# mini2440裸机试炼之——DMA直接存取 实现Uart（串口）通信

**实现功能:**

将字符串数据通过DMA0通道传递给UTXH0，然后在终端

显示。数据传输完后，DMA0产生中断，beep声, LED亮。

**DMA基本知识**

计算机系统中各种常用的数据输入/输出方法有查询方式(包括无条件及条件传送方式)和中断方式，这些方式适用于CPU与慢速及中速外设之间的数据交换。但当高速外设要与系统内存或者要在系统内存的不同区域之间进行大量数据的快速传送时，就在一定程度上限制了数据传送的速率。直接存储器存取(DMA)就是为解决这个问题提出的，采用DMA方式，在一定时间段内，由DMA控制器取代CPU，获得总线控制权，来实现内存与外设或者内存的不同区域之间大量数据的快速传送。

**DMA的概述：**

SC2440支持位于系统总线与外围总线之间的四通道DMA控制。每一通道的DMA都可以处理一下四种情况：

1.源和目的器件均可以在系统总线

2.源器件在系统总线而目的器件在外围总线

3.源器件在外围总线而目的器件在系统总线

4.源和目的器件均可以在外围总线

DMA最大的有点就是可以在没有CPU干涉的情况下进行数据的传送。可以通过软件控制DMA启动，或者通过内部请求或者外部请求引脚启动。

**DMA控制**

DMA使用三态的有限状态机FSM (Finite State Machine)，有限状态机制)对其进行控制，以下用三步进行描述：

状态1：

在初始状态，DMA等待DMA请求。当DMA请求到达时，进入状态2。在这 阶段DMA ACK和INT REQ均为0。

状态2：

在这个阶段，DMA ACK变成1以及计数器CURR\_TC从DCON[19:0]寄存器加载数据。（注意：DMA ACK保持1直到对其清零）

状态3：

在这个阶段，DMA对进行原子操作(atomic operation)的子有限状态机(sub-FSM)进行初始化。sub-FSM从源地址读取数据以后并写进目的地址。在这个操作前，数据的大小和传输的大小均应给予考虑。在整体模式(Whole service mode)下的计数器(CURR\_TC)为0之前，数据传输的操作将会继续。当sub-FSM完成原子操作后，主FSM进行倒计。另外，在计数器CRRR\_TC为0以及中断设置DCON[29]寄存器被置1时，主FSM发出INT REQ信号。除此之外，同时清除DMA ACK。

**DMA寄存器配置**

寄存器DISRCn是初始DMA源寄存器，它是用于设置DMA数据传输的源基址

寄存器DIDSTn是初始DMA目的寄存器，它是用于设置DMA数据传输的目的基址。

控制寄存器DISRCCn的第1位用于选择源的总线（系统总线AHB还是外设总线APB），第0位用于设置源基址在数据传输过程中是递增还是固定不变。

控制寄存器DIDSTCn的低两位与寄存器DISRCCn相识，但它是用来设置目的基址，而第2位用于设置是在传输完数据之后中断还是在自动重载后中断。

控制寄存器DCONn用于控制数据的DMA传输，第31位用于设置传输协议是需求模式还是握手模式，第30位用于选择同步时钟是PCLK还是HCLK，第29位用于设置DMA中断是否发生，第28位用于选择传输大小是单元传输还是突发传输，第27位用于选择服务模式是单步模式还是完全模式，第24位到第26位用于设置DMA的请求源，第23位用于设置DMA的源是软件还是硬件，第22位用于设置是否需要重载传输的目的和源基址，第20位和第21位用于设置数据传输的数据大小（字节、半字还是字），低20位用于初始化传输数据的个数。而通过读取DMA状态寄存器DSTATn的低20位可以获知当前的传输的计数。DMA掩码触发寄存器DMASKTRIGn的第2位可以终止当前DMA操作，第1位可以用于开启DMA通道，第0位则表示在软件请求模式下触发DMA通道。

## 总结

源地址: 字符串SendBuffer的地址

目的地址: uart0的地址

MDA只是实现数据传输的一种机制,桥梁搭建的作用

这里并没实现MDA字符串SendBuffer赋值,而是CPU完成的,MDA只是实现直接从内存(字符串已经被CPU存放于内存)传输到uart0, 字符串Hello mini2440!，长15，每次输出一个字节，要输出十五次，tc就是存放输出的次数每输出一次tc减1，为0时， DMA停止和等待其它DMA请求的单服务模式，如果设置中断则产生MDA中断

