专题15-快车道DMA

一、DMA原理解析

1.1、DMA原理

DMA 传输将数据从一个地址空间复制到另外一个地址空间。当CPU 初始化这个传输动作，传输动作本身是由 DMA 控制器来实行和完成。典型的例子就是移动一个外部内存的区块到芯片内部更快的内存区。像是这样的操作并没有让处理器工作拖延，反而可以被重新排程去处理其他的工作。DMA 传输对于高效能 嵌入式系统算法和网络是很重要的。

在实现DMA传输时，是由DMA控制器直接掌管总线，因此，存在着一个总线控制权转移问题。即DMA传输前，CPU要把总线控制权交给DMA控制器，而在结束DMA传输后，DMA控制器应立即把总线控制权再交回给CPU。一个完整的DMA传输过程必须经过DMA请求、DMA响应、DMA传输、DMA结束4个步骤。

1.2、工作模式

•Demond模式：

如果DMA完成一次请求后如果Request仍然有效，那么 DMA就认为这是下一次DMA请求，并立即开始下一次的传输

•Handshake模式：

DMA完成一次请求后等待Request信号无效，如果 Request无效，DMA会无效ACK两个时钟周期，再等待下一次Request。

1.3、DMA控制器（OK6410）

11.1 OVERVIEW

S3C6410 contains four DMA controllers. Each DMA controller consists of 8 transfer channels. Each channel of DMA controller can perform data movements between devices in the AXI SYSTEM bus and/or AXI PERIPHERAL bus through AHBtoAXI bridges without any restrictions. In other words, each channel can handle the following four cases:

1) Both source and destination are in the SYSTEM bus.
2) Source is available in the SYSTEM bus while destination is available in the PERIPHERAL bus.
3) Source is available in the PERIPHERAL bus while destination is available in the SYSTEM bus.
4) Both source and destination are available in the PERIPHERAL bus.

ARM PrimeCell DMA controller PL080 is used as S3C6410 DMA controller. The DMAC is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM Limited. The DMAC is an AMBA AHB module, and connects to the *Advanced High-performance Bus* (AHB).

The main advantage of DMA is that it can transfer the data without CPU intervention. The operation of DMA can be initiated by S/W, or the request from internal peripherals, or the external request pins.

11.3 BLOCK DIAGRAM

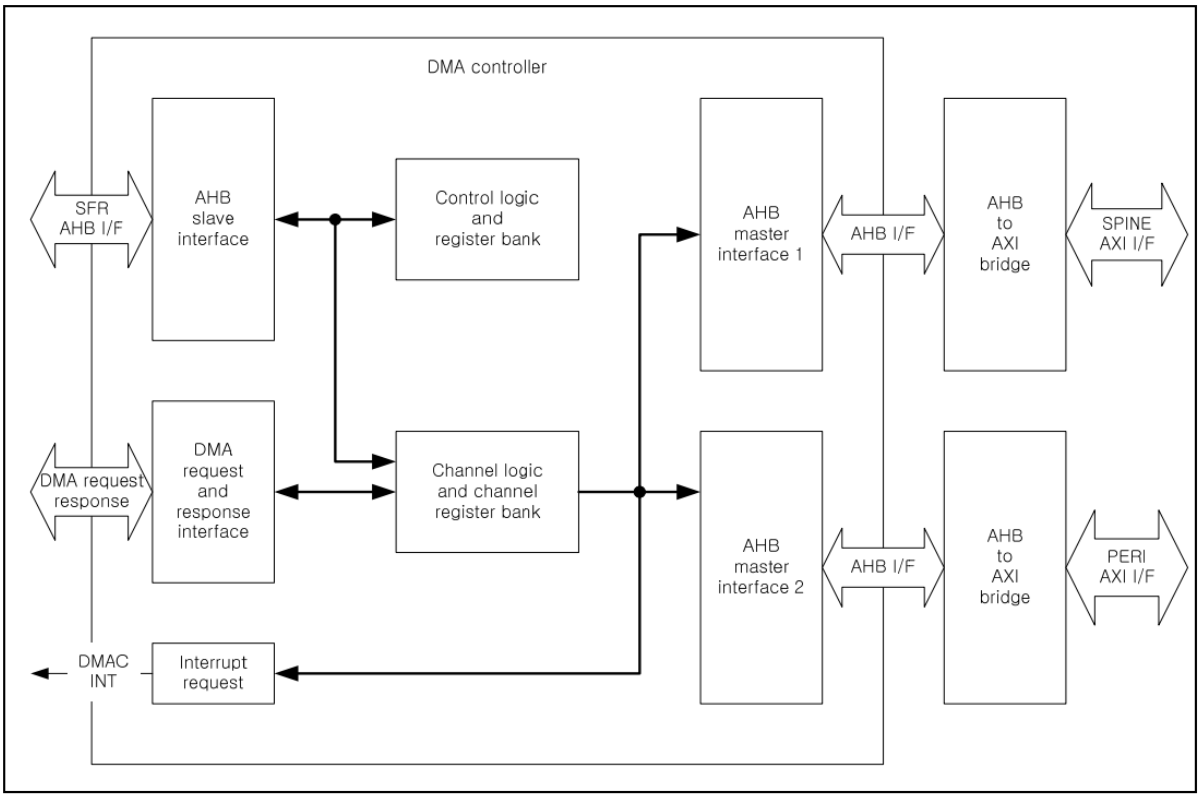

Figure 11-1. DMAC block diagram

11.4 DMA SOURCES

The S3C6410 supports 64 DMA sources as listed in the table below.
Reset value of SDMA_SEL register in System Controller is 0x0 which means SDMA selection. So, the configuration should be set to 1 in order to use general DMA. For more information, please refer to System Controller part.

