AT91SAM7X-EK softpack 1.5 代码解读之 DATAFLASH

——基于 SAM7X-EK 和 IAR EWARM

文档编号	MAN2016A_CH				
文档版本	Rev. A				
文档摘要	基于 AT91SAM7X-EK 开发板的代码解读，SPI DATAFLASH 代码解读				
关键词	AT91SAM7X256 SAM7X-EK IAR EWARM J-LINK DATAFLASH				
创建日期	2010-06-25	创建人员	Cust126	审核人员	Robin
文档类型	公开发布/开发板配套文件				
版权信息	Mcuzone 原创文档，转载请注明出处				

更新历史

版本	时间	更新	作者
Rev. A	2010-06-25	初始创建	Cust126

微控电子 乐微电子
杭州市登云路 639 号 2B143
销售 TEL：86-571-89908193 13957118045
支持 TEL：18913989166 13957118045
FAX：86-571-89908193
www.mcuzone.com www.atarm.com

1.概述

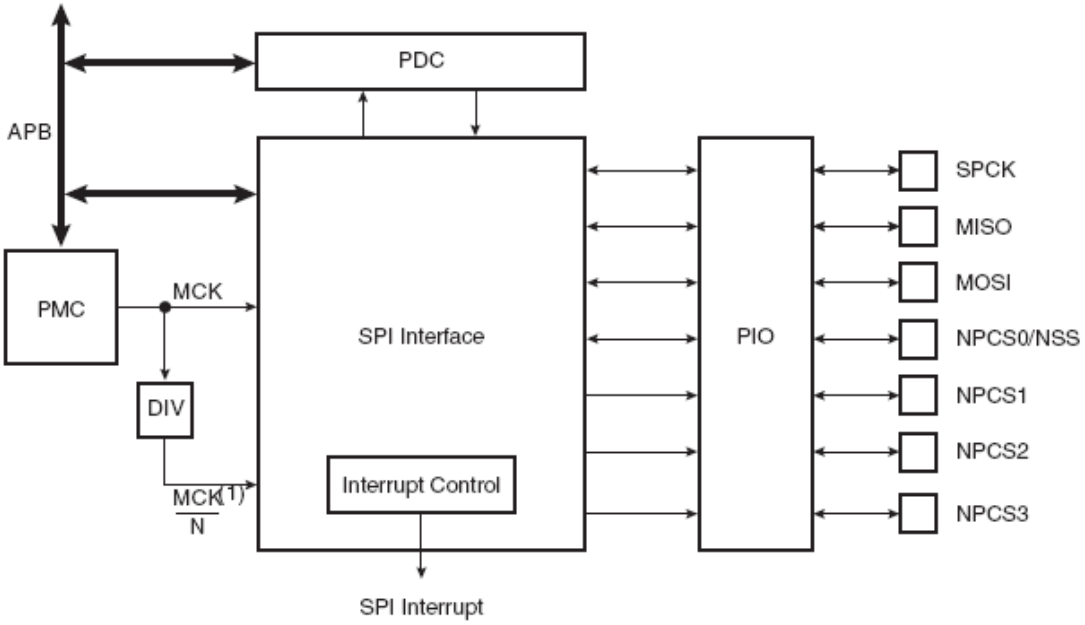
本文档以 SAM7X-EK 为硬件平台，IAR EWARM 为编译器平台，使用 J-Link 作为调试工具，演示并解读 AT91SAM7X256 的 SPI-DATAFLASH 操作流程。

2. SPI 操作

2.1 SPI 操作流程介绍

串行外设接口(SPI)电路是一种可在主机或从机模式下与外部器件进行通讯的数据链接。可通过该接口与外部处理器进行高速通讯。SPI 实质上是一个将串行传输数据位发送到其它 SPI 的移位寄存器。数据传输时，一个 SPI 系统作为“主机”控制数据流，其它 SPI 作为“ 从机" ，主机控制数据的移入与移出。不同的 CPU 可轮流作为主机(多主机协议与单主机协议不同，单主机协议中只有一个 CPU 始终作为主机，其它 CPU 始终作为从机) ，且一个主机可同时将数据移入多个从机。但只允许单个从机将其数据写入主机。

