OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide

Literature Number: SPRU755C
March 2005

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Preface

Read This First

About This Manual

This document describes the direct memory access (DMA) support of the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

❑ Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

Documentation that describes the OMAP5912 devices, related peripherals, and other technical collateral, is available in the OMAP5912 Product Folder on TI's website: www.ti.com/omap5912.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

This page is intentionally left blank.

Contents

Figures

Tables

Examples

This page is intentionally left blank.

Direct Memory Access (DMA) Support

This document describes the direct memory access (DMA) support of the OMAP5912 multimedia processor.

1 DMA Overview

The processor has two DMAs:

- ❑ The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- ❑ The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals. See the *OMAP5910/5912 Multimedia Processor DSP Subsystem Reference Guide* (SPRU890) for information on the DSP DMA.

The system DMA can support up to 31 hardware DMA requests. The DSP DMA supports 19 hardware DMA requests.

In OMAP5912, up to 56 different hardware DMA requests can be generated by MPU and shared peripherals. An embedded crossbar, called MPU GDMA handler, allows mapping any of these 56 requests to any of 31 the system DMA request.

Similarly, DSP and shared peripherals can generate up to 28 hardware DMA requests. An embedded crossbar, called DSP GDMA handler allows mapping any of these 28 requests to any of the 19 DSP DMA requests.

Both MPU and DSP GDMA handlers are configured through registers implemented in the configuration module.

2 GDMA Handlers

MPU and DSP peripherals DMA request mapping is configurable through two blocks called GDMA handlers. MPU GDMA handler is described in section 2.1. DSP GDMA handler is described in section 2.2.

2.1 MPU GDMA Handler

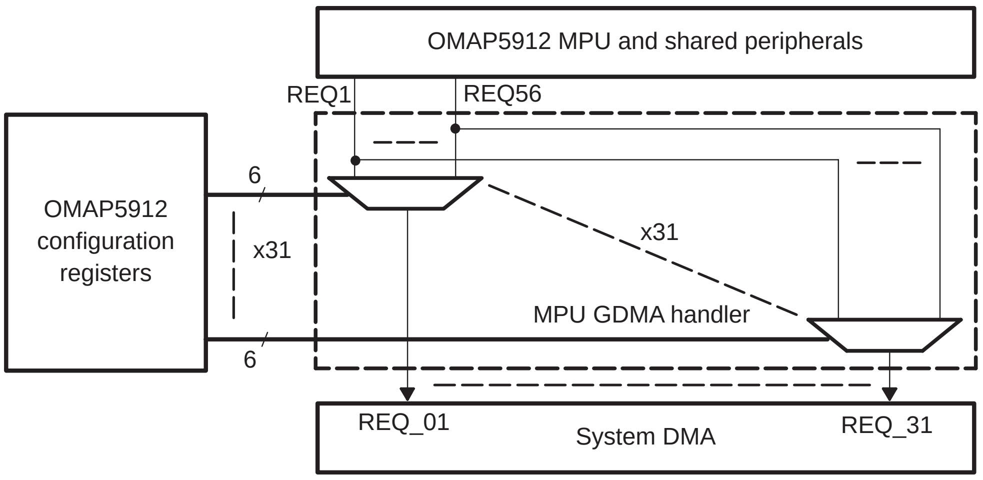
The MPU and shared peripherals control up to 56 DMA requests, whereas the system DMA can handle only 31 DMA requests.

The GDMA handler acts as a crossbar so that each of the incoming DMA requests can be remapped to any of the system DMA requests.

The mapping of each DMA request is done through the FUNC_MPU_DMA_x registers located in the configuration module.

A 6-bit field is associated with each system channel so that any incoming DMA request is remapped to the proper system DMA request. See Figure 1 for a description of the MPU GDMA handler.

Figure 1. MPU GDMA Handler


OMAP5912 MPU and shared peripherals
REQ1
REQ56
OMAP5912 configuration registers
6
x31
x31
MPU GDMA handler
6
REQ_01
System DMA
REQ_31

The default configuration after reset ensures compatibility with the previous OMAP5912 generation. Programmers have flexibility to remap up to 31 requests according to the application task requirements. Table 1 describes the mapping between MPU and shared peripherals DMA requests and the MPU GDMA handler inputs.

Table 1. MPU GDMA Handler Mapping

MPU GDMA Handler Input Line	Peripheral Request
REQ1	MCSI1 TX
REQ2	MCSI1 RX
REQ3	I^{2}C RX
REQ4	I^{2}C TX
REQ5	External DMA request 0
REQ6	External DMA request 1
REQ7	µWire TX
REQ8	McBSP1 DMA TX
REQ9	McBSP1 DMA RX
REQ10	McBSP3 DMA TX

Table 1. MPU GDMA Handler Mapping (Continued)

MPU GDMA Handler Input Line	Peripheral Request
REQ11	McBSP3 DMA RX
REQ12	UART1 TX
REQ13	UART1 RX
REQ14	UART2 TX
REQ15	UART2 RX
REQ16	McBSP2 TX
REQ17	McBSP2 RX
REQ18	UART3 TX
REQ19	UART3 RX
REQ20	Camera IF RX
REQ21	MMC/SDIO1 TX
REQ22	MMC/SDIO1 RX
REQ24	IRQ_LCD_LINE
REQ26	USB W2FC RX0
REQ27	USB W2FC RX1
REQ28	USB W2FC RX2
REQ29	USB W2FC TX0
REQ30	USB W2FC TX1
REQ31	USB W2FC TX2
REQ33	SPI TX
REQ34	SPI RX
REQ38	CMT-APE TX (channel 0)
REQ39	CMT-APE RV (channel 0)
REQ40	CMT-APE TX (channel 1)
REQ41	CMT-APE RV (channel 1)
REQ42	CMT-APE TX (channel 2)

Table 1. MPU GDMA Handler Mapping (Continued)

MPU GDMA Handler Input Line	Peripheral Request
REQ43	CMT-APE RV (channel 2)
REQ44	CMT-APE TX (channel 3)
REQ45	CMT-APE RV (channel 3)
REQ46	CMT-APE TX (channel 4)
REQ47	CMT-APE RV (channel 4)
REQ48	CMT-APE TX (channel 5)
REQ49	CMT-APE RV (channel 5)
REQ50	CMT-APE TX (channel 6)
REQ51	CMT-APE RV (channel 6)
REQ52	CMT-APE TX (channel 7)
REQ53	CMT-APE RV (channel 7)
REQ54	MMC/SDIO2 TX
REQ55	MMC/SDIO2 RX
REQ58	Unconnected
REQ59	Unconnected
REQ60	Unconnected
REQ61	Unconnected
REQ62	Unconnected
REQ63	Unconnected
REQ64	Unconnected

2.1.1 MPU GDMA Handler Configuration

The mapping of the system DMA requests is done through the GDMA registers (shown in Table 2), which are implemented in the configuration module. Table 3 through Table 9 describe the register bits.

The values programmed into these registers represent a zero-based numbering of the DMA request. Thus, peripheral request number *n* is programmed as *n-1*, not *n*.

Table 2. MPU GDMA Handler Control Registers

Base Address = 0xFFFE 1000			
Name	**Description**	**R/W**	**Offset**
FUNC_MUX_MPU_DMA_A	Controls mapping for system DMA requests 1 to 5.	R/W	0xEC
FUNC_MUX_MPU_DMA_B	Controls mapping for system DMA requests 6 to 10.	R/W	0xF0
FUNC_MUX_MPU_DMA_C	Controls mapping for system DMA requests 11 to 15.	R/W	0xF4
FUNC_MUX_MPU_DMA_D	Controls mapping for system DMA requests 16 to 20.	R/W	0xF8
FUNC_MUX_ARM_DMA_E	Controls mapping for system DMA requests 21 to 25.	R/W	0xFC
FUNC_MUX_ARM_DMA_F	Controls mapping for system DMA requests 26 to 30.	R/W	0x100
FUNC_MUX_ARM_DMA_G	Controls mapping for system DMA request 31.	R/W	0x104

Table 3. Functional Multiplexing MPU DMA A Register (FUNC_MUX_MPU_DMA_A)

Base Address = 0xFFFE 1000, Offset Address = 0xEC				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_05	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(5). *n* is between 0 and 55.	R/W	0x04
23:18	CONF_ARM_DMA_REQ_04	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(4). *n* is between 0 and 55.	R/W	0x03
17:12	CONF_ARM_DMA_REQ_03	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(3). *n* is between 0 and 55.	R/W	0x02
11:6	CONF_ARM_DMA_REQ_02	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(2). *n* is between 0 and 55.	R/W	0x01
5:0	CONF_ARM_DMA_REQ_01	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(1). *n* is between 0 and 55.	R/W	0x00

This register controls the system DMA crossbar and defines the mapping of MPU peripheral DMA requests 1 through 5 to system DMA requests. 56 MPU peripheral DMA requests can be mapped to the 31 system DMA controller requests. The values programmed in the register represent a zero-based numbering of peripheral DMA_REQn (starting with 1). Peripheral DMA_REQ1 is written as zero. For example, if bits 5:0 are equal to 3, then n+1 = 4, and DMA MPU peripheral request source 4 (I^2C TX) is connected to system DMA_REQ(1).

