AN0009
应用笔记

DMA with Flexible Mapping

前言

本使用指南描述了怎么使用DMA弹性映射请求，使得DMA请求配置更加灵活。此功能在AT32部分型号上支持，使用时需要注意是否可使用在当前型号。

支持型号列表：

支持型号	AT32F 系列

目录

表目录

图目录

1 概述

在使用 Artery 部分系列 MCU 时（如 AT32F413\ AT32F415\ AT32F403A\ AT32F407），可以使用 DMA 弹性映射功能。此功能使得 DMA 的通道配置更加灵活，可以将某外设的 DMA 请求通道指定到 DMA1 或者 DMA2 共 14 个通道中的任意一个通道。（如：可以将 SPI1 接受数据的 DMA 请求指定到 DMA1 的通道 7）。

本指南将介绍如何使用 DMA 弹性映射请求，从而使得 DMA 传输变得更加灵活多变。

2 配置及使用方法

2.1 常规 DMA 使用（DMA 固定映射）

常规的 DMA 使用以及配置方式为：外设的 DMA 通道已经固定且不可改变，使用时配置好再使能固定通道即可。这就意味着如果想开启某个外设的 DMA 功能，那么通道是不可改变的，例如想使用 SPI1 的 RX DMA 功能，那么就要查看 RM 的手册，如下：

表 1. DMA 固定映射请求

外设	通道 **1**	通道 **2**	通道 **3**	通道 **4**	…	通道 **7**
ADC1	ADC1				…	
SPI/I^2S		SPI1/I2S1_RX	SPI1/I2S1_TX	SPI2/I2S2_RX	…	
…	…	…	…	…	…	…
TMR4	TMR4_CH1			TMR4_CH2	…	TMR4_OVERFLOW

从表格中可以知道需要开启DMA1的通道2。

2.2 DMA 弹性映射使用

DMA 弹性映射请求功能提供了一种更灵活的使用方式，即外设的 DMA 通道不固定，可选择 DMA1 和 DMA2 中，共 14 个通道的任意一个通道。
想要使用此功能，需要通过以下几步的设定：

1) 开启 DMA 弹性映射功能
将 DMA 的通道来源寄存器 1 的第 24bit 写 1，即 DMA_SRC_SEL1 寄存器的 DMA_FLEX_EN 位。
向通道设置对应的寄存器中写入相应的硬件 ID 号
每个外设的 DMA 请求都分配了一个硬件 ID 号，只要将这个 ID 号写进通道来源寄存器中即可。ID 号可查看 RM 中的表格，以 403A 为例，如下：

表 2. 403A 各个信道的 DMA 弹性请求一览

CHx_SRC	请求来源	CHx_SRC	DMA 来源	CHx_SRC	请求来源	CHx_SRC	请求来源
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
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

CHx_SRC	请求来源	CHx_SRC	DMA 来源	CHx_SRC	请求来源	CHx_SRC	请求来源
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_OVERFLOW
56	TMR1_CH1	57	TMR1_CH2	58	TMR1_CH3	59	TMR1_CH4
60	reserved	61	TMR2_TRIG	62	reserved	63	TMR2_OVERFLOW
64	TMR2_CH1	65	TMR2_CH2	66	TMR2_CH3	67	TMR2_CH4
68	reserved	69	TMR3_TRIG	70	reserved	71	TMR3_OVERFLOW
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_OVERFLOW
88	TMR5_CH1	89	TMR5_CH2	90	TMR5_CH3	91	TMR5_CH4
92	reserved	93	reserved	94	reserved	95	TMR6_OVERFLOW
96	reserved	97	reserved	98	reserved	99	reserved
100	reserved	101	reserved	102	reserved	103	TMR7_OVERFLOW
104	reserved	105	reserved	106	reserved	107	Reserved
108	reserved	109	TMR8_TRIG	110	TMR8_HALL	111	TMR8_OVERFLOW
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
164	reserved	165	reserved	166	reserved	167	reserved
168	reserved	169	reserved	170	reserved	171	reserved
172	reserved	173	reserved	174	reserved	175	reserved