SPI 的方框图：



2.2 PWM 操作的目的与功能描述

AT91 softpack 1.5 的 SAM7X-EK 包内的 basic-sd-dataflash-project 工程的功能是擦除一页并写入数据，再读出看是否正确。

2.3 示例代码

以下代码截取自 AT91 softpack 1.5 的 SAM7X-EK 包内的 basic-sd-spi-project ，并由以上工程改写过来的代码，基于 IAR EWARM 平台

2.3.1 main 函数代码注释解读

下面对 basic-spi-dataflash-project 的主要代码进行改写注释解读，首先是 main 函数内容：

```
int main()
{//* Begin
    int i;
        // First, enable the clock of the PIO
        AT91F_PMC_EnablePeriphClock ( AT91C_BASE_PMC, 1 << AT91C_ID_PIOA ) ;

        // then, we configure the PIO Lines corresponding to LED1 to LED4
        // to be outputs. No need to set these pins to be driven by the PIO because it is GPIO pins only.
```

```
        AT91F_PIO_CfgOutput( AT91D_BASE_PIO_LED, AT91B_LED_MASK ) ;

        // Clear the LED's. On the EB55 we must apply a "1" to turn off LEDs
        AT91F_PIO_SetOutput( AT91D_BASE_PIO_LED, AT91B_LED_MASK ) ;
        // Clear the LED's. On the EB55 we must apply a "1" to turn off LEDs
        AT91F_PIO_ClearOutput( AT91D_BASE_PIO_LED, AT91B_LED_MASK ) ;

        //增加按键复位功能
        //set users reset function (reset about 512*(1/32k)=16ms ok!)
        AT91F_RSTSetMode(AT91C_BASE_RSTC,(8<<8)|(1<<0)); //in lib_AT91SAM7A3.h
        //#############################################################################
        //初始化串行口
         AT91F_DBGU_Init();
         AT91F_US0_Init();
         AT91F_US1_Init();
         AT91F_USRT_Printk((AT91PS_USART)AT91C_BASE_DBGU,"this is SPI dataflash test!\n\r");



        //#############################################################################
        //初始化 SPI_DataFlash
        AT91F_CfgSPIForDataFlash();
        //----------------------------擦除  DATAFLASH 的某一页（第 10 页）-----------------------
        AT91F_USRT_Printk((AT91PS_USART)AT91C_BASE_DBGU,"Erase one pages!\n\r");
        //等待 DATAFLASH 准备好

while(AT91F_DataFlashWaitReady(DataFlash.pDataFlashDesc,AT91C_DATAFLASH_TIMEOUT)!=DATAFLASH_OK);
        //擦除 DATAFLASH
        while(AT91F_DataFlashPageErase (&DataFlash,(unsigned
char*)Buffer_Dataflash,(10*DataFlash.pDevice->pages_size),(DataFlash.pDevice->pages_size) )!=DATAFLASH_OK);
        //等待擦除完

while(AT91F_DataFlashWaitReady(DataFlash.pDataFlashDesc,AT91C_DATAFLASH_TIMEOUT)!=DATAFLASH_OK);

        //---------------------------写  DATAFLASH 的某一页（第 10 页）---------------------------
        AT91F_USRT_Printk((AT91PS_USART)AT91C_BASE_DBGU,"Write one pages!\n\r");