Table 4. Functional Multiplexing MPU DMA B Register (FUNC_MUX_MPU_DMA_B)

Base Address = 0xFFFE 1000, Offset Address = 0xF0

Bit	Name	Function	R/W	Reset
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_10	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(10). *n* is between 0 and 55.	R/W	0x09
23:18	CONF_ARM_DMA_REQ_09	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(9). *n* is between 0 and 55.	R/W	0x08
17:12	CONF_ARM_DMA_REQ_08	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(8). *n* is between 0 and 55.	R/W	0x07
11:6	CONF_ARM_DMA_REQ_07	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(7). *n* is between 0 and 55.	R/W	0x06
5:0	CONF_ARM_DMA_REQ_06	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(6). *n* is between 0 and 55.	R/W	0x05

Table 5. Functional Multiplexing MPU DMA C Register (FUNC_MUX_MPU_DMA_C)

Base Address = 0xFFFE 1000, Offset Address = 0xF4

Bit	Name	Function	R/W	Reset
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_15	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(15). *n* is between 0 and 55.	R/W	0x0E

Table 5. Functional Multiplexing MPU DMA C Register (FUNC_MUX_MPU_DMA_C) (Continued)

Bit	Name	Function	R/W	Reset
23:18	CONF_ARM_DMA_REQ_14	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(14). *n* is between 0 and 55.	R/W	0x0D
17:12	CONF_ARM_DMA_REQ_13	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(13). *n* is between 0 and 55.	R/W	0x0C
11:6	CONF_ARM_DMA_REQ_12	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(12). *n* is between 0 and 55.	R/W	0x0B
5:0	CONF_ARM_DMA_REQ_11	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(11). *n* is between 0 and 55.	R/W	0x0A

Table 6. Functional Multiplexing MPU DMA D Register(FUNC_MUX_MPU_DMA_D)

Base Address = 0xFFFE 1000, Offset Address = 0xF8

Bit	Name	Function	R/W	Reset
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_20	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(20). *n* is between 0 and 55.	R/W	0x13
23:18	CONF_ARM_DMA_REQ_19	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(19). *n* is between 0 and 55.	R/W	0x12
17:12	CONF_ARM_DMA_REQ_18	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(18). *n* is between 0 and 55.	R/W	0x11
11:6	CONF_ARM_DMA_REQ_17	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(17). *n* is between 0 and 55.	R/W	0x10
5:0	CONF_ARM_DMA_REQ_16	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(16). *n* is between 0 and 55.	R/W	0x0F

Table 7.   Functional Multiplexing MPU DMA E Register (FUNC_MUX_MPU_DMA_E)

Base Address = 0xFFFE 1000, Offset Address = 0xFC				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_25	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(25). *n* is between 0 and 55.	R/W	0x18
23:18	CONF_ARM_DMA_REQ_24	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(24). *n* is between 0 and 55.	R/W	0x17
17:12	CONF_ARM_DMA_REQ_23	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(23). *n* is between 0 and 55.	R/W	0x16
11:6	CONF_ARM_DMA_REQ_22	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(22). *n* is between 0 and 55.	R/W	0x15
5:0	CONF_ARM_DMA_REQ_21	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(21). *n* is between 0 and 55.	R/W	0x14

Table 8.   Functional Multiplexing MPU DMA F Register (FUNC_MUX_ARM_DMA_F)

Base Address = 0xFFFE 1000, Offset Address = 0x100				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved.	R/W	0x0
29:24	CONF_ARM_DMA_REQ_30	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(30). *n* is between 0 and 55.	R/W	0x1D
23:18	CONF_ARM_DMA_REQ_29	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(29). *n* is between 0 and 55.	R/W	0x1C
17:12	CONF_ARM_DMA_REQ_28	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(28). *n* is between 0 and 55.	R/W	0x1B

Table 8. Functional Multiplexing MPU DMA F Register (FUNC_MUX_ARM_DMA_F) (Continued)

Bit	Name	Function	R/W	Reset
11:6	CONF_ARM_DMA_REQ_27	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(27). *n* is between 0 and 55.	R/W	0x1A
5:0	CONF_ARM_DMA_REQ_26	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(26). *n* is between 0 and 55.	R/W	0x19

Table 9. Functional Multiplexing MPU DMA G Register (FUNC_MUX_ARM_DMA_G)

Base Address = 0xFFFE 1000, Offset Address = 0x104

Bit	Name	Function	R/W	Reset
31:6	RESERVED	Reserved.	R/W	0x0000000
5:0	CONF_ARM_DMA_REQ_31	Writing value *n* in this register maps DMA request source *n+1* to system DMA controller DMA_REQ(31). *n* is between 0 and 55.	R/W	0x1E

2.2 DSP GDMA Handler

The various DSP and shared peripherals control up to 28 DMA requests, whereas the DSP DMA in the OMAP3.2 can handle only 19 DMA requests.

The DSP GDMA handler acts as a crossbar so that each of the incoming DMA requests can be remapped to any of the DSP DMA requests.

The mapping of each DMA request is done through the FUNC_DSP_DMA_x registers located in the configuration module.

A 5-bit field is associated with each DSP channel so that any incoming DMA request is remapped to the proper DSP DMA request.

The default configuration from reset ensures compatibility with the previous OMAP5912 generation. Programmers have the flexibility to remap up to 19 requests according to the application task requirements. See Figure 2 for a description of the DSP GDMA handler and Table 10 for DSP GDMA mapping.

Figure 2. DSP GDMA Handler

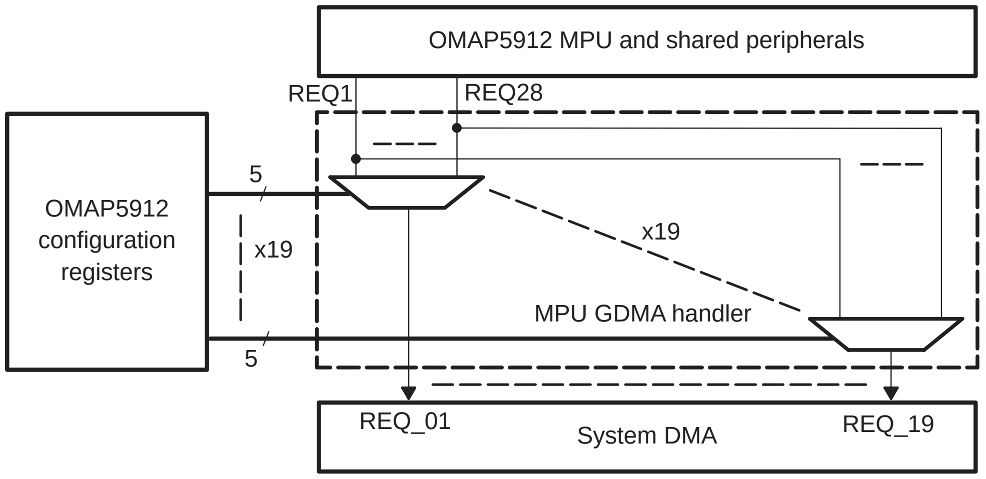

OMAP5912 MPU and shared peripherals
REQ1
REQ28
OMAP5912 configuration registers
5
x19
5
x19
MPU GDMA handler
REQ_01
System DMA
REQ_19

