMA request of SPI1 RX to the DMA1 channel 7, the 0x09 should be written to the CH7_SRC[23:16] in the DMA_SRC_SEL1 register. Other settings are the same as that of regular DMA. Going through the above three steps, the flexible mapping function is ready to use.

Note: The mapping mode of DMA1/2 must be aligned when the DMA_FLEX_EN of DMA1/2 has to be set 1 or 0 at the same time. It is not feasible that DMA1 is configured as fixed mapping mode but DMA2 as flexible mapping mode.

2) DMA flexible mapping library function

The corresponding library functions are available in the dma.h\dma.c of BSP, which can be used to configure the DMA flexible mapping mode.

Library functions are described as follows:

Table 3. DMA flexible mapping library functions

void DMA_Flexible_Config(DMA_Type *DMAx,uint8_t Flex_Channelx,uint8_t Hardware_ID);	
Parameters	Description
DMAx	DMA1 or DMA2
Flex_Channelx	Channel selection (ch1 to ch7)
Hardware_ID	Corresponding hardware ID

This function can be called to use as long as the regular DMA mapping function is set, as shown in Figure 1 below:

Figure 1. Example of DMA flexible mapping library functions

```
/* dma2 channel1 configuration */
dma_reset(DMA2_CHANNEL1);
dma_init_struct.buffer_size = BUFFER_SIZE;
dma_init_struct.direction = DMA_DIR_MEMORY_TO_PERIPHERAL;
dma_init_struct.memory_base_addr = (uint32_t)src_buffer;
dma_init_struct.memory_data_width = DMA_MEMORY_DATA_WIDTH_HALFWORD;
dma_init_struct.memory_inc_enable = TRUE;
dma_init_struct.peripheral_base_addr = (uint32_t)0x4001100C;
dma_init_struct.peripheral_data_width = DMA_PERIPHERAL_DATA_WIDTH_HALFWORD;
dma_init_struct.peripheral_inc_enable = FALSE;
dma_init_struct.priority = DMA_PRIORITY_MEDIUM;
dma_init_struct.loop_mode_enable = FALSE;
dma_init(DMA2_CHANNEL1, &dma_init_struct);

/* enable transfer full data intterrupt */
dma_interrupt_enable(DMA2_CHANNEL1, DMA_FDT_INT, TRUE);

/* dma2 channel1 interrupt nvic init */
nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);
nvic_irq_enable(DMA2_Channel1_IRQn, 1, 0);

/* tmr2 flexible function enable */
dma_flexible_config(DMA2, FLEX_CHANNEL1, DMA_FLEXIBLE_TMR2_OVERFLOW);
```

In this example, the update interrupt of TIMER1 is configured as flexible DMA mapping request.

3 Example code

Taking AT32F403A as an example, the codes= related to DMA flexible mapping function is included in the BSP, available in the

AT32F403A_407_Firmware_Library_V2.x.x\project\at_start_f403a\examples\dma\data_to_gpio_flexible.

The subsequent section presents how to use this example.

- data_to_gpio_flexible

3.1 data_to_gpio_flexible

This example shows how to transfer SRAM data to the output register of GPIO port through DMA so as to control the GPIO port output. The TMER2 is configured to generate an overflow interrupt and a DMA request, and the secondary DMA request is configured as flexible mapping mode. Every time TIMER2 generate a DMA request, the DMA transfers a group of data from SRAM to GPIO port.