Group	DMA No.	Sources	Description
DMA0, SDMA0	0	DMA_UART0[0]	UART0 DMA source 0
DMA0, SDMA0	1	DMA_UART0[1]	UART0 DMA source 1
DMA0, SDMA0	2	DMA_UART1[0]	UART1 DMA source 0
DMA0, SDMA0	3	DMA_UART1[1]	UART1 DMA source 1
DMA0, SDMA0	4	DMA_UART2[0]	UART2 DMA source 0
DMA0, SDMA0	5	DMA_UART2[1]	UART2 DMA source 1
DMA0, SDMA0	6	DMA_UART3[0]	UART3 DMA source 0
DMA0, SDMA0	7	DMA_UART3[1]	UART3 DMA source 1
DMA0, SDMA0	8	DMA_PCM0_TX	PCM0 DMA TX source
DMA0, SDMA0	9	DMA_PCM0_RX	PCM0 DMA RX source
DMA0, SDMA0	10	DMA_I2S0_TX	I2S0 TX DMA source
DMA0, SDMA0	11	DMA_I2S0_RX	I2S0 RX DMA source
DMA0, SDMA0	12	DMA_SPI0_TX	SPI0 TX DMA source
DMA0, SDMA0	13	DMA_SPI0_RX	SPI0 RX DMA source
DMA0, SDMA0	14	DMA_HSI_I2SV40_TX	I2S_V40 TX source or MIPI HSI DMA TX source
DMA0, SDMA0	15	DMA_HSI_I2SV40_RX	I2S_V40 RX source or MIPI HSI DMA RX source
DMA1, SDMA1	0	DMA_PCM1_TX	PCM1 DMA TX source
DMA1, SDMA1	1	DMA_PCM1_RX	PCM1 DMA RX source
DMA1, SDMA1	2	DMA_I2S1_TX	I2S1 TX DMA source
DMA1, SDMA1	3	DMA_I2S1_RX	I2S1 RX DMA source
DMA1, SDMA1	4	DMA_SPI1_TX	SPI1 TX DMA source
DMA1, SDMA1	5	DMA_SPI1_RX	SPI1 RX DMA source
DMA1, SDMA1	6	DMA_AC_PCMout	AC97 PCMout DMA source
DMA1, SDMA1	7	DMA_AC_PCMin	AC97 PCMin DMA source
DMA1, SDMA1	8	DMA_AC_MICin	AC97 MICin DMA source
DMA1, SDMA1	9	DMA_PWM	PWM DMA source
DMA1, SDMA1	10	DMA_IrDA	IrDA DMA source
DMA1, SDMA1	11	DMA_EXTERNAL	External DMA source
DMA1, SDMA1	12	Reserved	
DMA1, SDMA1	13	Reserved	
SDMA1	14	DMA_SECU_RX	Security RX DMA source
SDMA1	15	DMA_SECU_TX	Security TX DMA source

11.6 FUNCTIONAL TIMING DIAGRAM

A peripheral asserts a DMA request and holds it active. The **DMACCLR** signal is asserted by the DMA controller when the last data item has been transferred. When the peripheral notice that the **DMACCLR** signal has gone active it makes the DMA request signal inactive. The DMAC controller deasserts the **DMACLR** signal when the DMA request signal goes inactive.

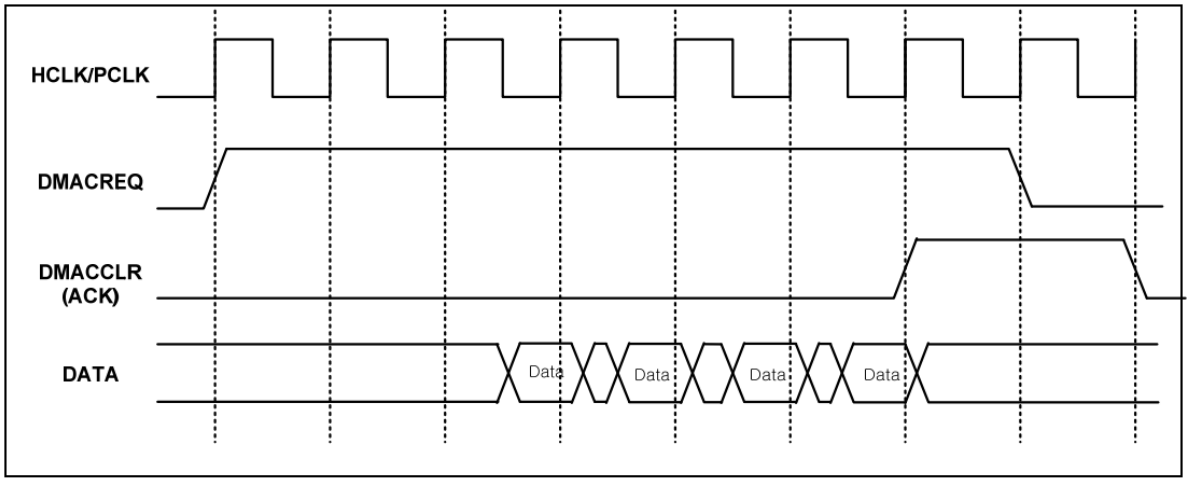

Figure 11-6. DMA interface timing

二、DMA程序设计（OK6410）

2.1、DMA操作流程

11.7.8 PROGRAMMING A DMA CHANNEL

Steps to program a DMA channel:

1. Decide whether use secure DMAC(SDMAC) or general DMAC(DMAC). In order to use general DMAC, disable secure DMA control register(SDMA_SEL) of system controller. (Reset value is SDMAC)
2. Select a free DMA channel with the priority needed. Where DMA channel 0 has the highest priority and DMA channel 7 the lowest priority.
3. Clear any pending interrupts on the channel to be used by writing to the DMACIntTCClr and DMACIntErrClr registers. The previous channel operation might have left interrupts active.
4. Write the source address into the DMACCxSrcAddr register.
5. Write the destination address into the DMACCxDestAddr register.
6. Write the address of the next LLI into the DMACCxLLI register. If the transfer comprises of a single packet of data then must be written into this register.

Offset	Contents
Next LLI address	Source Address for next transfer
Next LLI address + 0x04	Destination Address for next transfer
Next LLI address + 0x08	Next LLI address for next transfer
Next LLI address + 0x0C	DMACCxControl0 data for next transfer
Next LLI address + 0x10	DMACCxControl1 data for next transfer

7. Write the control information into the DMACCxControl register.
8. Write the channel configuration information into the DMACCxConfiguration register. If the Enable bit is set then the DMA channel is automatically enabled.