Table 10. DSP GDMA Mapping

DSP GDMA Handler Input Line	**Peripheral Request**
REQ1	MCSI1 TX
REQ2	MCSI1 RX
REQ3	MCSI2 TX
REQ4	MCSI2 RX
REQ5	MMC/SDIO2 TX
REQ6	MMC/SDIO2 RX
REQ7	Reserved
REQ8	McBSP1 TX
REQ9	McBSP1 RX
REQ10	McBSP3 TX
REQ11	McBSP3 RX
REQ12	UART1 TX
REQ13	UART1 RX
REQ14	UART2 TX
REQ15	UART2 RX
REQ16	I^{2}C RX
REQ17	I^{2}C TX

Table 10. DSP GDMA Mapping (Continued)

DSP GDMA Handler Input Line	Peripheral Request
REQ18	UART3 TX
REQ19	UART3 RX
REQ20	CMT-APE TX (channel 1)
REQ21	CMT-APE RV (channel 1)
REQ22	CMT-APE TX (channel 2)
REQ23	CMT-APE RV (channel 2)
REQ25	SPI TX
REQ26	SPI RX
REQ27	McBSP2 TX
REQ28	McBSP2 RX
REQ29	Unconnected
REQ30	Unconnected
REQ31	Unconnected
REQ32	Unconnected

2.3 DSP GDMA Handler configuration

The mapping of the DSP DMA requests is done through the FUNC_MUX_DSP_DMA_X registers (shown in Table 11), which are implemented in the configuration module. Table 12 through Table 15 describe the register bits.

The values programmed into these registers represent a zero-based numbering of the DMA request. Thus, peripheral request number *n* is programmed as *n-1*, not *n*.

Table 11. DSP DMA Handler Control Registers

Base Address = 0xFFFE 1000			
Name	**Description**	**R/W**	**Offset**
FUNC_MUX_DSP_DMA_A	Controls mapping for DSP DMA requests 1 to 6	R/W	0xD0
FUNC_MUX_DSP_DMA_B	Controls mapping for DSP DMA requests 7 to 12	R/W	0xD4
FUNC_MUX_DSP_DMA_C	Controls mapping for DSP DMA requests 13 to 18	R/W	0xD8
FUNC_MUX_DSP_DMA_D	Controls mapping for DSP DMA request 19	R/W	0xDC

Table 12. Functional Multiplexing DSP DMA A Register (FUNC_MUX_DSP_DMA_A)

Base Address = 0xFFFE 1000, Offset Address = 0xD0				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved	R/W	0x0
29:25	CONF_DSP_DMA_REQ_06	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(6). *n* is between 0 and 27.	R/W	0x05
24:20	CONF_DSP_DMA_REQ_05	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(5). *n* is between 0 and 27.	R/W	0x04
19:15	CONF_DSP_DMA_REQ_04	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(4). *n* is between 0 and 27.	R/W	0x03
14:10	CONF_DSP_DMA_REQ_03	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(3). *n* is between 0 and 27.	R/W	0x02
9:5	CONF_DSP_DMA_REQ_02	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(2). *n* is between 0 and 27.	R/W	0x01
4:0	CONF_DSP_DMA_REQ_01	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(1). *n* is between 0 and 27.	R/W	0x00

This register controls the DSP DMA crossbar and defines the mapping of DSP peripheral DMA requests 1 through 6 to OMAP3.2 DSP DMA requests. 28 DSP peripheral DMA requests can be mapped to the 19 DSP DMA controller requests. The values programmed in the register represent a zero-based numbering of DMA_REQn (starting with 1). Thus, peripheral DMA_REQ1 is written as zero. For example, if bits 4:0 are equal to 3, then n+1=4, and DSP DMA_REQ(1) maps to DMA DSP peripheral request source 4 (MCSI2 RX).

Table 13. Functional Multiplexing DSP DMA B Register (FUNC_MUX_DSP_DMA_B)

Base Address = 0xFFFE 1000, Offset Address = 0xD4				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved	R/W	0x0
29:25	CONF_DSP_DMA_REQ_12	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(12). *n* is between 0 and 27.	R/W	0x0B
24:20	CONF_DSP_DMA_REQ_11	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(11). *n* is between 0 and 27.	R/W	0x0A
19:15	CONF_DSP_DMA_REQ_10	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(10). *n* is between 0 and 27.	R/W	0x09
14:10	CONF_DSP_DMA_REQ_09	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(9). *n* is between 0 and 27.	R/W	0x08
9:5	CONF_DSP_DMA_REQ_08	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(8). *n* is between 0 and 27.	R/W	0x07
4:0	CONF_DSP_DMA_REQ_07	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(7). *n* is between 0 and 27.	R/W	0x06

Table 14. Functional Multiplexing DSP DMA C Register (FUNC_MUX_DSP_DMA_C

Base Address = 0xFFFE 1000, Offset Address = 0xD8				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:30	RESERVED	Reserved	R/W	0x0
29:25	CONF_DSP_DMA_REQ_18	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(18). *n* is between 0 and 27.	R/W	0x11
24:20	CONF_DSP_DMA_REQ_17	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(17). *n* is between 0 and 27.	R/W	0x10
19:15	CONF_DSP_DMA_REQ_16	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(16). *n* is between 0 and 27.	R/W	0x0F
14:10	CONF_DSP_DMA_REQ_15	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(15). *n* is between 0 and 27.	R/W	0x0E
9:5	CONF_DSP_DMA_REQ_14	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(14). *n* is between 0 and 27.	R/W	0x0D
4:0	CONF_DSP_DMA_REQ_13	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(13). *n* is between 0 and 27.	R/W	0x0C

Table 15. Functional Multiplexing DSP DMA D Register (FUNC_MUX_DSP_DMA_D)

Base Address = 0xFFFE 1000, Offset Address = 0xDC				
Bit	**Name**	**Function**	**R/W**	**Reset**
31:5	RESERVED	Reserved for future expansion	R/W	0x0000000
4:0	CONF_DSP_DMA_REQ_19	Writing value *n* in this register maps DMA request source *n+1* to DSP DMA controller DMA_REQ(19). *n* is between 0 and 27.	R/W	0x12

3 System DMA

The system DMA is designed to off-load the block data transfer function from the MPU.

The OMAP 3.2 system DMA controller consists of:

- ❑ Sixteen logical channels plus one LCD logical channel
- ❑ Seven physical ports plus one for configuration
- ❑ Three physical channels plus one LCD dedicated physical channel

The ports are connected to the OCP-T1 and OCP-T2 targets, the external memory, the TIPB bridge, the MPUI, and one dedicated port connected to an LCD controller. The system DMA controller can be controlled via the MPU private TIPB or by an external host via the OCP-I port.

The system DMA controller is designed for low-power operation. It is partitioned into several clock domains where each clock domain is enabled only when it is used. All clocks are disabled when no DMA transfers are active.

Five different logical channels types are supported; each one represents a specific feature set.

- ❑ LCh-2D for memory to memory transfers, 1D and 2D
- ❑ LCh-P for peripheral transfers
- ❑ LCh-PD for peripheral transfers on a dedicated channel
- ❑ LCh-G for graphical transfers/operations
- ❑ LCh-D for display transfers

The features available are:

- ❑ Support for up to four address modes:
  - ■ Constant
  - ■ Post-incremented mode
  - ■ Single-indexed
  - ■ Double-indexed
- ❑ Different indexing for source-respective destination
- ❑ Logical channel chaining
- ❑ Software enabling
- ❑ Hardware enabling− 31 DMA request lines available
- ❑ Logical channel interleaving
- ❑ Logical channel preemption