        for(i=0;i<BUFFER_SIZE_DATAFLASH;i++) {Buffer_Write_Dataflash[i]=0xcc;
Buffer_Dataflash[i]=Buffer_Write_Dataflash[i]; }
        //等待 DATAFLASH 准备好

while(AT91F_DataFlashWaitReady(DataFlash.pDataFlashDesc,AT91C_DATAFLASH_TIMEOUT)!=DATAFLASH_OK);
        //写数据到 DATAFLASH
        while(AT91F_DataFlashPagePgmBuf(&DataFlash,(unsigned
char*)Buffer_Dataflash,(10*DataFlash.pDevice->pages_size),(DataFlash.pDevice->pages_size) )!=DATAFLASH_OK);
```

```
        //等待写完

while(AT91F_DataFlashWaitReady(DataFlash.pDataFlashDesc,AT91C_DATAFLASH_TIMEOUT)!=DATAFLASH_OK);

        //----------------------------读 DATAFLASH 的某一页（第 10 页）-----------------------------
        AT91F_USRT_Printk((AT91PS_USART)AT91C_BASE_DBGU,"Read one pages!\n\r");
        for(i=0;i<BUFFER_SIZE_DATAFLASH;i++)  Buffer_Dataflash[i] = 0x00;
        //等待 DATAFLASH 准备好

while(AT91F_DataFlashWaitReady(DataFlash.pDataFlashDesc,AT91C_DATAFLASH_TIMEOUT)!=DATAFLASH_OK);
        //从 DATAFLASH 读数据
        while(AT91F_DataFlashPageRead(&DataFlash,(10*DataFlash.pDevice->pages_size),(unsigned
char*)Buffer_Dataflash,(DataFlash.pDevice->pages_size) )!=DATAFLASH_OK);
        //等待读完

while(AT91F_DataFlashWaitReady(DataFlash.pDataFlashDesc,AT91C_DATAFLASH_TIMEOUT)!=DATAFLASH_OK);

        for(i=0;i<BUFFER_SIZE_DATAFLASH;i++){Buffer_Read_Dataflash[i]=Buffer_Dataflash[i];}

        //比较写入与读出的数据，如果不相等，则数据出错
        for(i=0;i<BUFFER_SIZE_DATAFLASH;i++)
        {
        if(Buffer_Read_Dataflash[i]!=0xcc) {AT91F_USRT_Printk((AT91PS_USART)AT91C_BASE_DBGU,"the data is
error--test is failure!\n\r"); break;}
        }

        if(i==BUFFER_SIZE_DATAFLASH) { AT91F_USRT_Printk((AT91PS_USART)AT91C_BASE_DBGU,"the test is
right!\n\r");}//写入如读出的数据相等

         AT91F_SPI_Close(AT91C_BASE_SPI0);
       // Loop forever
       for (;;)
         {
         // Once a Shot on each led
     for ( i=0 ; i < AT91B_NB_LEB ; i++ )
          {
               AT91F_PIO_ClearOutput( AT91D_BASE_PIO_LED, led_mask[i]) ;
          wait();
          AT91F_PIO_SetOutput( AT91D_BASE_PIO_LED, led_mask[i] ) ;
          wait();
          }// End for
          // Once a Shot on each led
          for ( i=(AT91B_NB_LEB-1) ; i >= 0 ; i-- )
          {
```

```
            AT91F_PIO_ClearOutput( AT91D_BASE_PIO_LED, led_mask[i]) ;
            wait();
            AT91F_PIO_SetOutput( AT91D_BASE_PIO_LED, led_mask[i] ) ;
            wait();
            }
      } // End for
}//* End
```