2.2、寄存器操作

2.2.1、选择SDMAC还是DMAC

3.4.2.7 Secure DMA control register

REGISTER	ADDRESS	R/W	DESCRIPTION	RESET VALUE
SDMA_SEL	0x7E00_F110	R/W	Secure DMA input selection	0x0000_0000

SDMA_SEL	BIT	DESCRIPTION	RESET VALUE
SECURITY_TX	[31]	DMA selection for security Tx (always selects SDMA1 regardless of SECURITY_TX field)	0
SECURITY_RX	[30]	DMA selection for security Rx (always selects SDMA1 regardless of SECURITY_TX field)	0
RESERVED	[29]	RESERVED	0
RESERVED	[28]	RESERVED	0
EXTERNAL	[27]	DMA selection for external (0: SDMA1, 1: DMA1)	0
IRDA	[26]	DMA selection for IrDA (0: SDMA1, 1: DMA1)	0
PWM	[25]	DMA selection for IrDA (0: SDMA1, 1: DMA1)	0
AC_MICIN	[24]	DMA selection for IrDA (0: SDMA1, 1: DMA1)	0
AC_PCMIN	[23]	DMA selection for PCM input (0: SDMA1, 1: DMA1)	0
AC_PCMOUT	[22]	DMA selection for PCM output (0: SDMA1, 1: DMA1)	0
SPI1_RX	[21]	DMA selection for SPI1 Rx (0: SDMA1, 1: DMA1)	0
SPI1_TX	[20]	DMA selection for SPI1 Tx (0: SDMA1, 1: DMA1)	0
I2S1_RX	[19]	DMA selection for I2S1 Rx (0: SDMA1, 1: DMA1)	0
I2S1_TX	[18]	DMA selection for I2S1 Tx (0: SDMA1, 1: DMA1)	0
PCM1_RX	[17]	DMA selection for PCM1 Rx (0: SDMA1, 1: DMA1)	0
PCM1_TX	[16]	DMA selection for PCM1 Tx (0: SDMA1, 1: DMA1)	0
HSI_RX	[15]	DMA selection for HSI Rx (0: SDMA0, 1: DMA0)	0
HSI_TX	[14]	DMA selection for HSI Tx (0: SDMA0, 1: DMA0)	0
SPI0_RX	[13]	DMA selection for SPI0 Rx (0: SDMA0, 1: DMA0)	0
SPI0_TX	[12]	DMA selection for SPI0 Tx (0: SDMA0, 1: DMA0)	0
I2S0_RX	[11]	DMA selection for I2S0 Rx (0: SDMA0, 1: DMA0)	0
I2S0_TX	[10]	DMA selection for I2S0 Tx (0: SDMA0, 1: DMA0)	0
PCM0_RX	[9]	DMA selection for PCM0 Rx (0: SDMA0, 1: DMA0)	0
PCM0_TX	[8]	DMA selection for PCM0 Tx (0: SDMA0, 1: DMA0)	0
UART3[1]	[7]	DMA selection for UART3 (0: SDMA0, 1: DMA0)	0
UART3[0]	[6]	DMA selection for UART3 (0: SDMA0, 1: DMA0)	0
UART2[1]	[5]	DMA selection for UART2 (0: SDMA0, 1: DMA0)	0
UART2[0]	[4]	DMA selection for UART2 (0: SDMA0, 1: DMA0)	0
UART1[1]	[3]	DMA selection for UART1 (0: SDMA0, 1: DMA0)	0
UART1[0]	[2]	DMA selection for UART1 (0: SDMA0, 1: DMA0)	0
UART0[1]	[1]	DMA selection for UART0 (0: SDMA0, 1: DMA0)	0
UART0[0]	[0]	DMA selection for UART0 (0: SDMA0, 1: DMA0)	0

```
1 #define SDMAC0_BASE            (*((volatile unsigned long *)0x7db00000))
2
3 #define SDMA_SEL               (*((volatile unsigned long *)0x7e00f110))
4 #define DMACIntTCClear         (*((volatile unsigned long *)0x7db00008))
5 #define DMACIntErrClr          (*((volatile unsigned long *)0x7db00010))
6 #define DMACConfiguration      (*((volatile unsigned long *)0x7db00030))
7 #define DMACSync               (*((volatile unsigned long *)0x7db00034))
8 #define DMACC0SrcAddr          (*((volatile unsigned long *)0x7db00100))
9 #define DMACC0DestAddr         (*((volatile unsigned long *)0x7db00104))
10 #define DMACC0Control0         (*((volatile unsigned long *)0x7db0010c))
11 #define DMACC0Control1         (*((volatile unsigned long *)0x7db00110))
12 #define DMACC0Configuration    (*((volatile unsigned long *)0x7db00114))
13
14
15 #define UTXH0          ((volatile unsigned long *)0x7f005020)
16
17 char src[100] = "\n\rDMA transfer test!\n\r";
18
```

```
19 void dma_init()
20 {
21        //Decide whether use secure DMAC(SDMAC) or general DMAC(DMAC).
22        SDMA_SEL = 0x0;
```

2.2.2、选择DMA channel（没有找到对应寄存器）

11.8 REGISTER DESCRIPTION

There are four DMA Controller named as DMAC0, DMAC1, SDMAC0, and SDMAC1.
The register base addresses of DMAC0, DMAC1, SDMAC0, and SDMAC1 are 0x7500_0000, 0x7510_0000, 0x7DB0_0000, and 0x7DC0_0000 respectively.
Page- access feature for OneNAND Controller is added to channel 3 of DMAC0 and SDAMC0.