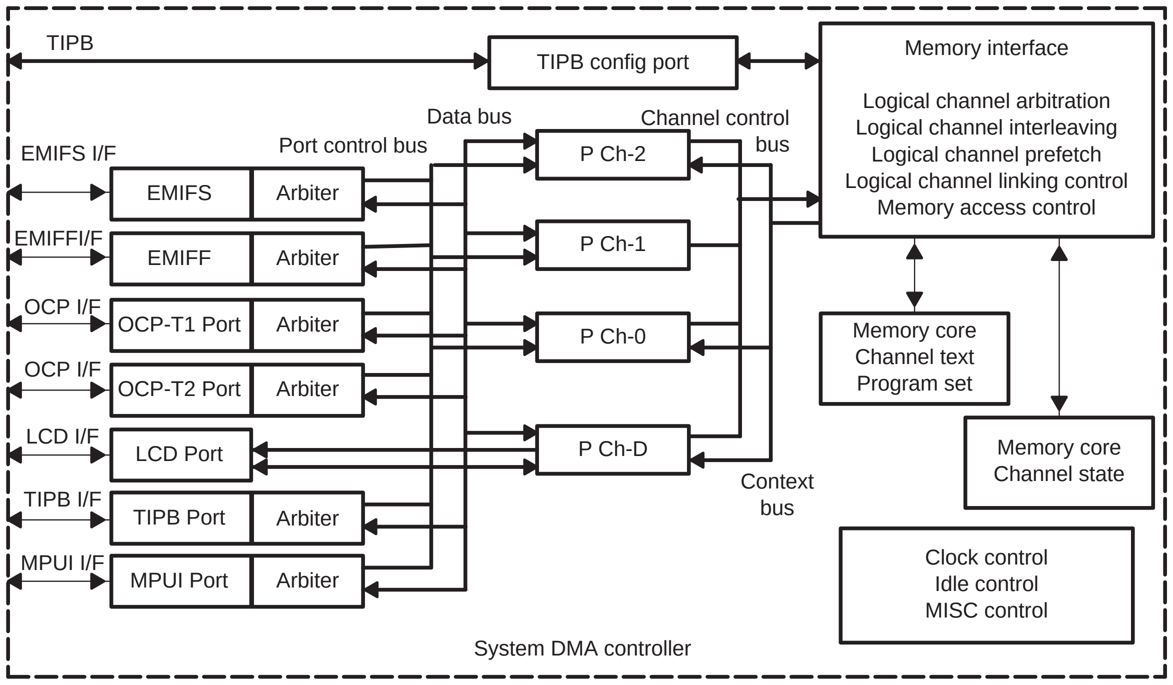
- Two choices of logical channel arbitration of physical resources: round robin or fixed
- Two levels of logical channel priority
- Constant fill
- Transparent copy
- Rotation 0°, 90°, 180°, and 270°
- Seven ports enabling:
  - Memory-to-memory transfers
  - Peripheral-to-memory transfers
  - Memory-to-peripheral transfers
  - Peripheral-to-peripheral transfers
- Binary backward-compatible by default configuration
- Up to four logical channels active in parallel

The logical channel dedicated to the display, LCh-D, has several additional features.

- Channel can be shared by two LCD controllers
- Supports both single and dual block modes
- Supports separate indexing and numbering for dual block mode for both elements and frames

A graphical overview of the system DMA and its external connections is shown in Figure 3.

Figure 3. System DMA Controller Simplified Block Diagram



3.1 Functional Description

This section describes the system DMA capabilities and programming.

The configuration of the system DMA logical channel registers can be done in any order with the exception of the enable bit in the channel control register, DMA_CCR. This bit enables all logical channel types, so this must be the last thing done when configuring a logical channel (LCh). During the time an LCh is enabled it is not allowed to change its configuration registers, which causes undefined effects. All global system DMA configuration registers must be changed before any LChs are enabled; if not, it undefined effects result.

The dedicated LCD channel has some additional features and different channel behavior compared to the generic channels. For more details, see section 3.2.1. Some channel features and behavior are common to both generic and LCD channels; this section describes them and indicates which are supported by the LCD channel.

3.1.1 Logical Channel Types

The OMAP system DMA is based on logical channels (LChs).

Each generic LCh can be configured to one of four different logical channel types; the dedicated LCD LCh can only be configured to LCh type D.

Logical channel types (LCh types) supported are:

- ❑ LCh-2D for nonsynchronized transfers (memory transfers, 1D and 2D)
- ❑ LCh-P for synchronized transfers (mostly peripheral transfers)
- ❑ LCh-PD similar to LCh-P but runs on a dedicated physical channel
- ❑ LCh-G for graphical transfers/operations
- ❑ LCh-D for display transfers

Note:

The logical channel dedicated for the display, LCh-D, must be configured to LCh type LCh-D.

Each logical channel type represents a specific subset of the system DMA features list. Even if the requirements of a specific transfer can fit into several LCh-types, it is important to select the correct LCh-type. Table 16 summarizes features per logical channel type.

Table 16. Summary of Features Per Logical Channel Type (Without LCh-D)

Supported Transfer Features		**LCh-2D (2D)**	**LCh-P (P)**	**LCh-PD (PD)**	**LCh-G (G)**
LCh Control	*Transfer type*				
	❑ Synchronized	√	√	√	√
	Two levels of priority	√	√	√	√
	Preemption at element boundary	√	√		√
	LCh interleaving on DMA request		√	√	
	Linking logical channel capability	√	√	√	√
Addressing	*Types of addressing modes for SRC and DST*				
	❑ Constant mode	√	√	√	√
	❑ Post-increment mode	√	√	√	√
	❑ Single-indexed mode with separate SRC/DST index	√	See Note	See Note	√
	❑ Double-indexed mode with separate SRC/DST index	√	See Note	See Note	√
Other	*Other LCh-type features*				
	Constant fill	√			√
	Transparent color				√

Note: Supported only on one end of the transfer (memory side—can be a source or a destination).

A separate table applies for the LCh-D (D), because several elements differ between that channel and these generic LChs. See Table 29 in section 3.2.1.

The abbreviations in parentheses in the Table 16 headings are used throughout this document as a guide to features supported for different LCh types.

3.1.2 OMAP 3.2 System DMA Instances

The OMAP 3.2 system DMA has 16 generic logical channels plus one logical channel dedicated to the LCD controller as well as three physical channels plus one dedicated to the LCD controller (PCh-0, PCh-1, PCh-2 plus PCh-D).

PCh-0 and PCh-1 are always dynamically allocated to any active LCh of type LCh-2D, LCh-G, and LCh-P. Dynamic allocation means that the physical channel that becomes free first services the active logical channel. PCh-2 is different from PCh-0 and PCh-1, because it is possible to configure one or several synchronized LChs to only use this physical channel. For example, the physical channel can be dedicated to one LCh or to several interleaved synchronized channels. This is done by specifying an LCh as LCh-PD type. The fourth channel, PCh-D, is specifically designed for transfers to the display and can only be used by the dedicated LCh-D. LCh-D must always be configured as type LCh-D.

Table 17. Associated Physical Channels Per Logical Channel Type

LCh-Type	PCh Assigned
LCh-2D	PCh-0 or PCh-1
LCh-P	(dynamic allocation)
LCh-G	
LCh-PD	PCh-2
LCh-D	PCh-D

There are six global registers: PCh status register, PCh ID register, and one PCh status register per PCh channel. See respective register descriptions for more details.

3.1.3 Synchronized Channel

LCh Types Supporting Synchronized Channels	2D	P	PD	G	D
	√	√	√	√	√

A synchronized channel (hardware activated) is a channel that only becomes active when it is enabled by software and subsequently receives a DMA request signal either from an peripheral or from an dedicated ball. There are 31 possible hardware DMA request sources, 1 to 31. A hardware DMA request cannot be shared between several concurrent channels (enabled and active). However, a hardware DMA request can be shared between different channels if they are part of a chain. The peripheral hardware request associated with each of the 31 system DMA request is controllable through MPU GDMA handler configuration.

Software must program the five SYNC bits located in the channel control register, DMA_CCR, to configure the external DMA request that activates the channel. The logical channel becomes a synchronized channel when this field is set to reflect the number of the request line. Remember that no DMA request can be mapped as 0, which is reserved to specify a nonsynchronized transfer.

Each time a DMA request is received for a synchronized channel, the logical channel is activated and a block of data is transferred when a physical channel is assigned to it. This block of data can be:

- ❑ An element

  A complete element, which is defined by Data_type. For example, 8/16/32 bits are transferred in response to a DMA request.

- ❑ An entire frame

  A complete frame of several elements is transferred in response to a DMA request.

- ❑ An entire block

  A complete block of several frames is transferred in response to a DMA request.

One DMA request can trigger several logical channels at the same time.

To configure an LCh to synchronize by element, frame, or block, program the frame synchronization (FS) bit in the DMA_CCR register and the block synchronization (BS) bit in the DMA_CCR2 register. If both bits are set to 0, the channel is synchronized by element. Setting both bits to 1 causes undefined effects.

