Insper

14 - DMA - Trabalhe por mim !

Rafael Corsi - rafael.corsi@insper.edu.br

Maio - 2017

1 Introdução

Modificar o código 13 - UART para automatizarmos a transferência de dados entre o CORE e o periférico USART, nessa modificação a transferência dos dados será realizada inteiramente pelo DMA. Para isso será necessário as seguintes configurações no código :

Favor copiar o código [13-UART] para –> [14 - DMA] e salvar no seu repositório do git

1.1 Inserir via ASF a biblioteca do XDMA

1. Atmel Studio –> ASF –> ASF Wizard
2. Adicionar o módulo XDMA

1.2 Inserir os defines a seguir no main.c

```
/*****************************************************************/
/* XDMA CONFIG                                                   */
/*****************************************************************/

/** XDMAC channel used in this example. */
#define XDMAC_TX_CH 0
#define XDMAC_RX_CH 1
```

```
/** XDMAC channel HW Interface number for SPI0,refer to datasheet. */
#define USART_XDMAC_TX_CH_NUM
#define USART_XDMAC_RX_CH_NUM

/** XDMA peripheral interface*/
/** XDMA peripheral interface*/
#define USART_XDMA_DEST_REG    &CONSOLE_UART->US_THR // Mem Two Peripheral
#define USART_XDMA_ORIG_REG    &CONSOLE_UART->US_RHR // Peripheral Two Mem

/** The buffer size for transfer */
#define BUFFER_SIZE          100
```

Note que definimos aqui os endereços do periférico USART onde o DMA irá escrever (USART_XDMA_DEST_REG) e ler (USART_XDMA_ORIG_REG).

1.2.1 Datasheet XDMA

Pesquisar no datasheet (pg. 480), os canais relativos ao **USART1 TX** e **USART1 RX** e completar os defines :

- **USART_XDMAC_TX_CH_NUM**, e
- **USART_XDMAC_RX_CH_NUM**

1.3 Variáveis globais

Adicione as seguintes variáveis globais no main.c :

```
/*----------------------------------------------------------------------------*/
/* VAR globais                                                                */
/*----------------------------------------------------------------------------*/

/** XDMAC channel configuration. */
static xdmac_channel_config_t g_xdmac_tx_cfg;
static xdmac_channel_config_t g_xdmac_rx_cfg;

/**
 * XDMA
 */
uint8_t  g_tx_buffer[]             = "This is message from USART master
                                      transferred by XDMAC test \n";
uint8_t  g_rx_buffer[BUFFER_SIZE]  = "0";
uint32_t g_buffer_size             = sizeof(g_tx_buffer);
```

```
/*----------------------------------------------------------------------------*/
/* Flags                                                                      */
/*----------------------------------------------------------------------------*/

volatile uint8_t flag_led0  = 1;
volatile uint8_t flag_rx    = 0;
```

Aqui definimos :

- g_xdmac_tx_cfg (rx) : Structure com definições de configuração para o DMA
- g_tx_buffer[] : Vetor alocado em memória com o texto a ser enviado pelo o DMA para o periféico
- g_rx_buffer[BUFFER_SIZE] : Vetor alocado em memória para recebimento dos dados transferidos via DMA do periférico USART1
- g_buffer_size : tamanho em bytes, do vetor de transmissão
- flag_rx : irá indicar que o DMA de recepção foi finalizado
- flag_led0 : flag com o status do led

1.4 Funções

Inclua e utilize as seguintes funções no código para operar com o DMA.

```
static void uart_xdmac_configure();

static void uart_xdmac_Tx(uint32_t *dest_address,
                          uint32_t *orgin_address,
                          uint32_t buffer_size);

static void uart_xdmac_Rx(uint32_t *peripheral_address,
                          uint32_t *orgin_address,
                          uint32_t buffer_size);
```