2.3.2 重要子函数代码注释解读

以下是对几个重要的子函数进行分析：

```
//*-----------------------------------------------------------------------------
//*  函数名: AT91F_SPI0_CfgSPI
//*  功能:   配置 SPI0
//*-----------------------------------------------------------------------------
static void    AT91F_SPI0_CfgSPI(void)
{
      //*  复位 SPI0
      AT91F_SPI_Reset(AT91C_BASE_SPI0);

      //*  配置  SPI0  工作在主模式--固定外设--NPCS0 片选  //如果选用不同的片选可在此处修改
      AT91F_SPI_CfgMode(AT91C_BASE_SPI0, AT91C_SPI_MSTR | AT91C_SPI_MODFDIS | (0xe<<16));

      //*  配置 8  位数据位  AT45DCB321C      //可在此处设置数据位 8-16 位
      AT91F_SPI_CfgCs(AT91C_BASE_SPI0,0, AT91C_SPI_CPOL | (AT91C_SPI_DLYBS & 0x100000)
|((AT91B_MCK/(DATAFLASH_CLK)) << 8));

       //*  启动  SPI0
          AT91F_SPI_Enable(AT91C_BASE_SPI0);
}
 //*-----------------------------------------------------------------------------
//*  函数名:    AT91F_CfgDataFlash
//*  功能:      初始化 DataFlash  数据结构
//*-----------------------------------------------------------------------------
static void AT91F_CfgDataFlash (void)
{
        //  初始化 DATAFLASH 特性,使之适合 AT45D321C      //如果选用其他型号的的 FLASH 则依据 FLASH 的
特性，在此修改
      DeviceAT45DB321C.pages_number = 8192;
      DeviceAT45DB321C.pages_size      = 528;
      DeviceAT45DB321C.page_offset    = 10;
      DeviceAT45DB321C.byte_mask       = 0x300;

      //  初始化  AT91S_DataflashDesc  结构
```

```
    DataflashDesc.state                 = IDLE;
    DataflashDesc.DataFlash_state       = IDLE;

    // 初始化 AT91S_DataFlash 全局结构, AT45DB321C 作为默认选择
    DataFlash.pDataFlashDesc = &DataflashDesc;
    DataFlash.pDevice    = &DeviceAT45DB321C;
}

//*----------------------------------------------------------------------------
//* 函数名:   AT91F_DataFlashHandler
//* 功能:     SPI0 中断处理程序
//*----------------------------------------------------------------------------
static void AT91F_DataFlashHandler(
    AT91PS_DataflashDesc pDesc,
    unsigned int status)
{
   //* 判断是否接收完，产生的中断
    if (( status & AT91C_SPI_RXBUFF))
    {
        if( pDesc->state == BUSY)
            {
            //*设置 DATAFLASH 的下一个状态为空闲状态,
        pDesc->state = IDLE;
        if (pDesc->DataFlash_state == GET_STATUS)
        pDesc->DataFlash_state = *( (unsigned char *) (pDesc->rx_cmd_pt) +1);

                //AT91F_USRT_Printk(AT91C_BASE_US0,"receive interrupt\n\r");

            //* 禁止传输中断
            AT91C_BASE_SPI0->SPI_IDR = AT91C_SPI_RXBUFF;
               //*允许 PDC 发送和接收 --PDC 传输控制寄存其
            AT91C_BASE_SPI0->SPI_PTCR = AT91C_PDC_TXTDIS + AT91C_PDC_RXTDIS;
            return;
            }
    }
        //否则状态错误
    pDesc->state = ERROR;
       // 禁止 PDC 接收和发送中断
    AT91C_BASE_SPI0->SPI_PTCR = AT91C_PDC_TXTDIS + AT91C_PDC_RXTDIS;
      //禁止出 RXBUFF 之外的所有中断
       AT91C_BASE_SPI0->SPI_IDR = status;
}

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

```
//*  函数名: AT91F_SPI_Handler
//*  功能:     SPI 中断入口
//*----------------------------------------------------------------------------
void AT91F_SPI_Handler(void)
{
    unsigned int status;
    //禁止  SPI0  中断
    AT91F_AIC_DisableIt(AT91C_BASE_AIC,AT91C_ID_SPI0);
  //----------------------------------------------------------------------------
    //取得  SPI_SR  和  SPI_IMR ----的状态
    status=( AT91F_SPI_GetStatus (AT91C_BASE_SPI0)&AT91F_SPI_GetInterruptMaskStatus(AT91C_BASE_SPI0));
    //调用中断处理函数
    AT91F_DataFlashHandler(DataFlash.pDataFlashDesc, status);
    //-----------------------------------------------------------------
    //  允许 SPI1 中断
    AT91F_AIC_EnableIt(AT91C_BASE_AIC,AT91C_ID_SPI0);
}