11.8.1 DMA REGISTER LOCATION

Table 11-1. DMA register summary

Name	Type	Width	Description	Offset	Reset Value
DMACIntStatus	R	8	This register provides the interrupt status of the DMA controller. A HIGH bit indicates that a specific DMA channel interrupt is active.	0x000	0x00
DMACIntTCStatus	R	8	This register is used to determine whether an interrupt was generated due to the transaction completing (terminal count). A HIGH bit indicates that the transaction is completed.	0x004	0x00
DMACIntTCClear	W	8	When writing to this register, each data bit that is HIGH causes the corresponding bit in the DMACIntTCStatus and DMACRawIntTCStatus registers to be cleared. Data bits that are LOW have no effect on the corresponding bit in the register.	0x008	-
DMACIntErrorStatus	R	8	This register is used to determine whether an interrupt was generated due to an error being generated.	0x00C	0x00
DMACIntErrClr	W	8	When writing to this register, each data bit that is HIGH causes the corresponding bit in the DMACIntErrorStatus and DMACRawIntErrorStatus registers to be cleared. Data bits that are LOW have no effect on the corresponding bit in the register.	0x010	-
DMACRawIntTCStatus	R	8	This register provides the raw status of DMA terminal count interrupts prior to masking. A HIGH bit indicates that the interrupt request is active prior to masking.	0x014	-
DMACRawIntErrorStatus	R	8	This register provides the raw status of DMA error interrupts prior to masking. A HIGH bit indicates that the interrupt request is active prior to masking.	0x018	-
DMACEnbldChns	R	8	This register shows which DMA channels are enabled. A HIGH bit indicates that a DMA channel is enabled.	0x01C	0x00

Table 11-1 DMA register summary (continued)

Name	Type	Width	Description	Offset	Reset Value
DMACSoftBReq	R/W	16	This register allows DMA burst requests to be generated by software.	0x020	0x0000
DMACSoftSReq	R/W	16	This register allows DMA single requests to be generated by software.	0x024	0x0000
Reserved	-	16	-	0x028	0x0000
Reserved	-	16	-	0x02C	0x0000
DMACConfiguration	R/W	3	This register is used to configure the DMA controller.	0x030	0x000
DMACSync	R/W	16	This register enables or disables synchronization logic for the DMA request signals.	0x034	0x0000
DMACC0SrcAddr	R/W	32	DMA channel 0 source address.	0x100	0x00000000
DMACC0DestAddr	R/W	32	DMA channel 0 destination address.	0x104	0x00000000
DMACC0LLI	R/W	32	DMA channel 0 linked list address.	0x108	0x00000000
DMACC0Control0	R/W	32	DMA channel 0 control0.	0x10C	0x00000000
DMACC0Control1	R/W	32	DMA channel 0 control1.	0x110	0x00000000
DMACC0Configuration	R/W	19	DMA channel 0 configuration register.	0x114	0x00000
DMACC0ConfigurationExp	R/W	3	DMA channel 0 configuration expension reg.	0x118	0x0
DMACC1SrcAddr	R/W	32	DMA channel 1 source address.	0x120	0x00000000
DMACC1DestAddr	R/W	32	DMA channel 1 destination address.	0x124	0x00000000
DMACC1LLI	R/W	32	DMA channel 1 linked list address.	0x128	0x00000000
DMACC1Control0	R/W	32	DMA channel 1 control0.	0x12C	0x00000000
DMACC1Control1	R/W	32	DMA channel 1 control1.	0x130	0x00000000
DMACC1Configuration	R/W	19	DMA channel 1 configuration register.	0x134	0x00000
DMACC1ConfigurationExp	R/W	3	DMA channel 1 configuration expension reg.	0x138	0x0
DMACC2SrcAddr	R/W	32	DMA channel 2 source address.	0x140	0x00000000
DMACC2DestAddr	R/W	32	DMA channel 2 destination address.	0x144	0x00000000
DMACC2LLI	R/W	32	DMA channel 2 linked list address.	0x148	0x00000000
DMACC2Control0	R/W	32	DMA channel 2 control.	0x14C	0x00000000
DMACC2Control1	R/W	32	DMA channel 2 control.	0x150	0x00000000
DMACC2Configuration	R/W	19	DMA channel 2 configuration register.	0x154	0x00000
DMACC2ConfigurationExp	R/W	3	DMA channel 2 configuration expension reg.	0x158	0x0
DMACC3SrcAddr	R/W	32	DMA channel 3 source address.	0x160	0x00000000
DMACC3DestAddr	R/W	32	DMA channel 3 destination address.	0x164	0x00000000
DMACC3LLI	R/W	32	DMA channel 3 linked list address.	0x168	0x00000000

Table 11-1 DMA register summary (continued)