上表中的 CHx_SRC 设定值就是硬件 ID 号，将这个 ID 号写进通道来源寄存器中的对应通道 bit 位就可以了。例如：要将 SPI1 的 RX 的 DMA 请求映射到 DMA1 的通道 7，那么就要将 0x09 写入到 DMA_SRC_SEL1 寄存器的 CH7_SRC[23:16]。其他配置与常规 DMA 配置相同通过以上 3 步的配置，弹性映射功能即可使用。

注：DMA1/2 的 DMA_FLEX_EN 必须要同时设定为 1 或 0 时，DMA1/2 的映像模式必须一致。无法 DMA1 是固定式映像，DMA2 是弹性式映像。

2) DMA 弹性映射库函数使用

以上的配置在BSP中的dma.h\dma.c的库文件中有提供相应的库函数，使用者只需调用库函数即可完成DMA弹性映射模式的配置。

库函数说明如下：

表 3. DMA 弹性映射库函数说明

void DMA_Flexible_Config(DMA_Type *DMAx,uint8_t Flex_Channelx,uint8_t Hardware_ID);	
参数	说明
DMAx	DMA1或者DMA2
Flex_Channelx	选择通道（ch1到ch7）
Hardware_ID	对应的硬件ID号

此函数只需在配置好DMA常规功能后调用即可，如下：

图 1. DMA 固定映射库函数调用范例

```
/* 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);
```

上图中为设置TIMER1的更新中断为DMA弹性映射请求范例。

3 例程说明

DMA弹性映射功能在BSP中例程，路径为：

AT32F403A_407_Firmware_Library_V2.x.x\project\at_start_f403a\examples\dma\data_to_gpio_flexible（以403A路径为例）

下面将对这两个例程做一个使用说明。

- data_to_gpio_flexible

3.1 例程 data_to_gpio_flexible

本例程实现的功能为利用DMA将SRAM的数据传输到GPIO口的输出寄存器中，从而达到控制GPIO口输出的目的。同时配置TMER2产生overflow中断并产生DMA请求，配置次DMA请求为弹性映射模式。TIMER2每产生一次DMA请求，DMA就从SRAM搬运一笔数据到GPIO口。

DMA相关的配置代码：

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

  at32_board_init();
  /* 使能dma2/gpioc/tmr2 时钟*/
  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);

  /* 初始化GPIO口 */
  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);

  /* 初始化TMR2 */
  tmr_base_init(TMR2, 0xFF, 0);
  tmr_cnt_dir_set(TMR2, TMR_COUNT_UP);
  /* 打开TMR2溢出DMA请求 */
  tmr_dma_request_enable(TMR2, TMR_OVERFLOW_DMA_REQUEST, TRUE);

  /* 配置DMA2通道1作为数据传输*/
```

```
  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);
  /*  配置DMA弹性功能  */
  dma_flexible_config(DMA2, FLEX_CHANNEL1, DMA_FLEXIBLE_TMR2_OVERFLOW);
  /*  使能DMA通道  */
  dma_channel_enable(DMA2_CHANNEL1, TRUE);
  /*  使能tmr2 */
  tmr_counter_enable(TMR2, TRUE);
  while(1)
  {
  }
}
```

实验结果可采用逻辑分析仪抓取GPIO口数据查看。

4 版本历史

表 4. 文档版本历史

日期	版本	变更
2021.12.21	2.0.0	最初版本

重要通知 – 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途(及其依据任何司法管辖区的法律的对应情况)，或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：(A) 对安全性有特别要求的应用，如：生命支持、主动植入设备或对产品功能安全有要求的系统；(B) 航空应用；(C) 汽车应用或汽车环境；(D) 航天应用或航天环境，且/或(E) 武器。因雅特力产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。