//*----------------------------------------------------------------------------
//*函数名:   AT91F_CfgSPIForDataFlash
//*  功能:     初始化 SPI0,与串口 DataFlash 通信
//*----------------------------------------------------------------------------
void AT91F_CfgSPIForDataFlash(void )
{
  // ============================  初始化  SPI  接口  ============================
    AT91F_USRT_Printk(AT91C_BASE_US0,"Init SPI0 Interface\n\r");
    //  初始化  SPI  作为  DataFlash  接口
    AT91F_SPI0_CfgPIO();
    //开启 SPI PMC
    AT91F_SPI0_CfgPMC();
    //允许接收缓冲区满中断---RXBUFF
    AT91F_SPI_EnableIt(AT91C_BASE_SPI0,0x1<<6);
    //配置 SPI  特性
    AT91F_SPI0_CfgSPI();
    //开启  PDC
    AT91F_PDC_Open(AT91C_BASE_PDC_SPI0);
  //配置  AT45DB321C
    AT91F_CfgDataFlash();
    //  配置  SPI  中断
    AT91F_AIC_ConfigureIt(AT91C_BASE_AIC,
                AT91C_ID_SPI0,
                        SpiDataflash_SYS_LEVEL,
                AT91C_AIC_SRCTYPE_INT_HIGH_LEVEL,
                AT91F_SPI_Handler);
```

```
    // 允许 SPI0 中断
    AT91F_AIC_EnableIt(AT91C_BASE_AIC,AT91C_ID_SPI0);
}

//*----------------------------------------------------------------------------
//* 函数名：AT91F_SpiWrite
//* 功能：  设置 PDC 传输功能
//  说明：  此处为重先编写的直接 PDC 传输功能，因为 SPI 接受和发送是同时的，所以直接对寄存器操作比较方便，
//          也可以调用库函数 AT91F_SPI_ReceiveFrame（）其功能与此处相同，
//          可参考 usart.c 文件，里面的传输采用调用库函数来实现的！
//          详细了解，参见 DATASHEET 的 PDC 部分。
//*----------------------------------------------------------------------------
static void AT91F_SpiWrite ( AT91PS_DataflashDesc pDesc )
{
       //设置设备的状态为忙，等待中断清除这种状态
    pDesc->state = BUSY;

    // 禁止 PDC 接收和发送中断
    AT91C_BASE_SPI0->SPI_PTCR = AT91C_PDC_TXTDIS + AT91C_PDC_RXTDIS;

    //* 初始化发送和接收指针
        AT91C_BASE_SPI0->SPI_RPR = (unsigned int)pDesc->rx_cmd_pt ;
        AT91C_BASE_SPI0->SPI_TPR = (unsigned int)pDesc->tx_cmd_pt ;

        //* 初始化发送和接收计数器
        AT91C_BASE_SPI0->SPI_RCR = pDesc->rx_cmd_size ;
        AT91C_BASE_SPI0->SPI_TCR = pDesc->tx_cmd_size ;

    if ( pDesc->tx_data_size != 0 )
    {
     //* 初始化下一个接受和发送指针
        AT91C_BASE_SPI0->SPI_RNPR = (unsigned int)pDesc->rx_data_pt ;
        AT91C_BASE_SPI0->SPI_TNPR = (unsigned int)pDesc->tx_data_pt ;

    //* 初始化下一个发送和接收计数器
     AT91C_BASE_SPI0->SPI_RNCR = pDesc->rx_data_size ;
     AT91C_BASE_SPI0->SPI_TNCR = pDesc->tx_data_size ;
    }
    //* 允许接收满中断---- RXBUFF
        AT91C_BASE_SPI0->SPI_IER = AT91C_SPI_RXBUFF;
        //* PDC 传输控制-----  允许 PDC 传输和接收
    AT91C_BASE_SPI0->SPI_PTCR = AT91C_PDC_TXTEN + AT91C_PDC_RXTEN;
```