Name	Type	Width	Description	Offset	Reset Value
DMACC3Control0	R/W	32	DMA channel 3 control0.	0x16C	0x00000000
DMACC3Control1	R/W	32	DMA channel 3 control1.	0x170	0x00000000
DMACC3Configuration	R/W	19	DMA channel 3 configuration register.	0x174	0x00000
DMACC3ConfigurationExp	R/W	3	DMA channel 3 configuration expension reg.	0x178	0x0
DMACC4SrcAddr	R/W	32	DMA channel 4 source address.	0x180	0x00000000
DMACC4DestAddr	R/W	32	DMA channel 4 destination address.	0x184	0x00000000
DMACC4LLI	R/W	32	DMA channel 4 linked list address.	0x188	0x00000000
DMACC4Control0	R/W	32	DMA channel 4 control0.	0x18C	0x00000000
DMACC4Control1	R/W	32	DMA channel 4 control1.	0x190	0x00000000
DMACC4Configuration	R/W	19	DMA channel 4 configuration register.	0x194	0x00000
DMACC4ConfigurationExp	R/W	3	DMA channel 4 configuration expension reg.	0x198	0x0
DMACC5SrcAddr	R/W	32	DMA channel 5 source address.	0x1A0	0x00000000
DMACC5DestAddr	R/W	32	DMA channel 5 destination address.	0x1A4	0x00000000
DMACC5LLI	R/W	32	DMA channel 5 linked list address.	0x1A8	0x00000000
DMACC5Control0	R/W	32	DMA channel 5 control0.	0x1AC	0x00000000
DMACC5Control1	R/W	32	DMA channel 5 control1.	0x1B0	0x00000000
DMACC5Configuration	R/W	19	DMA channel 5 configuration register.	0x1B4	0x00000
DMACC5ConfigurationExp	R/W	3	DMA channel 5 configuration expension reg.	0x1B8	0x0
DMACC6SrcAddr	R/W	32	DMA channel 6 source address.	0x1C0	0x00000000
DMACC6DestAddr	R/W	32	DMA channel 6 destination address.	0x1C4	0x00000000
DMACC6LLI	R/W	32	DMA channel 6 linked list address.	0x1C8	0x00000000
DMACC6Control0	R/W	32	DMA channel 6 control0.	0x1CC	0x00000000
DMACC6Control1	R/W	32	DMA channel 6 control1.	0x1D0	0x00000000
DMACC6Configuration	R/W	19	DMA channel 6 configuration register.	0x1D4	0x00000
DMACC6ConfigurationExp	R/W	3	DMA channel 6 configuration expension reg.	0x1D8	0x0
DMACC7SrcAddr	R/W	32	DMA channel 7 source address.	0x1E0	0x00000000
DMACC7DestAddr	R/W	32	DMA channel 7 destination address.	0x1E4	0x00000000
DMACC7LLI	R/W	32	DMA channel 7 linked list address.	0x1E8	0x00000000
DMACC7Control0	R/W	32	DMA channel 7 control0.	0x1EC	0x00000000
DMACC7Control1	R/W	32	DMA channel 7 control1.	0x1F0	0x00000000
DMACC7Configuration	R/W	19	DMA channel 7 configuration register.	0x1F4	0x00000
DMACC7ConfigurationExp	R/W	3	DMA channel 7 configuration expension reg.	0x1F8	0x0

2.2.3、使能DMAC控制器

11.8.14 CONFIGURATION REGISTER, DMACCONFIGURATION

The DMACConfiguration read/write register is used to configure the operation of the DMA controller. The AHB master interfaces are set to little-endian mode on reset.

Table 11-12 shows the bit assignment of the DMACConfiguration register.

Table 11-12. Bit Assignment of DMACConfiguration register

DMACConfiguration	Bits	Type	Function
Reserved	[2:1]	R/W	Should be 0
E	[0]	R/W	DMA controller enable: 0 =disabled 1 =enabled. This bit is reset to 0.Disabling the DMA controller reduces power consumption.

```
24      //enable DMA controller
25      DMACConfiguration = 0x1;
```

2.2.4、清除中断

11.8.4 INTERRUPT TERMINAL COUNT CLEAR REGISTER, DMACINTTCCLEAR

The DMACIntTCClear register is write-only and is used to clear a terminal count interrupt request.

When writing to this register, each data bit that is set HIGH causes the corresponding bit in the status register to be cleared. Data bits that are LOW have no effect on the corresponding bit in the register.

Table 11-4 shows the bit assignment of the DMACIntTCClear register.

Table 11-4. Bit Assignment of DMACIntTCClear register

DMACIntTCClear	Bits	Type	Function
IntTCClear	[7:0]	W	Terminal count request clear

11.8.6 INTERRUPT ERROR CLEAR REGISTER, DMACINTERRCLR

The DMACIntErrClr register is a write-only register and is used to clear the error interrupt requests. When writing to this register, each data bit that is HIGH causes the corresponding bit in the status register to be cleared. Data bits that are LOW have no effect on the corresponding bit in the register.

Table 11-6 shows the bit assignment of the DMACIntErrClr register.

Table 11-6. Bit Assignment of DMACIntErrClr register

DMACIntErrClr	Bits	Type	Function
IntErrClr	[7:0]	W	Interrupt error clear

```
30      //Clear any pending interrupts on the channel to be used by writing to the DMACIntTCClr and DMACIntErrClr
31      //registers.
32      DMACIntTCClear = 0x0f;
33      DMACIntErrClr = 0x0f;
```

11.8.15 SYNCHRONIZATION REGISTER, DMACSYNC

The DMACSync read/write register is used to enable or disable synchronization logic for the DMA request signals.
A bit set to 0 enables the synchronization logic for a particular group of DMA requests.
A bit set to 1 disables the synchronization logic for a particular group of DMA requests.
This register is reset to 0, synchronization logic enabled.

Table 11-13 shows the bit assignment of the DMACSync register.

Table 11-13. Bit Assignment of DMACSync register

DMACSync	Bits	Type	Function
DMACSync	[15:0]	R/W	DMA synchronization logic for DMA request signals enabled or disabled. A LOW bit indicates that the synchronization logic for the **DMACBREQ[15:0]**, **DMACSREQ[15:0]**, **DMACLBREQ[15:0]**, and **DMACLSREQ[15:0]** request signals is enabled. A HIGH bit indicates that the synchronization logic is disabled.

NOTE: Synchronization logic must be used when the peripheral generating the DMA request runs on a different clock to the DMA controller. For peripherals running on the same clock as the DMA controller disabling the synchronization logic improves the DMA request response time. If necessary, the DMA response signals, **DMACCLR** and **DMACTC**, must be synchronized in the peripheral.

2.2.5、设置源地址

11.8.16 CHANNEL SOURCE ADDRESS REGISTER, DMACCXSRCADDR

The eight read/write DMACCxSrcAddr registers contain the current source address (byte-aligned) of the data to be transferred.

Each register is programmed directly by software before the appropriate channel is enabled. When the DMA channel is enabled this register is updated:

- As the source address is incremented
- By following the linked list when a complete packet of data has been transferred.

Reading the register when the channel is active does not provide useful information. This is because by the time that software has processed the value read, the channel might have progressed. It is intended to be read only when the channel has stopped, in which case it shows the source address of the last item read.

NOTE:

The source and destination addresses must be aligned to the source and destination widths.

Table 11-14 shows the bit assignment of the DMACCxSrcAddr registers.