DMA is configured as follows:

```
int main(void)
{
  system_clock_config();

  at32_board_init();
  /* Enable dma2/gpioc/tmr2 clocks*/
  crm_periph_clock_enable(CRM_DMA2_PERIPH_CLOCK, TRUE);
  crm_periph_clock_enable(CRM_GPIOC_PERIPH_CLOCK, TRUE);
  crm_periph_clock_enable(CRM_TMR2_PERIPH_CLOCK, TRUE);

  /* Initialize GPIO port */
  gpio_init_struct.gpio_pins = GPIO_PINS_ALL;
  gpio_init_struct.gpio_mode = GPIO_MODE_OUTPUT;
  gpio_init_struct.gpio_out_type = GPIO_OUTPUT_PUSH_PULL;
  gpio_init_struct.gpio_pull = GPIO_PULL_NONE;
  gpio_init_struct.gpio_drive_strength = GPIO_DRIVE_STRENGTH_STRONGER;
  gpio_init(GPIOC, &gpio_init_struct);

  /* Initialize TMR2 */
  tmr_base_init(TMR2, 0xFF, 0);
  tmr_cnt_dir_set(TMR2, TMR_COUNT_UP);
  /* Enable TMR2 overflow DMA request */
  tmr_dma_request_enable(TMR2, TMR_OVERFLOW_DMA_REQUEST, TRUE);
```

```
  /* Configure DMA2 channel 1 for data transfer */
  dma_reset(DMA2_CHANNEL1);
  dma_init_struct.buffer_size = BUFFER_SIZE;
  dma_init_struct.direction = DMA_DIR_MEMORY_TO_PERIPHERAL;
  dma_init_struct.memory_base_addr = (uint32_t)src_buffer;
  dma_init_struct.memory_data_width = DMA_MEMORY_DATA_WIDTH_HALFWORD;
  dma_init_struct.memory_inc_enable = TRUE;
  dma_init_struct.peripheral_base_addr = (uint32_t)0x4001100C;
  dma_init_struct.peripheral_data_width = DMA_PERIPHERAL_DATA_WIDTH_HALFWORD;
  dma_init_struct.peripheral_inc_enable = FALSE;
  dma_init_struct.priority = DMA_PRIORITY_MEDIUM;
  dma_init_struct.loop_mode_enable = FALSE;
  dma_init(DMA2_CHANNEL1, &dma_init_struct);

  /* enable transfer full data intterrupt */
  dma_interrupt_enable(DMA2_CHANNEL1, DMA_FDT_INT, TRUE);

  /* dma2 channel1 interrupt nvic init */
  nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);
  nvic_irq_enable(DMA2_Channel1_IRQn, 1, 0);
  /* Configure DMA flexible function */
  dma_flexible_config(DMA2, FLEX_CHANNEL1, DMA_FLEXIBLE_TMR2_OVERFLOW);
  /* Enable DMA channel */
  dma_channel_enable(DMA2_CHANNEL1, TRUE);
  /* Enable tmr2 */
  tmr_counter_enable(TMR2, TRUE);
  while(1)
  {
  }
}
```

The test results can be viewed by fetching the GPIO port data with a logic analyzer.

4 Revision history

Table 4. Document revision history

Date	Version	Revision note
2021.12.21	2.0.0	Initial release

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers are solely responsible for the selection and use of ARTERY's products and services; ARTERY assumes no liability for purchasers' selection or use of the products and the relevant services.

No license, express or implied, to any intellectual property right is granted by ARTERY herein regardless of the existence of any previous representation in any forms. If any part of this document involves third party's products or services, it does NOT imply that ARTERY authorizes the use of the third party's products or services, or permits any of the intellectual property, or guarantees any uses of the third party's products or services or intellectual property in any way.

Except as provided in ARTERY's terms and conditions of sale for such products, ARTERY disclaims any express or implied warranty, relating to use and/or sale of the products, including but not restricted to liability or warranties relating to merchantability, fitness for a particular purpose (based on the corresponding legal situation in any unjudicial districts), or infringement of any patent, copyright, or other intellectual property right.

ARTERY's products are not designed for the following purposes, and thus not intended for the following uses: (A) Applications that have specific requirements on safety, for example: life-support applications, active implant devices, or systems that have specific requirements on product function safety; (B) Aviation applications; (C) Auto-motive application or environment; (D) Aerospace applications or environment, and/or (E) weapons. Since ARTERY products are not intended for the above-mentioned purposes, if purchasers apply ARTERY products to these purposes, purchasers are solely responsible for any consequences or risks caused, even if any written notice is sent to ARTERY by purchasers; in addition, purchasers are solely responsible for the compliance with all statutory and regulatory requirements regarding these uses.

Any inconsistency of the sold ARTERY products with the statement and/or technical features specification described in this document will immediately cause the invalidity of any warranty granted by ARTERY products or services stated in this document by ARTERY, and ARTERY disclaims any responsibility in any form.