```
}

//*----------------------------------------------------------------------------
//* 函数名：    AT91F_DataFlashGetStatus
//* 功能：      读 DATAFLASH 的状态
//*----------------------------------------------------------------------------
static AT91S_DataFlashStatus AT91F_DataFlashGetStatus(AT91PS_DataflashDesc pDesc)
{
    //* 如果数据传输没有结束 ==> 返回 0
    if( (pDesc->state) != IDLE)
    return DATAFLASH_BUSY;
    //* 首先发送和读取状态命令(D7H)
    pDesc->command[0] = DB_STATUS;
    pDesc->command[1] = 0;

    pDesc->DataFlash_state   = GET_STATUS;
    pDesc->tx_data_size     = 0 ; //* 传输命令，接收读取的状态----非数据传输
    pDesc->tx_cmd_pt            = pDesc->command ;
    pDesc->rx_cmd_pt            = pDesc->command ;
    pDesc->rx_cmd_size          = 2 ;
    pDesc->tx_cmd_size          = 2 ;
    AT91F_SpiWrite (pDesc);
    return DATAFLASH_OK;
}


//*----------------------------------------------------------------------------
//* 函数名：AT91F_DataFlashSendCommand
//* 功能：  通过 SPI 发送命令到 dataflash
//*----------------------------------------------------------------------------
static AT91S_DataFlashStatus AT91F_DataFlashSendCommand (
    AT91PS_DataFlash pDataFlash,
    unsigned char OpCode,
    unsigned int CmdSize,
    unsigned int DataflashAddress)
{
    unsigned int adr;

    if ( (pDataFlash->pDataFlashDesc->state) != IDLE)
    return DATAFLASH_BUSY;

    //* process the address to obtain page address and byte address
    adr = ((DataflashAddress / (pDataFlash->pDevice->pages_size)) << pDataFlash->pDevice->page_offset) +
(DataflashAddress % (pDataFlash->pDevice->pages_size));
```

```
    //* fill the   command   buffer */
    pDataFlash->pDataFlashDesc->command[0] = OpCode;
    pDataFlash->pDataFlashDesc->command[1] = (unsigned char)((adr & 0x00FF0000) >> 16);
    pDataFlash->pDataFlashDesc->command[2] = (unsigned char)((adr & 0x0000FF00) >> 8);
    pDataFlash->pDataFlashDesc->command[3] = (unsigned char)(adr & 0x000000FF) ;
    pDataFlash->pDataFlashDesc->command[4] = 0;
    pDataFlash->pDataFlashDesc->command[5] = 0;
    pDataFlash->pDataFlashDesc->command[6] = 0;
    pDataFlash->pDataFlashDesc->command[7] = 0;

    /* Initialize the SpiData structure for the spi write fuction */
    pDataFlash->pDataFlashDesc->tx_cmd_pt   =   pDataFlash->pDataFlashDesc->command ;
    pDataFlash->pDataFlashDesc->tx_cmd_size =   CmdSize ;
    pDataFlash->pDataFlashDesc->rx_cmd_pt   =   pDataFlash->pDataFlashDesc->command ;
    pDataFlash->pDataFlashDesc->rx_cmd_size =   CmdSize ;

       /* send the command and read the data */
    AT91F_SpiWrite (pDataFlash->pDataFlashDesc);

    return DATAFLASH_OK;
}

//*----------------------------------------------------------------------------
//*  函数名：  AT91F_DataFlashPageRead
//*  功能：    读取 DATAFLASH 的某一页
//*----------------------------------------------------------------------------
AT91S_DataFlashStatus AT91F_DataFlashPageRead (
    AT91PS_DataFlash pDataFlash,
    unsigned int src,
    unsigned char *dataBuffer,
    int sizeToRead )
{
    pDataFlash->pDataFlashDesc->rx_data_pt = dataBuffer ;    //*  读操作缓冲区的地址
    pDataFlash->pDataFlashDesc->rx_data_size = sizeToRead;   //*  读数据大小
    pDataFlash->pDataFlashDesc->tx_data_pt = dataBuffer ;
    pDataFlash->pDataFlashDesc->tx_data_size = sizeToRead;

   //*  发送读命令到 dataflash
   return (AT91F_DataFlashSendCommand (pDataFlash, DB_PAGE_READ, 8, src));
}