所以说tc也就是节拍

## 代码区

Main.c

|  |
| --- |
| #define GLOBAL\_CLK 1  #include "def.h"  #include "option.h"  #include "2440addr.h"  #include "2440lib.h" //函数声明  #include "2440slib.h"  #include "mmu.h"  #include "profile.h"  //函数声明处  extern void DMA\_UART**(**void**);**    void Main**(**void**)**  **{**  U32 mpll\_val **=** 0**,**consoleNum**;**  Port\_Init**();**    mpll\_val **=** **(**92**<<**12**)|(**1**<<**4**)|(**1**);**    //init FCLK=400M,  ChangeMPllValue**((**mpll\_val**>>**12**)&**0xff**,** **(**mpll\_val**>>**4**)&**0x3f**,** mpll\_val**&**3**);**  ChangeClockDivider**(**14**,** 12**);** //the result of rCLKDIVN [0:1:0:1] 3-0 bit  cal\_cpu\_bus\_clk**();** //HCLK=100M PCLK=50M    consoleNum **=** 0**;** // Uart 1 select for debug.  Uart\_Init**(** 0**,**115200 **);**  Uart\_Select**(** consoleNum **);**    MMU\_Init**();**//中断映射地址初始化  Beep**(**2000**,** 100**);**  DMA\_UART**();** //DMA方式实现Uart（串口）通信  **}** |

DMA\_uart.c

|  |
| --- |
| //====================================================================  // 实现功能：  // DMA直接存取 实现Uart（串口）通信,  // 将字符串数据通过DMA0通道传递给UTXH0，然后在终端  // 显示。数据传输完后，DMA0产生中断，beep声, LED亮。  // by：梁惠涌  //====================================================================  #include "2440addr.h"  #include "2440lib.h" //beep函数  char **\***SendBuffer **=** "Hello mini2440!" **;** //source data  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  DMA初始化  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Dma\_init**()**  **{**  rDISRC0**=(**U32**)**SendBuffer**;** //数据地址  rDISRCC0 **|=((**0**<<**1**)|(**0**<<**0**));** //[1]系统总线,[0]地址将根据单次和突发模式中每次传输后其数据大小而增加  rDIDST0**=(**U32**)**UTXH0**;** //传输目标地址UTXH0 {2440addr.h文件中定义 #define UTXH0 (0x50000020) }  rDIDSTC0 **|=((**0**<<**2**)|(**1**<<**1**)|(**1**<<**0**));** //[2]在TC到达0时发生中断, [1] APB(外设总线上) , [0]传输后地址总线不变  //对上述寄存器说明，rDISRC0、rDIDST0分别配置为数据地址(DMA内部)、传输目标地址(uart是属于外设设备)，再相应的配置rDISRCC0、rDIDSTC0    rDCON0 **|=((**U32**)**1**<<**31**)|(**0**<<**30**)|(**1**<<**29**)|(**0**<<**28**)|(**0**<<**27**)|(**1**<<**24**)|(**1**<<**23**)|(**1**<<**22**)|(**0**<<**20**)|(**15**);**  //[31] 握手模式; [30]同步PCLK=50M(APB时钟< 外设 >); [29]当所有传输完成产生中断请求（即CURR\_TC变为0）;[28]执行一个单元传输;  //[27]选择每次原子传输（单次或突发4）后DMA停止和等待其它DMA请求的单服务模式; [24]UART0; [23]选择DMA源触发DMA操作;  //[22]当传输计数的当前值变为0时DMA通道（DMA REQ）关闭; [20]字节(每次一个字节传输);[19:0]初始传输计数15，15=SendBuffer字符数组长度;    rDMASKTRIG0**=(**0**<<**2**)|(**1**<<**1**)|(**0**<<**0**);** //[1]打开DMA通道并且处理此通道DMA请求  **}**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  DMA中断  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void \_\_irq Dma0\_isr**(){**  rSRCPND**|=**0x1**<<**17**;** //清除中断挂起状态  rINTPND**|=**0x1**<<**17**;**  Uart\_Printf**(**"\n \*\*\*DMA TO Uart finished\*\*\*\n"**);**  Beep**(**2000**,**100**);**  rGPBCON**=**0x015400**;**  rGPBDAT**=**0x6**<<**5**;**  **}**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  | DMA子函数  | 需设置2440lib.c里 rUCON0 |=((1<<0) | (1<<3) | (2<<10) );  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void DMA\_UART**(){**  Uart\_Printf**(**"\n \*\*\*HELLO DMA TO Uart \*\*\*\n"**);**  Delay**(**1000**);**    Dma\_init**();**//DMA初始化  rINTMSK **&=~(**1**<<**17**);** //开 dma0 中断  pISR\_DMA0**=(**U32**)**Dma0\_isr**;**      **while(**1**){**  **}**  **}** |

截图：