- ❑ To configure an LCh to transfer *one element* per DMA request:

  1) Configure the data type, also referenced as element size (ES), in the field Data_type in the channel source destination parameter register (DMA_CSDP).
  2) Configure the number of transfers (elements) to take place before the LCh gets disabled again in the channel element number register (DMA_CEN).
  3) Configure both FS and BS to 0.

- ❑ To configure an LCh to transfer *one frame* per DMA request:

  1) Configure the element size as described above.
  2) Configure the element number as described above. This represents the number of elements sent per frame, hence per DMA request.
  3) Configure the number of transfers (frames) to take place before the LCh gets disabled again in the channel frame number registers (DMA_CFN).
  4) Configure FS to 1 and BS to 0.

- To configure an LCh to transfer *one block* per DMA request:
  1) Configure the element size as described above.
  2) Configure the element number as described above. This represents the number of elements sent per frame.
  3) Configure the frame number as described above. This represents the number of frames sent per block.
  4) Configure FS to 0 and BS to 1.

It is also possible to stop a transfer by disabling the channel. This is done by resetting the ENABLE bit in the DMA_CCR register.

3.1.4 Physical Ports

The system DMA can access different data types, depending on which DMA port is used and what the memory or peripheral is supporting. The system DMA ports can support different types of protocols and bus widths. This is important to understand, because it governs which type of transfer an LCh can perform. Table 18 summarizes the type of transfers that each port supports.

Table 18. OMAP3.2 System DMA Supported Interface Port Type Table

Port _Name	Port Functionality
OMAP EMIFF port OMAP EMIFS port OMAP OCP-T1 port OMAP OCP-T2 port	Supports: ❑ 8/16/32-bit, 4x32-bit burst serialized access ❑ 4x32-bit burst serialized access, if it is the source port of the LCD channel.
OMAP TIPB port OMAP MPUI port	Supports: ❑ 8/16/32-bit serialized access
OMAP LCD port	Supports: ❑ 16-bit serialized access for OMAP LCD controller ❑ 32-bit serialized access for OMAP external LCD controller
OMAP TIPB configuration port	TIPB configuration interface for system DMA. It supports 16-bit serialized access.

Six of the physical ports can be selected to be the source and/or destination of a DMA transfer. The DMA LCD port can only function as a destination port. The TIPB configuration port cannot be used as a source or destination port.

- TIPB configuration interface:
  - Used by the MPU to control/configure the system DMA. Cannot be used as source or destination in a DMA transfer.
- External slow memory interface (Flash/ROM):
  - DMA accesses can be either single (8/16/32-bit) or burst (4x32-bit). If burst access is used and data packets are not 16-byte aligned, then single accesses (8/16/32-bit) are performed.
- External fast memory interface (SDRAM, DDR):
  - DMA accesses can be either single (8/16/32-bit) or burst (4x32-bit). If burst access is used and data packets are not 16-byte aligned, then single accesses (8/16/32-bit) are performed.
- OCP-T1 and OCP-T2 interface (Test RAM, Camera peripheral):
  - DMA accesses can be either single (8/16/32-bit) or burst (4x32-bit). If burst access is used and data packets are not 16-byte aligned, then single accesses (8/16/32-bit) are performed.
- TIPB interface (to peripherals via TIPB bridge):
  - All DMA accesses are done in single-access mode (8/16/32 bits).
- MPU interface (to SARAM and DARAM inside DSP subsystem):
  - All DMA accesses are done in single access mode (8/16/32 bits). For 32-bit transfer, the MPU interface does the packing and unpacking.
- LCD interface
  - Port to the OMAP embedded LCD controller. Supports 16-bit access for the OMAP LCD controller.

For all ports, only the number of programmed bytes are transferred; that is, there are no trailing or dirty bytes at the end of transfer.

Table 19 provides possible source ports (SRC), destination ports (DST), and data transfers.

Table 19. Possible Data Transfer

	DST						
SRC	**EMIFS**	**EMIFF**	**OCP-T1**	**OCP-T2**	**TIPB Bridge**	**MPUI**	**LCD**[1]
EMIFS	Yes	Yes	Yes	Yes	Yes	Yes	No
EMIFF	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OCP-T1	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OCP-T2	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TIPB bridge	Yes	Yes	Yes	Yes	Yes	Yes	No
MPUI	Yes	Yes	Yes	Yes	Yes	Yes	No
LCD[1]	No	No	No	No	No	No	No

† Used for the OMAP internal LCD controller.

The port to use for source respective destination is configured in the channel source destination parameters register, DMA_CSDP. The data type is also configured in this register.

Note:

No address space check is performed by the system DMA. All addressing outside the source and destination memory space causes undefined effects.

3.1.5 Port Channel Scheduling

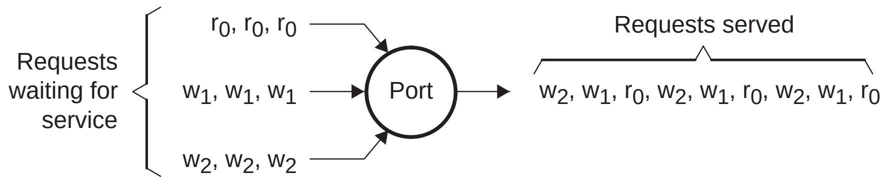
Each DMA logical channel can be configured independently from other logical channels. Each DMA physical channel can have its own port or share it with other physical channels. If physical channels share a DMA port, a port arbiter prioritizes physical channel access.

The system DMA port arbiters follow a round robin scheme.

In Figure 4, a DMA port services three DMA physical channel requests:

- ❑ Physical channel 0 as a source port (read requests r_0)
- ❑ Physical channel 1 as a destination port (write requests w_1)
- ❑ Physical channel 2 as a destination port (write requests w_2)

Figure 4. Time Sharing Access on a System DMA Port


Requests waiting for service
r_0, r_0, r_0
w_1, w_1, w_1
w_2, w_2, w_2
Port
Requests served
$w_2, w_1, r_0, w_2, w_1, r_0, w_2, w_1, r_0$

The system DMA port arbiters use a round robin scheme, but they are also dependent on the LCh priority. For more information about LCh priority, see the section on *Logical Channel Priorities* in section 3.1.6.

The arbiters use the following scheme each time the previous DMA port access has finished:

1) Service all high-priority PChs using a round robin scheme. A low-priority transfer currently being served by the same port is not aborted or suspended by a high-priority transfer. The high-priority transfer waits.

   Port arbitration is each-access based. If the current access is burst access, then the arbitration is on the boundary of burst. The higher priority PCh transfer takes the port as soon as the next available slot is open at the burst boundary.

2) If no highly prioritized PCh is present, then all PCh are arbitrated/served according to the round robin scheme.

3.1.6 Logical Channel Scheduling

A logical channel is marked as an active channel only when either of the following conditions has been meet:

- Case 1**:**
  - The logical channel is a synchronized channel.
  - The logical channel enable field is set.
  - The associated DMA request is triggered.
- Case 2**:**
  - The logical channel is a nonsynchronized channel.
  - The logical channel enable field is set.

A logical channel can be assigned to a physical channel only when the logical channel is active.

When a physical channel is granted, the DMA controller loads the physical channel with the logical channel configuration register set, which controls the data transfer through the system DMA. At the end of a channel transfer, the current channel status is updated into the logical channel status register, and the current logical channel enable is cleared if it is a nonsynchronized channel.

Logical Channel Scheduling Scheme

Each physical channel can only serve one logical channel at a time. If several logical channels are active and waiting to be served, they are interleaved based on an arbitration scheme in a TDMA manner. The supported arbitration schemes are:

- ❏ Round robin scheduling
- ❏ Fixed scheduling from low LCH ID to high LCH ID

The scheme to be used can be controlled by software on a global basis.

In the global control register, (DMA_GCR), the ROUND_ROBIN_DISABLE bit controls which scheme to follow. This bit can only be changed when the DMA is quiescent (that is, when no LChs are enabled). Any change of this bit when a LCh is enabled causes undefined behaviors.