//*----------------------------------------------------------------------------
//*  函数名:AT91F_DataFlashPagePgmBuf
```

```
//* 功能：通过缓冲区 1 或缓冲区 2 向 DATAFLASH 写一页数据
//*----------------------------------------------------------------------------
AT91S_DataFlashStatus AT91F_DataFlashPagePgmBuf(
      AT91PS_DataFlash pDataFlash,
      unsigned char *src,
      unsigned int dest,
      unsigned int SizeToWrite)
{
      pDataFlash->pDataFlashDesc->tx_data_pt = src ;
      pDataFlash->pDataFlashDesc->tx_data_size = SizeToWrite ;
      pDataFlash->pDataFlashDesc->rx_data_pt = src;
      pDataFlash->pDataFlashDesc->rx_data_size = SizeToWrite;

    /* 发送写命令到 dataflash */
    return(AT91F_DataFlashSendCommand (pDataFlash,DB_PAGE_PGM_BUF1, 4, dest));
}

//*----------------------------------------------------------------------------
//* 函数名：    AT91F_DataFlashPageErase
//* 功能：      通过缓冲区 1 或缓冲区 2 擦除 DATAFLASH
//*----------------------------------------------------------------------------
AT91S_DataFlashStatus AT91F_DataFlashPageErase(
      AT91PS_DataFlash pDataFlash,
      unsigned char *src,
      unsigned int address,
      unsigned int SizeToWrite)
{
      pDataFlash->pDataFlashDesc->tx_data_pt = src ;
      pDataFlash->pDataFlashDesc->tx_data_size = SizeToWrite ;
      pDataFlash->pDataFlashDesc->rx_data_pt = src;
      pDataFlash->pDataFlashDesc->rx_data_size = SizeToWrite;

    /* 发送擦除命令到 dataflash */
    return(AT91F_DataFlashSendCommand (pDataFlash, DB_PAGE_ERASE, 4, address));
}

//*----------------------------------------------------------------------------
//* 函数名：AT91F_DataFlashWaitReady
//* 功能：等待 DATAFLAH 操作完 (bit7 of the status register == 1)
//*----------------------------------------------------------------------------
AT91S_DataFlashStatus AT91F_DataFlashWaitReady(AT91PS_DataflashDesc pDataFlashDesc, unsigned int timeout)
{
      unsigned int i;
```

```
    pDataFlashDesc->DataFlash_state = IDLE;

    do
    {
        AT91F_DataFlashGetStatus(pDataFlashDesc);
        timeout--;
        // 简单的等待
        for(i=0;i<10;i++);
    }
    while( ((pDataFlashDesc->DataFlash_state & 0x80) != 0x80) && (timeout>0) );

    if((pDataFlashDesc->DataFlash_state & 0x80) != 0x80)
        return DATAFLASH_ERROR;

    return DATAFLASH_OK;
}
/*
//############################附注：供外部调用的函数#####################################
extern   void AT91F_CfgSPIForDataFlash(void);   //dataflash 初始化
extern   AT91S_DataFlashStatus AT91F_DataFlashPageRead ( //读 FLASH 的一页
    AT91PS_DataFlash pDataFlash,
    unsigned int src,
    unsigned char *dataBuffer,
    int sizeToRead );
extern   AT91S_DataFlashStatus AT91F_DataFlashPagePgmBuf( //写一页
    AT91PS_DataFlash pDataFlash,
    unsigned char *src,
    unsigned int dest,
    unsigned int SizeToWrite);
extern   AT91S_DataFlashStatus AT91F_DataFlashPageErase( //擦除一页
    AT91PS_DataFlash pDataFlash,
    unsigned char *src,
    unsigned int address,
    unsigned int SizeToWrite);
extern   AT91S_DataFlashStatus AT91F_DataFlashWaitReady( //取得 DATAFLASH 的状态
        AT91PS_DataflashDesc pDataFlashDesc,
        unsigned int timeout);
//#############################################################################################
*/
```

2.3.3 运行结果

在终端上显示如下：