本章教程讲解DMA存储器到存储器模式。存储器到存储器模式可以实现数据在两个内存的快速拷贝。程序中，首先定义一个静态的源数据，存放在内部FLASH，然后使用DMA传输把源数据拷贝到目标地址上（内部SRAM），最后对比源数据和目标地址的数据，判断传输是否准确。

1、DMA简介及相关函数介绍

直接存储器访问控制器（DMA）提供在外设和存储器之间或者存储器和存储器之间的高速数据传输方式，无须CPU干预，数据可以通过DMA快速地移动，以节省CPU的资源来做其他操作。

DMA控制器有7个通道，每个通道专门用来管理来自于一个或多个外设对存储器访问的请求。还有一个仲裁器来协调各通道之间的优先级。

关于DMA具体信息，可参考CH32V103应用手册。DMA标准库函数具体内容如下：

```
1. void DMA_DeInit(DMA_Channel_TypeDef* DMAy_Channelx);
2. void DMA_Init(DMA_Channel_TypeDef* DMAy_Channelx,
   DMA_InitTypeDef* DMA_InitStruct);
3. void DMA_StructInit(DMA_InitTypeDef* DMA_InitStruct);
4. void DMA_Cmd(DMA_Channel_TypeDef* DMAy_Channelx,
   FunctionalState NewState);
5. void DMA_ITConfig(DMA_Channel_TypeDef* DMAy_Channelx,
   uint32_t DMA_IT, FunctionalState NewState);
6. void DMA_SetCurrDataCounter(DMA_Channel_TypeDef*
   DMAy_Channelx, uint16_t DataNumber);
7. uint16_t DMA_GetCurrDataCounter(DMA_Channel_TypeDef*
   DMAy_Channelx);
8. FlagStatus DMA_GetFlagStatus(uint32_t DMAy_FLAG);
9. void DMA_ClearFlag(uint32_t DMAy_FLAG);
10. ITStatus DMA_GetITStatus(uint32_t DMAy_IT);
11. void DMA_ClearITPendingBit(uint32_t DMAy_IT);
```

复制代码

1.1、void DMA_DeInit(DMA_Channel_TypeDef* DMAy_Channelx)
功 能：将DMAy Channelx寄存器初始化为其默认重置值。
输 入：DMAy_Channelx：这里y可以是1或2来选择DMA，x可以是1到7对于DMA1，1到5对于DMA2选择DMA频道。

1.2、void DMA_Init(DMA_Channel_TypeDef* DMAy_Channelx, DMA_InitTypeDef* DMA_InitStruct)
功 能：根据DMA_InitStruct中指定的参数初始化DMAy Channelx。
输 入：DMAy_Channelx：这里y可以是1或2来选择DMA，x可以是1到7对于DMA1，1到5对于DMA2选择DMA通道。DMA_InitStruct：指向包含指定DMA频道的配置信息的DMA_InitTypeDef结构的指针。

1.3、void DMA_StructInit(DMA_InitTypeDef* DMA_InitStruct)
功 能：用默认值填充每个DMA_InitStruct成员。
输 入：DMA_InitStruct：指向要初始化的DMA_InitTypeDef结构的指针，对于DMA1，它是1到7，对于DMA2是1到5。

1.4、void DMA_Cmd(DMA_Channel_TypeDef* DMAy_Channelx, FunctionalState NewState)
功 能：启用或禁用指定的DMAy Channelx。
输 入：DMAy_Channelx：其中y可以是1或2来选择DMA，x可以是1到7（对于DMA1）和1到5（对于DMA2）来选择DMA通道。NewState：DMAy Channelx的新状态（启用或禁用）。

1.5、void DMA_ITConfig(DMA_Channel_TypeDef* DMAy_Channelx, uint32_t DMA_IT, FunctionalState NewState)
功 能：启用或禁用指定的DMAy Channelx中断。
输 入：DMAy_Channelx：其中y可以是1或2来选择DMA，x可以是1到7（对于DMA1）和1到5（对于DMA2）来选择DMA通道。NewState：DMAy Channelx的新状态（启用或禁用）。