Logical Channel Priorities

	2D	P	PD	G	D
LCh Types Supporting this Feature	√	√	√	√	N/A

Each logical channel can be given a low or high priority level. When a DMA physical channel receives requests from several logical channels, it looks at their priorities. The physical channel assignment to logical channels follow the scheme described below:

1) Requests from high-priority logical channels are served first. A higher priority logical channel can preempt the current on-going low-priority logical transfer and start the higher priority logical channel transferring. The preempted logical channel continues the transfer as soon as all high-priority channels are served. A transfer can be preempted on element boundary. See section 3.1.9 for more information on channel preempting.

2) Requests from low-priority logical channels are served only if there are no requests from high-priority logical channels. This can occur if no high-priority logical channels are activated, or if the high-priority logical channels are waiting for a synchronization event.

3) Requests of the same priority level are served in a round robin or fixed scheduling scheme, as mentioned earlier in this document.

Use the PRIO bit in the logical channel register DMA_CCR to configure the LCh priority.

3.1.7 Logical Channel Interleaving For Synchronized Transfers

LCh Types Supporting this Feature	2D	P	PD	G	D
		√	√		

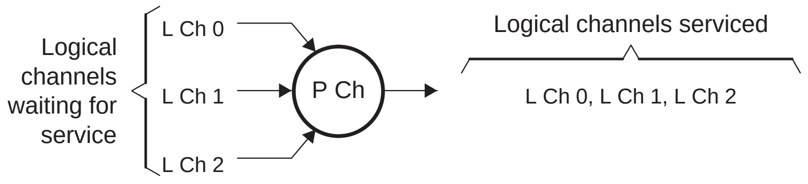
Logical channel interleaving is a term used when more than one synchronized channel shares the same physical channel. A synchronized channel is only active and requesting access to a physical channel when a DMA request is received.

When a DMA request is served but more data remains to be transmitted, the logical channel stays enabled waiting for next DMA request. During this time, the physical channel is released if the LCh-type currently running supports logical channel interleaving.

A nonsynchronized LCh can use the physical channel as well (i.e., interleave), but it runs to finish before it releases the channel again. This means it is the responsibility of the software to configure nonsynchronized transfers in small enough blocks so that synchronized LChs are served in time. This is true only if all PCh are occupied.

Figure 5 shows an example of a logical channel interleaving scheme for three synchronized LChs where the DMA requests are received in order: LCh 0 request, LCh 1 request, and LCh 2 request.

Figure 5. Logical Channel Interleaving on Channel Boundary With the Same Priority


Logical channels waiting for service
L Ch 0
L Ch 1
L Ch 2
P Ch
Logical channels serviced
L Ch 0, L Ch 1, L Ch 2

It is possible to disable the interleaving on a logical channel basis. If this is done for a specific logical channel, that logical channel does not release the physical channel between the DMA requests until all the transfers are done or until the LCh is disabled by software. This is a way of dedicating a physical channel to a logical channel of type LCh-P or LCh-PD.

To disable LCh interleave for synchronized transfers, set the LCH_INTERLEAVE_DISABLE bit in the logical channel control register, DMA_LCH_CTRL.

A nonsynchronized transfer is not affected by the LCH_INTERLEAVE_DISABLE bit in any circumstances. A nonsynchronized transfer always releases the physical channel between transfers even if it is linked to another LCh using the logical channel linking feature.

3.1.8 Linking Logical Channels

LCh Types Supporting This Feature	2D	P	PD	G	D
	√	√	√	√	

Software can configure DMA logical channels to an LCh chain.

To configure and start a linked LCh chain:

1) To link the LChs, configure the NEXTLCH_ID bit in the logical channel link control register DMA_CLNK_CTRL to indicate the next logical channel to be enabled as soon as the current logical channel has finished the transfer.
2) To define the LCh as being part of a linked queue, set the ENABLE_LNK bit in the same register. In this way, the logical channel, defined by NEXTLCH_ID, is enabled after the current channel finishes transferring.
3) Start the chain by enabling the first LCh in the chain (set the ENABLE bit of the DMA_CCR register to 1).

In order to stop a linked chain, software can write to the STOP_LNK bit in logical channel link control register to disable chain and queue. This must be done to each LCh that is part of the chain. Writing to the STOP_LNK bit resets both the ENABLE_LNK bit and the LCh enable bit in the DMA_CCR register. If a linked chain is run to finish without being stopped, the whole chain remains linked; that is, all LChs still have the ENABLE_LNK bit set to 1. To start the same linked chain again, just enable the first LCh in the chain.

An LCh can be linked to all LCH 0 to 15 but not to/from the LCH-D. An LCh can link to itself, and two LChs can link to each other. Linking several LCh in a circular fashion is also possible.

Note:

When a linked LCh is stopped with STOP_LNK, the bit ENABLE_LNK is disabled and hence the LCh is not part of the chain any more.

The restrictions on linking logical channels are:

❑ Constraint 1**:**

  It is the software's responsibility to make sure that all NEXTLCH_ID fields and ENABLE_LNK bits are configured before the logical channel chain is started (first LCh is enabled).

  Undefined effects occur if the software modifies these fields when the chain already is enabled.

- ❑ Constraint 2**:**

  It is the software’s responsibility to make sure that only the head of chained logical channel is enabled. Undefined effects occur if the software enables a logical channel that is inside a chain.

- ❑ Constraint 3:

  It is the software’s responsibility to ensure the chained logical channels have the same priority level; otherwise, undefined effects result.

- ❑ Constraint 4**:**

  It is the software’s responsibility to make sure that the channel context does not change on the fly for a chained logic channel. The channel context must be set up by the software before the head of the chained logic channel is enabled. It can only be updated when the linked chain is disabled. Otherwise**,** undefined effects occur. An exception to this constraint is the STOP_LNK bit.

- ❑ Constraint 5:

  If the chain is configured as a looping chain (for example, the LCh is linked to it self or two LChs are linked to each other), it is the software’s responsibility to disable/stop it.

- ❑ Constraint 6:

  If the bit STOP_LNK is set while the logical channel is running, then the logical channel is deactivated, the transfer is stopped, and no further linked logical channel is activated.

- ❑ Constraint 7:

  It is the software’s responsibility to set the bit ENABLE_LNK to 0 (or use STOP_LNK) for all the LCh of a chain, if the LCh is to be reused without the chaining capability. Otherwise, activating any of these LCh activates the chain starting from this logical channel.

  In other words, the ENABLE_LNK bit is not reset by hardware at the end of a linked logical channel. A linked chain remains linked after the chain has been executed (if STOP_LNK is not used).

- ❑ Constraint 8:

  If a synchronized LCh is part of the chain, it is the responsibility of the software to synchronize the LCh chain with the peripheral so DMA requests are not issued before the LCh is enabled.

  A DMA request is ignored if it is received before the synchronized LCh is enabled by the chain.

3.1.9 Logical Channel Preempting

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√		√	N/A

A logical channel can be preempted on the element boundary, so that the current element and any ongoing bursts are fully transferred before the channel gets preempted.

Preemption occurs when a high-priority LCh suspends a low-priority LCh. This happens if no other PChs are free when the high-priority LCh gets active. For more details, see section 3.1.6.

A synchronized channel goes into an interleaved state once a DMA request is served, and it waits for a new DMA request. It goes into a preempted state once a higher priority logical channel is activated on the same physical channel.

A nonsynchronized channel goes into a preempted state once a higher logical channel is activated on the same physical channel.

A suspended (preempted) low-priority LCh always has a higher priority than a lower priority LCh to be scheduled later. In other words, the preempted LCh does not have to be rearbitrated. However, a suspended LCh takes over the next free PCh. This means that the same PCh is not necessarily to be used if the preempted LCh is of an LCh type supported by several PChs.

An active, preempted, synchronized LCh (the preempted LCh that received a DMA request and was active before the preemption) does not need a new DMA request. The LCh continues the transfer as soon as a PCh becomes free.

If new DMA requests are received when a synchronized LCh is preempted, an event drop is issued on the second new DMA request and the LCH is disabled. If the DROP_IE bit is set to 1 in the DMA_CICR register, then an interrupt is generated and the DROP bit in the DMA_CSR register is set.

The content in the FIFOs in the physical channels is always transferred before the logical channel releases the physical channel. When a physical channel is preempted, it is always on the source element boundary. Therefore, the physical channel drains the FIFO data to the destination before the LCh releases the PCh.