Table 11-14. Bit Assignment of DMACCxSrcAddr register

DMACCxSrcAddr	Bits	Type	Function
SrcAddr	[31:0]	R/W	DMA Source address

```
35      //Write the source address into the DMACCxSrcAddr register.
36      DMACC0SrcAddr = (unsigned int)src;
```

2.2.6、设置目的地址

11.8.17 CHANNEL DESTINATION ADDRESS REGISTER, DMACCXDESTADDR

The eight read/write DMACCxDestAddr registers contain the current destination address (byte-aligned) of the data to be transferred.

Each register is programmed directly by software before the channel is enabled. When the DMA channel is enabled, the register is updated as the destination address is incremented and by following the linked list when a complete packet of data has been transferred.

Reading the register when the channel is active does not provide useful information. This is because by the time that software has processed the value read, the channel might have progressed. It is intended to be read only when a channel has stopped, in which case it shows the destination address of the last item read.

Table 11-15 shows the bit assignment of a DMACCxDestAddr register.

Table 11-15. Bit Assignment of DMACCxDestAddr register

DMACCxDestAddr	Bits	Type	Function
DestAddr	[31:0]	R/W	DMA destination address

```
38      //Write the destination address into the DMACCxDestAddr register.
39      DMACC0DestAddr = (unsigned int)UTXH0;
```

2.2.7、设置LLI寄存器（不使用）

11.8.18 CHANNEL LINKED LIST ITEM REGISTER, DMACCXLLI

The eight read/write DMACCxLLI registers contain a word aligned address of the next *Linked List Item* (LLI). If the LLI is, then the current LLI is the last in the chain, and the DMA channel is disabled once all DMA transfers associated with it are completed.

NOTE:

Programming this register when the DMA channel is enabled has unpredictable side effects.
To make loading LLIs more efficient for some systems, the LLI data structures can be made 4-word aligned.

Table 11-16 shows the bit assignment of a DMACCxLLI register.

Table 11-16. Bit Assignment of DMACCxLLI register

DMACCxLLI	Bits	Type	Function
LLI	[31:2]	R/W	Linked list item. Bits [31:2] of the address for the next LLI. Address bits [1:0] are.
R	[1]	R/W	Reserved, and must be written as 0,masked on read.
LM	[0]	R/W	AHB master select for loading the next LLI: LM = 0 = AHB master 1 LM = 1 = AHB master 2.

2.2.8、设置 DMACC0Control0（各参数设置）

11.8.19 CHANNEL CONTROL REGISTER, DMACCXCONTROL0

The eight read/write DMACCxControl0 registers contain DMA channel control information such as the burst size, and transfer width.
Each register is programmed directly by software before the DMA channel is enabled. When the channel is enabled the register is updated by following the linked list when a complete packet of data has been transferred. Reading the register whilst the channel is active does not give useful information. This is because by the time that software has processed the value read, the channel might have progressed. It is intended to be read only when a channel has stopped.

Table 11-17 shows the bit assignment of a DMACCxControl0 register.

Table 11-17. Bit Assignment of DMACCxControl0 register

DMACCxControl	Bits	Type	Function
I	[31]	R/W	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
Prot	[30:28]	R/W	Protection. Refer to Table 11-20
DI	[27]	R/W	Destination increment. When set the destination address is incremented after each transfer.
SI	[26]	R/W	Source increment. When set the source address is incremented after each transfer.
D	[25]	R/W	Destination AHB master select: 0 = AHB master 1 (AXI_SYSTEM) selected for the destination transfer. 1 = AHB master 2 (AXI_PERI) selected for the destination transfer.

Table 11-17. Bit Assignment of DMACCxControl register (continued)

DMACCxControl	Bits	Type	Function
S	[24]	R/W	Source AHB master select:

	[24]	R/W	Source AHB master select: 0 = AHB master 1 (AXI_SYSTEM) selected for the source transfer 1 = AHB master 2 (AXI_PERI) selected for the source transfer.
Dwidth	[23:21]	R/W	Destination transfer width. Transfers wider than the AHB master bus width are illegal. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data as required.
SWidth	[20:18]	R/W	Source transfer width. Transfers wider than the AHB master bus width are illegal. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data as required.
DBSize	[17:15]	R/W	Destination burst size. Indicates the number of transfers, which make up a destination burst transfer request. This value must be set to the burst size of the destination peripheral, or if the destination is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the **DMACxBREQ** signal goes active in the destination peripheral. The burst size is not related to the AHB **HBURST** signal.
SBSize	[14:12]	R/W	Source burst size. Indicates the number of transfers, which make up a source burst. This value must be set to the burst size of the source peripheral, or if the source is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the **DMACxBREQ** signal goes active in the source peripheral. The burst size is not related to the AHB **HBURST** signal.
Reserved	[11:0]	R	Reserved

Table 11-18. Source or destination burst size

Bit value of SBSize or DBSize	Source or destination burst transfer request size
0b000	1
0b001	4
0b010	8
0b011	16
0b100	Reserved
0b101	Reserved
0b110	Reserved
0b111	Reserved

Table 11-19. Source or destination transfer width

Bit value of SWidth or DWidth	Source or destination width
0b000	Byte (8-bit)
0b001	Half word (16-bit)
0b010	Word (32-bit)
0b011	Reserved
0b100	Reserved
0b101	Reserved
0b110	Reserved
0b111	Reserved

Note. DMAC has internal 4 word FIFO per each channel. So, burst size and transfer width is limited by the FIFO size. For example, if data width is word, available burst size is under 4. If data width is byte, available burst size is under 16.

AHB access information is provided to the source and destination peripherals when a transfer occurs. The transfer information is provided by programming the DMA channel (the Prot bit of the DMACCxControl register, and the Lock bit of the DMACCxConfiguration register). These bits are programmed by software and peripherals can use this information if necessary. Three bits of information are provided, and Table 11-20 shows the purpose of the three protection bits.