1.4.1 Configuração do DMA

Configura o DMA para operar em modo Memória -> Periférico com interrupção na finalização das transmissões (rx e tx).

```
static void uart_xdmac_configure()
{
    uint32_t xdmaint;

    /* Initialize and enable DMA controller */
    pmc_enable_periph_clk(ID_XDMAC);
```

```
    xdmaint = ( XDMAC_CIE_BIE   |
                XDMAC_CIE_DIE   |
                XDMAC_CIE_FIE   |
                XDMAC_CIE_RBIE  |
                XDMAC_CIE_WBIE  |
                XDMAC_CIE_ROIE);

    xdmac_channel_enable_interrupt(XDMAC, XDMAC_TX_CH, xdmaint);
    xdmac_enable_interrupt(XDMAC, XDMAC_TX_CH);

    xdmac_channel_enable_interrupt(XDMAC, XDMAC_RX_CH, xdmaint);
    xdmac_enable_interrupt(XDMAC, XDMAC_RX_CH);

    /*Enable XDMAC interrupt */
    NVIC_ClearPendingIRQ(XDMAC_IRQn);
    NVIC_SetPriority( XDMAC_IRQn ,1);
    NVIC_EnableIRQ(XDMAC_IRQn);
}
```

1.4.2 Envio de dados via DMAC

Função que deve ser chamada para inicializar a transferência do DMA (Memória –> Periférico), exemplo de chamada :

```
uart_xdmac_Tx(USART_XDMA_DEST_REG, (uint32_t) g_tx_buffer, g_buffer_size);
```

Nesse caso, é configurado uma transferência do endereço g_tx_buffer de tamanho g_buffer_size para o registrador USART_XDMA_DEST_REG.

```
/*
45.6.10.12 / pg. 1185
The DMA uses the trigger flags, TXRDY and RXRDY, to write or read into the USART. The DM
the Transmit Holding register (US_THR) and it always reads in the Receive Holding regist
of the data written or read by the DMA in the USART is always a byte
*/
static void uart_xdmac_Tx(uint32_t *peripheral_address,
                          uint32_t *orgin_address,
                          uint32_t buffer_size)
{

    /* Initialize channel config for transmitter */
    g_xdmac_tx_cfg.mbr_ubc = buffer_size;
    g_xdmac_tx_cfg.mbr_sa  =  (uint32_t)orgin_address;
    g_xdmac_tx_cfg.mbr_da  =  (uint32_t)peripheral_address;
```

```
    g_xdmac_tx_cfg.mbr_cfg =  XDMAC_CC_TYPE_PER_TRAN |
                              XDMAC_CC_MBSIZE_SINGLE |
                              XDMAC_CC_DSYNC_MEM2PER |
                              XDMAC_CC_CSIZE_CHK_1 |
                              XDMAC_CC_DWIDTH_BYTE |
                              XDMAC_CC_SIF_AHB_IF0 |
                              XDMAC_CC_DIF_AHB_IF1 |
                              XDMAC_CC_SAM_INCREMENTED_AM |
                              XDMAC_CC_DAM_FIXED_AM |
                              XDMAC_CC_PERID(USART_XDMAC_TX_CH_NUM);

    g_xdmac_tx_cfg.mbr_bc  = 0;
    g_xdmac_tx_cfg.mbr_ds  = 0;
    g_xdmac_tx_cfg.mbr_sus = 0;
    g_xdmac_tx_cfg.mbr_dus = 0;

    xdmac_configure_transfer(XDMAC, XDMAC_TX_CH, &g_xdmac_tx_cfg);
    xdmac_channel_set_descriptor_control(XDMAC, XDMAC_TX_CH, 0);
    xdmac_channel_enable(XDMAC, XDMAC_TX_CH);

}
```

1.4.3 Recepção de dados via DMA

Função que deve ser chamada para inicializar a transferência do DMA (Periférico –> Memória), exemplo de chamada :

```
uart_xdmac_Rx(USART_XDMA_ORIG_REG, (uint32_t) g_rx_buffer, g_buffer_size);
```