3.1.10 Addressing Modes

An addressing mode is an address computation algorithm that a DMA channel uses to find where to access data. The system DMA supports four types of addressing modes:

- ❑ Constant index mode
- ❑ Post-incremented mode
- ❑ Single-indexed (element index) mode
- ❑ Double-indexed (element and frame index) mode

Based on LCh types, the summary of addressing modes is as follows:

Table 20. Logical Channel Type Address Mode Summary

	LCh-P and LCh-PD			
Addressing Mode	**Peripheral Port**	**Memory Port**	**LCh-2D**	**LCh-G**
Constant	√	√	√	√
Post-incremented	√	√	√	√
Single-indexed		√	√	√
Double-indexed		√	√	√

The amount of data (block size) to be transferred is programmed in bytes.

The data block to transfer is split into frames and elements. The data block size in bytes can be expressed as:

BSi = FN x EN x ES

where:

BSi: The block size in bytes.

FN: The number of frames in the block, $1 \leq FN \leq 65535$ (unsigned) which is defined in the DMA channel frame number register (DMA_CFN).

EN: The number of elements per frame, $1 \leq EN \leq 65535$ (unsigned) which is defined in each DMA channel element number register (DMA_CEN).

ES: The number of bytes per element, $ES \in \{1, 2, 4\}$, which is defined with the field DATA_TYPE in register DMA channel source destination parameters (DMA_CSDP).

A frame size in bytes: FSi = ES x EN.

Setting FN or EN equal to 0 is not allowed, because it causes undefined effects.

An element (data type) can be:

- 8-bit scalar data, s8 (which means that ES = 1)
- 16-bit scalar data, s16 (which means that ES = 2)
- 32-bit scalar data, s32 (which means that ES = 4)

To set up a channel for a transfer, the software must program two addressing modes:

- The source addressing mode, source index size
- The destination addressing mode, destination index size

Both address modes for source and destination are independent. For example, to transfer data from TIPB port to internal memory, the source-addressing mode can be constant and the destination mode can be post-incremented. However, the number of frames, the number of elements, and the element size are the same for both source and destination.

Note:

The channel source and destination start addresses are configured in separate registers: DMA_CSSA_L/U and DMA_CDSA_L/U respectively. Frame index and element index are configured in separate registers for both source and destination: DMA_CSFI and DMA_CSEI, respectively, for source, and DMA_CDFI and DMA_CDEI, respectively, for destination.

Data Alignment

During a transfer, the start address and all the addresses computed by the DMA must be aligned on the type of data transferred (data type):

- If the data type is s8 (8-bit scalar data), then addresses can have any value.
- If the data type is s16 (16-bit scalar data), then addresses must be aligned on 16-bit word boundary (the lowest bit of the address always 0).
- If the data type is s32 (32-bit scalar data), then addresses must be aligned on 32-bit word boundary (the two lowest bit of the address always 00).
- If it is 4x32 burst access with s32 (32-bit scalar data), then addresses must be aligned on burst boundary (the four lowest bit of the address always 0000).
- If frame index is used, it must always produce addresses aligned on data type boundary.
- If element index is used, it must always produce addresses aligned on data type boundary.
- Transfer block size must be aligned on the data type boundary.

Failure to follow these rules generates undefined operation due to wrong endianism switching.

In summary, the general constraints are:

Start address (SA), Element_Index (EI), and Frame_Index (FI) must be aligned on data type (ES):

SA	mod ES=	0
(FI-1)	mod ES=	0
(EI-1)	mod ES=	0

Constant Addressing Mode

Address remains constant for each element to be transferred.

A(n+1) = A(n)

where:

A(n): Byte address of the element n within the transfer.

Note:

A(0) is always equal to the start address of the transfer. The same goes for all addressing modes.

Post-Incremented Addressing Mode

Address is post-incremented by element size (ES).

A(n+1) = A(n) + ES

where:

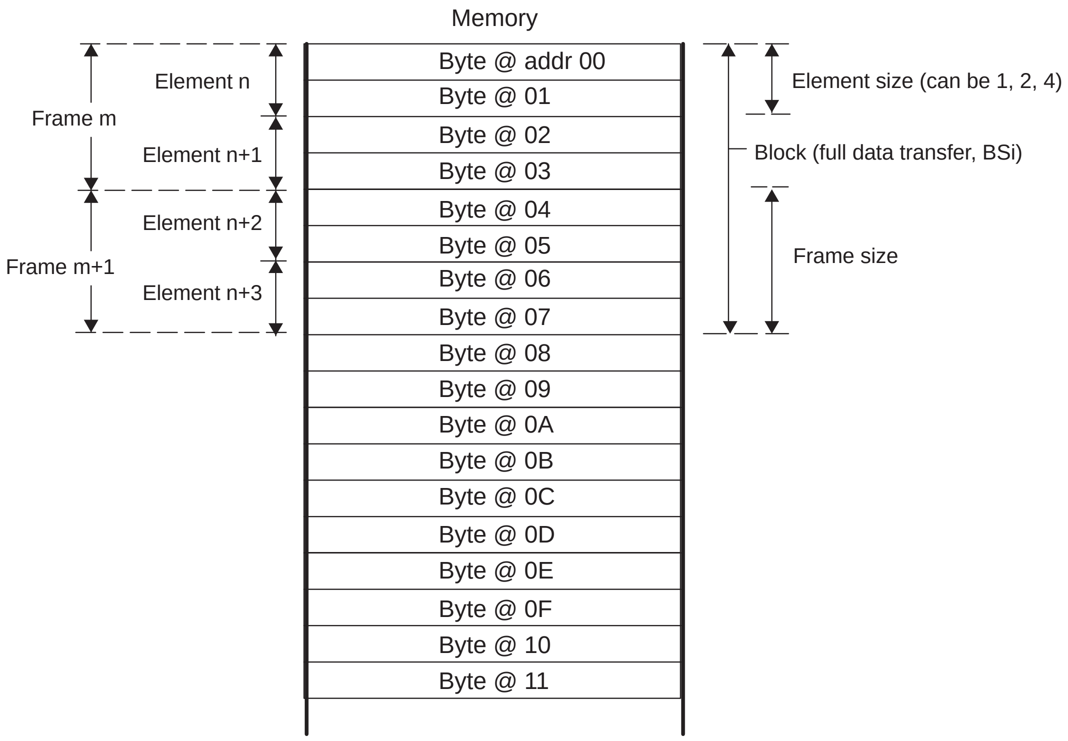
An: Byte address of the element n within the transfer.

ES: Element size in bytes, ES $\in$ {1, 2, 4}.

Figure 6 illustrates how the memory accesses are performed if an LCh is configured with post-increment addressing mode and:

- Starting address: 00
- Element size: 2 (16 bits)
- Element number: 2
- Element index: Ignored
- Frame number: 2
- Frame index: Ignored

Figure 6. Post-Incremented Addressing Mode Memory Accesses



Single-Indexed Addressing Mode

Address is post-incremented by element size and an element index (expressed in bytes).

A(n+1) = A(n) + ES + (EI – 1)

EI = ((Stride EI – 1) * ES) + 1

where:

A(n): Byte address of the element n within the transfer.

ES: Element size in bytes, ES $\in$ {1, 2, 4}.

EI: Element index in bytes, specified in a configuration register, $-32768 \leq EI \leq 32767$.

Stride EI: Number of elements between the beginning of current element, n, to the beginning of next element, n+1.