1.6、void DMA_SetCurrDataCounter(DMA_Channel_TypeDef* DMAy_Channelx, uint16_t DataNumber)
功 能：设置当前DMAy Channelx传输中的数据单元数。
输 入：DMAy_Channelx：其中y可以是1或2来选择DMA，x可以是1到7（对于DMA1）和1到5（对于DMA2）来选择DMA通道。DataNumber：当前DMAy Channelx传输中的数据单元数。

1.7、uint16_t DMA_GetCurrDataCounter(DMA_Channel_TypeDef* DMAy_Channelx)
功 能：返回当前DMAy Channelx传输中剩余的数据单元数。
输 入：DMAy_Channelx：其中y可以是1或2来选择DMA，x可以是1到7（对于DMA1）和1到5（对于DMA2）来选择DMA通道。

1.8、FlagStatus DMA_GetFlagStatus(uint32_t DMAy_FLAG)
功 能：检查是否设置了指定的DMAy Channelx标志。
输 入：DMAy_FLAG：指定要检查的标志。

1.9、void DMA_ClearFlag(uint32_t DMAy_FLAG)
功 能：清除DMAy Channelx的挂起标志。
输 入：DMAy_FLAG：指定要检查的标志。

1.10、ITStatus DMA_GetITStatus(uint32_t DMAy_IT)
功 能：检查指定的DMAy Channelx中断是否发生。
输 入：DMAy_IT：指定要检查的DMAy中断源。

1.11、void DMA_ClearITPendingBit(uint32_t DMAy_IT)
功 能：清除DMAy Channelx的中断挂起位。
输 入：DMAy_IT：指定要清除的DMAy中断源。

以上函数在程序编写时直接调用即可。

2、硬件设计

本章教程所用资源均为CH32V103开发板内部资源，无需进行其他硬件连接，只需进行程序配置即可。

3、软件设计

DMA存储器到存储器模式程序如下：
dma.h文件

```
#ifndef __DMA_H
#define __DMA_H

#include "ch32v10x_conf.h"

/* Global define */
#define Buf_Size 32       //定义要发送的数据大小

/* Global Variable */
u32 SRC_BUF[Buf_Size];    //定义SRC_BUF数组作为DMA传输数据源
u32 DST_BUF[Buf_Size];    //定义DMA传输目标存储器，存储在内部SRAM中

void DMA1_CH3_Init(void); //DMA传输参数初始化配置
u8 BufCmp( u32* buf1,u32* buf2,u16 buflength); //源数据与目标地址数据对比

#endif
```