Nesse caso, é configurado uma transferência do endereço de registrador do periférico USART_XDMA_ORIG_REG para o buffer em memória : g_rx_buffer de tamanho g_buffer_size.

```
static void uart_xdmac_Rx(uint32_t *peripheral_address,
                          uint32_t *orgin_address,
                          uint32_t buffer_size)
{
    /* Initialize channel config for receiver */
    g_xdmac_rx_cfg.mbr_ubc = buffer_size;
    g_xdmac_rx_cfg.mbr_da = (uint32_t)orgin_address;
    g_xdmac_rx_cfg.mbr_sa = (uint32_t)peripheral_address;

    g_xdmac_rx_cfg.mbr_cfg = XDMAC_CC_TYPE_PER_TRAN |
```

```
                                XDMAC_CC_MBSIZE_SINGLE |
                                XDMAC_CC_DSYNC_PER2MEM |
                                XDMAC_CC_CSIZE_CHK_1 |
                                XDMAC_CC_DWIDTH_BYTE|
                                XDMAC_CC_SIF_AHB_IF1 |
                                XDMAC_CC_DIF_AHB_IF0 |
                                XDMAC_CC_SAM_FIXED_AM |
                                XDMAC_CC_DAM_INCREMENTED_AM |
                                XDMAC_CC_PERID(USART_XDMAC_RX_CH_NUM);

    g_xdmac_rx_cfg.mbr_bc = 0;
    g_xdmac_tx_cfg.mbr_ds =  0;
    g_xdmac_rx_cfg.mbr_sus = 0;
    g_xdmac_rx_cfg.mbr_dus =0;

    xdmac_configure_transfer(XDMAC, XDMAC_RX_CH, &g_xdmac_rx_cfg);
    xdmac_channel_set_descriptor_control(XDMAC, XDMAC_RX_CH, 0);
    xdmac_channel_enable(XDMAC, XDMAC_RX_CH);
}
```

1.4.4 Handler

Toda transferência (completa ou parcial) de um DMA o handler a seguir é chamado:

```
/**
 * \brief XDMAC interrupt handler.
 */
void XDMAC_Handler(void)
{
  uint32_t dma_status_tx, dma_status_rx;

  dma_status_tx = xdmac_channel_get_interrupt_status(XDMAC, XDMAC_TX_CH);
  dma_status_rx = xdmac_channel_get_interrupt_status(XDMAC, XDMAC_RX_CH);

  UNUSED(dma_status_tx);
  UNUSED(dma_status_rx);

  // Verificamos se a transferência foi completa
  if(dma_status_rx & (XDMAC_CIS_BIS | XDMAC_CIS_LIS)){
    flag_rx = 1;
  }
}
```

- Dependendo do tamanho da transferência do DMA (se ele exceder o tamanho do

buffer interno) o DMA irá fragmentar a transferência em diversos pacotes.

2 Desafio

O desafio da aula é implementar duas funções usart_puts_dma() e usart_gets_dma() de modo não blocante, ou seja, pelo uso de DMA.

A função usart_gets_dma deverá ser um pouco diferente da implementando anteriormente, essa função necessitará de um parâmetro extra que informa o tamanho da string a ser recebida, exemplo de implementação :

```
void sart\_gets_dma(uint8_t *pstring, int nChars)
```

onde **nChars** indica o tamanho de dados a serem recebidos. Como é uma função não blocante o código deverá ficar observando a flag flag_rx para verificar a finalização da recepção.

Passos de execução

1. Implemente as funções e variáveis descritas nesse documento.
2. Busque no datasheet as informações sobre o DMA e insira no #define (sec: 1.2)
3. (TX) Implemente a função usart_puts_dma(. . .)
4. (RX) Implemente a função usart_gets_dma(. . .)