Table 11-20. Protection bits

Bits	Description	Purpose
0	Privileged or	Indicates that the access is in User or privileged mode:

0	Privileged or User	Indicates that the access is in User, or privileged mode: 0 = User mode 1 = privileged mode. This bit controls the AHB **HPROT[1]** signal.
1	Bufferable or not bufferable	Indicates that the access is bufferable, or not bufferable: 0 = not bufferable 1 = bufferable. This bit indicates that the access is bufferable. This bit can, for example, be used to indicate to an AMBA bridge that the read can complete in zero wait states on the source bus without waiting for it to arbitrate for the destination bus and for the slave to accept the data. This bit controls the AHB **HPROT[2]** signal.
2	Cacheable or not cacheable	Indicates that the access is cacheable or not cacheable: 0 = not cacheable 1 = cacheable. This indicates that the access is cacheable. This bit can, for example, be used to indicate to an AMBA bridge that when it saw the first read of a burst of eight it can transfer the whole burst of eight reads on the destination bus, rather than pass the transactions through one at a time. This bit controls the AHB **HPROT[3]** signal.

```
44      //Write the control information into the DMACCxControl register.
45
46      //Destination AHB master select AHB 2,Source increment,Destination fixed,Source AHB master select AHB master
1,transfer width 8bit.1 size burst transfer
47      DMACC0Control0 = ((0 << 31) | (0 << 27) | (1 << 26) | (1 << 25) | (0 << 24));
48
```

2.2.9、设置传输数量

11.8.20 CHANNEL CONTROL REGISTER, DMACCXCONTROL1

The eight read/write DMACCxControl1 registers contain DMA channel control information such as the transfer size.

Each register is programmed directly by software before the DMA channel is enabled. When the channel is enabled the register is updated by following the linked list when a complete packet of data has been transferred.

Reading the register whilst the channel is active does not give useful information. This is because by the time that software has processed the value read, the channel might have progressed. It is intended to be read only when a channel has stopped.
Table 11-21 shows the bit assignment of a DMACCxControl1 register.

Table 11-21. Bit Assignment of DMACCxControl1 register

DMACCxControl	Bits	Type	Function
TransferSize	[24:0]	R/W	Transfer size. A write to this field indicates the size of transfer. A read from this field indicates the number of transfers completed on the destination bus. Reading the register when the channel is active does not give useful information, as by the time that the software has processed the value read, the channel might have progressed. It is intended to be used only when a channel is enabled and then disabled.

```
50      //set Transfer size
51      DMACC0Control1 = 0x64;
52
```

2.2.10、配置通道（不清楚具体设置参数，可不设置）

11.8.21 CHANNEL CONFIGURATION REGISTER, DMACCXCONFIGURATION

The eight DMACCxConfiguration registers are read/write and are used to configure the DMA channel. The registers are not updated when a new LLI is requested.

Table 11-22 shows the bit assignment of a DMACCxConfiguration register.

Table 11-22. Bit Assignment of DMACCxConfiguration register

DMACCxConfiguration	Bits	Type	Function
H	[18]	R/W	Halt: 0 = allow DMA requests 1 = ignore further source DMA requests. The contents of the channels FIFO are drained. This value can be

			used with the Active and Channel Enable bits to cleanly disable a DMA channel.
A	[17]	R	Active: 0 = there is no data in the FIFO of the channel 1 = the FIFO of the channel has data. This value can be used with the Halt and Channel Enable bits to cleanly disable a DMA channel.
L	[16]	R/W	Lock. When set this bit enables locked transfers.
ITC	[15]	R/W	Terminal count interrupt mask. When cleared this bit masks out the terminal count interrupt of the relevant channel.
IE	[14]	R/W	Interrupt error mask. When cleared this bit masks out the error interrupt of the relevant channel.
FlowCntrl	[13:11]	R/W	Flow control and transfer type. This value is used to indicate the flow controller and transfer type. The supported flow controller is only the DMA controller. The transfer type can be memory-to-memory, memory-to-peripheral, peripheral-to-memory, or peripheral-to-peripheral.
OneNandModeDst	[10]	R/W	This bit is used to support page-write features for OneNAND Controller. If this bit is set to 1 and the destination address points the address field of OneNAND Controller, destination address increment setting can support 01 command of OneNAND Controller. To be sure, when this bit is set to one, D should be “AHB master 1”, DI should be “increment”, DWidth should be “word”, and DBSize should be the multiple of four. Note that using DMAC0 does not guarantee correct operation.
DestPeripheral	[9:6]	R/W	Destination peripheral. This value selects the DMA destination request peripheral. This field is ignored if the destination of the transfer is to memory.

DMACCxConfiguration	Bits	Type	Function
OneNandModeSrc	[5]	R/W	This bit is used to support page-read features for OneNAND Controller. If this bit is set to one and the source address points the address field of OneNAND Controller, source address increment setting can support 01 command of OneNAND Controller. To be sure, when this bit is set to one, S should be “AHB master 1”, SI should be “increment”, SWidth should be “word”, and SBSize should be the multiple of four. Note that using DMAC0 does not guarantee correct operation.
SrcPeripheral	[4:1]	R/W	Source peripheral. This value selects the DMA source request peripheral. This field is ignored if the source of the transfer is from memory.
E	[0]	R/W	Channel enable. Reading this bit indicates whether a channel is currently enabled or disabled: 0 = channel disabled

			1 = channel enabled. The Channel Enable bit status can also be found by reading the DMACEnbldChns register. A channel is enabled by setting this bit. A channel can be disabled by clearing the Enable bit. This causes the current AHB transfer (if one is in progress) to complete and the channel is then disabled. Any data in the channels FIFO is lost. Restarting the channel by simply setting the Channel Enable bit has unpredictable effects and the channel must be fully re-initialized. The channel is also disabled, and Channel Enable bit cleared, when the last LLI is reached or if a channel error is encountered. If a channel has to be disabled without losing data in a channels FIFO the Halt bit must be set so that further DMA requests are ignored. The Active bit must then be polled until it reaches 0, indicating that there is no data left in the channels FIFO. Finally the Channel Enable bit can be cleared.

Table 11-23 describes the bit values of the three flow control and transfer type bits.