复制代码
dma.h文件主要包含相关定义及函数声明。

dma.c文件

```
#include "dma.h"

//定义SRC_BUF数组作为DMA传输数据源，并将数据存储在内部FLASH中
u32 SRC_BUF[Buf_Size]=
                    {
 0x01020304,0x05060708,0x090A0B0C,0x0D0E0F10,
   0x11121314,0x15161718,0x191A1B1C,0x1D1E1F20,
   0x21222324,0x25262728,0x292A2B2C,0x2D2E2F30,
   0x31323334,0x35363738,0x393A3B3C,0x3D3E3F40,
   0x41424344,0x45464748,0x494A4B4C,0x4D4E4F50,
   0x51525354,0x55565758,0x595A5B5C,0x5D5E5F60,
   0x61626364,0x65666768,0x696A6B6C,0x6D6E6F70,
   0x71727374,0x75767778,0x797A7B7C,0x7D7E7F80
                    };
//定义DST_BUF数组作为DMA传输目标存储器，并将数据存储在内部SRAM中
u32 DST_BUF[Buf_Size]={0};

/**************************************************************************************
* Function Name  : DMA1_CH3_Init
* Description    : Initializes Channel3 of DMA1 collection.
* Input          : None
* Return         : None
**************************************************************************************/
//DMA1通道3传输参数配置
void DMA1_CH3_Init(void)
{
  DMA_InitTypeDef DMA_InitStructure;
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
    //使能DMA1时钟

  DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)(SRC_BUF);
  //设置源数据地址
  DMA_InitStructure.DMA_MemoryBaseAddr = (u32)DST_BUF;
   //设置目标地址
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
  //设置传输方向：外设到存储器（此处外设为内部FLASH）
  DMA_InitStructure.DMA_BufferSize = Buf_Size*4;                //设置传输大小
```

```
33.   DMA_InitStructure.DMA_PeripheralInc =
    DMA_PeripheralInc_Enable;         //指定外设地址寄存器递增。
34.   DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
                //指定内存地址寄存器递增。
35.   DMA_InitStructure.DMA_PeripheralDataSize =
    DMA_PeripheralDataSize_Byte; //设置外设数据单位
36.   DMA_InitStructure.DMA_MemoryDataSize =
    DMA_PeripheralDataSize_Byte;     //设置存储器数据单位
37.   DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
     //设置对应DMA工作模式为正常模式
38.   DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;
    //DMA1通道3优先级高
39.   DMA_InitStructure.DMA_M2M = DMA_M2M_Enable;
    //使能DMA存储器到存储器的传输方式
40.   DMA_Init(DMA1_Channel3, &DMA_InitStructure);                  //根
    据DMA_InitStruct中指定的参数初始化DMA1通道3
41.   DMA_ClearFlag(DMA1_FLAG_TC3);   //清除DMA1通道3传输完成挂
    起标志
42.
43.   DMA_Cmd(DMA1_Channel3, ENABLE); //使能DMA1通道3
44. }
45.
46. /*****************************************************************************
47. * Function Name  : BufCmp
48. * Description    : Compare the  buf
49. * Input          : buf1:pointer of buf1
50. *                  buf2:pointer of buf1
51. *                  buflength: length of buf
52. * Return         : 1:Two arrays are identical
53. *                  0:Two arrays are inconsistent
54. *****************************************************************************/
55. //源数据与目标地址数据对比
56. u8 BufCmp( u32* buf1,u32* buf2,u16 buflength)
57. {
58.   while(buflength--)
59.     {
60.       if(*buf1 != *buf2)  //判断两个数据源是否对应相等
61.         {
62.           return 0;       //对应数据源不相等马上退出函数，并返回 0
63.         }
64.         /* 递增两个数据源的地址指针 */
65.         buf1++;
66.         buf2++;
67.     }
68.     return 1;             //完成判断并且对应数据相对 ,返回1
69. }
70.
```

复制代码
dma.c文件主要包括DMA1通道3传输参数初始化配置以及数据比较两个函数，DMA1_CH3_Init函数主要进行DMA1时钟使能、DMA1通道3参数配置、DMA1通道3传输完成挂起标志清除以及DMA1通道3使能。BufCmp函数主要进行数据对比，并根据对比结果返回相应值。

main.c函数

```
/*******************************************************************************
* Function Name  : main
* Description    : Main program.
* Input          : None
* Return         : None
*******************************************************************************/
int main(void)
{
    u8 i=0;
    u8 Flag=0;

    Delay_Init();
    USART_Printf_Init(115200);
    DMA1_CH3_Init();

    printf("SystemClk:%d\r\n",SystemCoreClock);
    printf("DMA MEM2MEM TEST\r\n");

    while(DMA_GetFlagStatus(DMA1_FLAG_TC3)==RESET) //等待DMA1通道3传输完成
    {
    }
    Flag=BufCmp(SRC_BUF,DST_BUF,Buf_Size);        //源数据与传输后数据比较结果
    if(Flag==0)                                   //判断源数据与传输后数据比较结果
    {
      printf("DMA Transfer Fail\r\n");
    }
    else
    {
      printf("DMA Transfer Success\r\n");
    }

    printf("SRC_BUF:\r\n");
    for(i=0;i<Buf_Size;i++)
    {
        printf("0x%08x\r\n",SRC_BUF[i]);
    }
```

```
37.
38.     printf("DST_BUF:\r\n");
39.     for(i=0;i<Buf_Size;i++)
40.     {
41.         printf("0x%08x\r\n",DST_BUF[i]);
42.     }
43.     while(1)
44.     {
45.     }
46. }
```

复制代码
main.c文件主要进行相关函数初始化以及DMA传输状态及结果显示。

4、下载验证

将编译好的程序下载到开发板并复位，串口打印情况具体如下：