Table 11-23. Flow control and transfer type bits

Bits value	Transfer type
000	Memory to memory
001	Memory to peripheral
010	Peripheral to memory
011	Source peripheral to destination peripheral
100~111	Reserved

```
53      //allow DMA requests, interrupt mask,memory-to-peripheral,
54      DMACC0Configuration = ((1 << 6) | (0 << 11) | (1 << 14) | (1 << 15));
55 }
56
```

2.2.11、启动传输

```
void dma_start(void)
58 {
59      DMACC0Configuration |= 0x1;
60 }
61
```

注意：UART使用DMA需要打开FIFO功能

31.6.3 UART FIFO CONTROL REGISTER

Register	Address	R/W	Description	Reset Value
UFCON0	0x7F005008	R/W	UART channel 0 FIFO control register	0x0
UFCON1	0x7F005408	R/W	UART channel 1 FIFO control register	0x0
UFCON2	0x7F005808	R/W	UART channel 2 FIFO control register	0x0
UFCON3	0x7F005C08	R/W	UART channel 3 FIFO control register	0x0

There are three UART FIFO control registers including UFCON0, UFCON1, UFCON2 and UFCON3 in the UART block.

UFCONn	Bit	Description	Initial State
Tx FIFO Trigger Level	[7:6]	Determine the trigger level of transmit FIFO. 00 = Empty 01 = 16-byte 10 = 32-byte 11 = 48-byte	00
Rx FIFO Trigger Level	[5:4]	Determine the trigger level of receive FIFO.1) 00 = 1-byte 01 = 8-byte 10 = 16-byte 11 = 32-byte	00
Reserved	[3]		0
Tx FIFO Reset	[2]	Auto-cleared after resetting FIFO 0 = Normal 1= Tx FIFO reset	0
Rx FIFO Reset	[1]	Auto-cleared after resetting FIFO 0 = Normal 1= Rx FIFO reset	0
FIFO Enable	[0]	0 = Disable 1 = Enable	0

NOTES:

1) For using RX DMA in FIFO mode, Rx FIFO trigger level must be same value as DMA burst size. When using DMA single operation, RX FIFO trigger level must be set to 1-byte.

31.6.4 UART MODEM CONTROL REGISTER

Register	Address	R/W	Description	Reset Value
UMCON0	0x7F00500C	R/W	UART channel 0 Modem control register	0x0
UMCON1	0x7F00540C	R/W	UART channel 1 Modem control register	0x0
Reserved	0x7F00580C	-	Reserved	Undef
Reserved	0x7F005C0C	-	Reserved	Undef

There are two UART MODEM control registers including UMCON0 and UMCON1 in the UART block.

UMCONn	Bit	Description	Initial State
RTS trigger Level	[7:5]	When AFC bit is enabled, these bits determine when to inactivate nRTS signal. 000 = When RX FIFO contains 63 bytes. 001 = When RX FIFO contains 56 bytes. 010 = When RX FIFO contains 48 bytes. 011 = When RX FIFO contains 40 bytes. 100 = When RX FIFO contains 32 bytes. 101 = When RX FIFO contains 24 bytes. 110 = When RX FIFO contains 16 bytes. 111 = When RX FIFO contains 8 bytes.	000
Auto Flow Control (AFC)	[4]	0 = Disable 1 = Enable	0
Modem Interrupt enable	[3]	Modem interrupt enable 0 = Disable 1 = Enable	000
Reserved	[2:1]	These bits must be 0's	000
Request to Send	[0]	If AFC bit is enabled, this value will be ignored. In this case the S3C6410 will control nRTS automatically. If AFC bit is disabled, nRTS must be controlled by software. 0 = 'H' level (Inactivate nRTS) 1 = 'L' level (Activate nRTS)	0

NOTE: UART 2 does not support AFC function, because the S3C6410 has no nRTS2 and nCTS2.
UART 3 does not support AFC function, because the S3C6410 has no nRTS3 and nCTS3.

31.6.7 UART FIFO STATUS REGISTER

Register	Address	R/W	Description	Reset Value
UFSTAT0	0x7F005018	R	UART channel 0 FIFO status register	0x00
UFSTAT1	0x7F005418	R	UART channel 1 FIFO status register	0x00
UFSTAT2	0x7F005818	R	UART channel 2 FIFO status register	0x00
UFSTAT3	0x7F005C18	R	UART channel 3 FIFO status register	0x00

There are three UART FIFO status registers including UFSTAT0, UFSTAT1, UFSTAT2 and UFSTAT3 in the UART block.

UFSTATn	Bit	Description	Initial State
Reserved	[15]		0
Tx FIFO Full	[14]	Set to 1 automatically whenever transmit FIFO is full during transmit operation 0 = 0-byte ≤ Tx FIFO data ≤ 63-byte 1 = Full	0
Tx FIFO Count	[13:8]	Number of data in Tx FIFO	0
Reserved	[7]		0
Rx FIFO Full	[6]	Set to 1 automatically whenever receive FIFO is full during receive operation 0 = 0-byte ≤ Rx FIFO data ≤ 63-byte 1 = Full	0
Rx FIFO Count	[5:0]	Number of data in Rx FIFO	0

```
4 #define UFCON0      (*((volatile unsigned long *)0x7f005008))
5 #define UMCON0      (*((volatile unsigned long *)0x7f00500c))
6
```

```
7 #define UTRSTAT0        (*((volatile unsigned long *)0x7f005010))
8 #define UFSTAT0         (*((volatile unsigned long *)0x7f005018))

30      UFCON0 = 0x01;
31
32      UMCON0 = 0x0;

42 void uart_put_char(unsigned char c)
43 {
44      while (UFSTAT0 & (1 << 14)) ;   //if TX FIFO FULL,wait
45      UTXH0 = c;
46 }
47
48 unsigned char uart_get_char(void)
49 {
50      unsigned char c;
51
52      while ((UFSTAT0 & 0x7f) == 0) ;
53      c = URXH0;
54
55      if ((c == 0x0d) || (c == 0x0a)) {
56              uart_put_char(0x0d);
57              uart_put_char(0x0a);
58      }
59      else
60              uart_put_char(c);
61
62      return c;
63 }
64
```