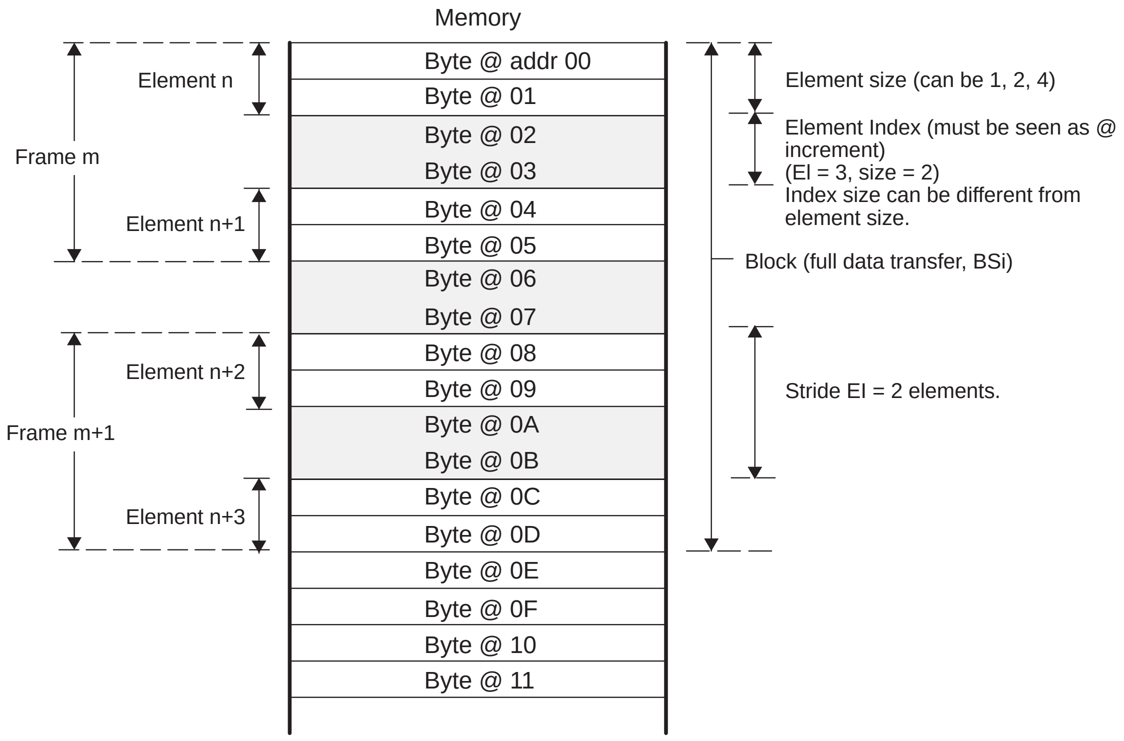
Note:

EI = 1 results in consecutive element accesses (the same behavior as with the post-incremented addressing mode).

Figure 7 illustrates how the memory accesses are performed if a LCh is configured with single indexed addressing mode and:

- ❑ Starting address: 00
- ❑ Element size: 2 (16 bits)
- ❑ Element number: 2
- ❑ Element index: 3
- ❑ Frame number: 2
- ❑ Frame index: Ignored

Figure 7. Single-Indexed Addressing Mode Memory Accesses



Note: EI used between frames for single-indexed addressing mode. This is why this figure has three equal, big strides between the elements.

Double-Indexed Addressing Mode

Address is incremented by element size and a frame index if the end of the current frame is reached. Address is incremented by element size and an element index if the end of the current element is reached but the end of frame is not reached.

When not at end of a frame or transfer, that is, as long as element counter $\neq 0$:

$A(n+1) = A(n) + ES + (EI – 1)$

When at end of a frame but not at the end of the transfer, that is, as long as element counter = 0 and frame counter $\neq 0$:

$A(n+1) = A(n) + ES + (FI – 1)$

Calculate element and frame index as follows:

$EI = ((\text{Stride } EI – 1) * ES) + 1$
$FI = ((\text{Stride } FI – 1) * ES) + 1$

where:

A(n): Byte address of the element n within the transfer.

ES: Element size in bytes, $ES \in \{1, 2, 4\}$.

EI: Element index in bytes, specified in a configuration register, $-32768 \leq EI \leq 32767$.

Stride EI: Number of elements between the beginning of current element, n, to the beginning of next element, n+1.

Element: Counter that is (re)initiated with the number of element per frame or per transfer. Decreased by 1 for each element transferred. Initial value is configured in register DMA channel element number, DMA_CEN.

FI: Frame index in bytes, specified in a configuration register, $- 32768 \leq FI \leq 32767$.

Stride FI: Number of elements between the beginning of last element of the current element and the beginning of first element of the next frame.

Frame counter: Counter that is (re)initiated with the number of frames per transfer. Decreased by 1 for each frame transferred. Initial value is configured in register DMA channel frame number, DMA_CFN.

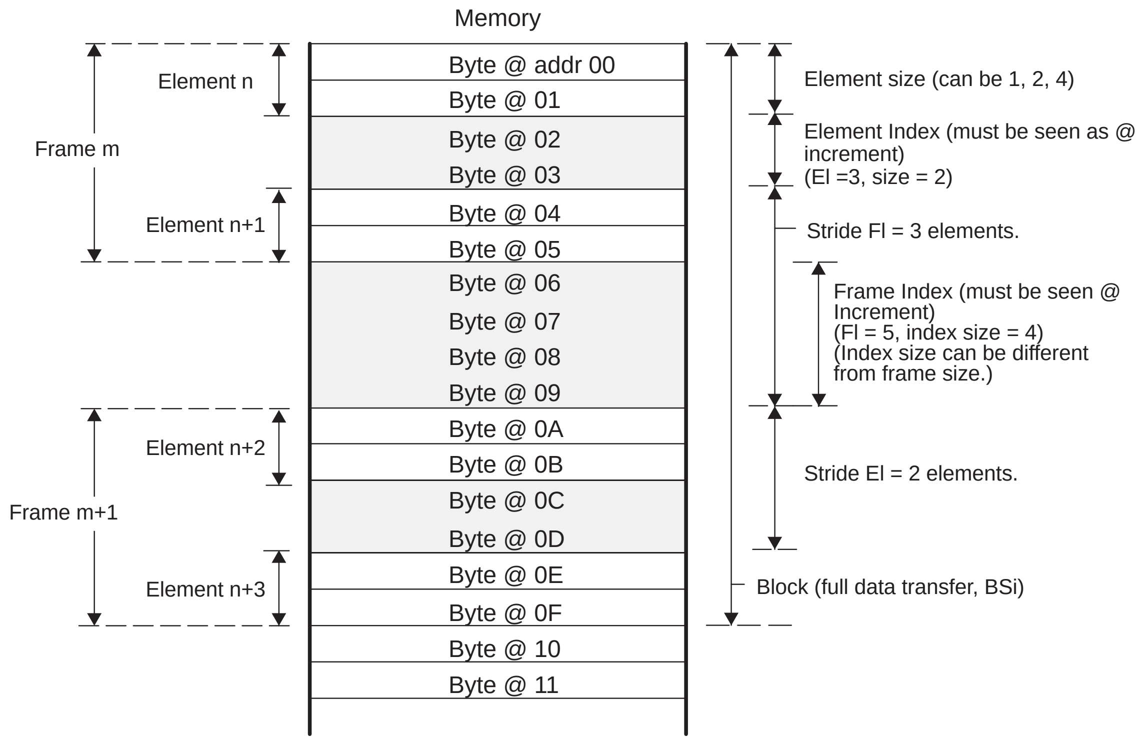
Note:

FI = 1 provides consecutive element accesses at end of frames. If EI = 1, the behavior is the same as with the post incremented addressing mode.

Figure 8 illustrates how the memory accesses are performed if an LCh is configured with double-indexed addressing mode and:

- Starting address: 00
- Element size: 2 (16 bits)
- Element number: 2
- Element index: 3
- Frame number: 2
- Frame index: 5

Figure 8. Double-Indexed Addressing Mode Memory Accesses



Note:

For all addressing modes, independent element and frame indexing were not supported for OMAP 3.0 and 3.1. The EI and FI configuration is dependent on the OMAP3_1_Compatible_Disable bit in the DMA_CCR register.

If DMA_CCR[OMAP3_1_Compatible_Disable]= 0:

- ❑ EI(source) = EI(destination) = DMA_CSEI register
- ❑ FI(source) = FI(destination) = DMA_CSFI register

If DMA_CCR[OMAP3_1_Compatible_Disable] = 1:

- ❑ EI(source) = DMA_CSEI register
- ❑ EI(destination) = DMA_CDEI register
- ❑ FI(source) = DMA_CSFI register
- ❑ FI(destination) = DMA_CDFI register

3.1.11 Data Packing and Bursting

LCh Types Supporting this Feature	**2D**	**P**	**PD**	**G**	**D**
	√	√	√	√	√

A DMA channel has the capacity to:

- ❑ Pack several consecutive byte accesses in a single word16 or word32 access. This increases the transfer rate. For a channel, the decision to pack to its source port is dependent on the source port access capability and whether source packing is enabled by software. The same goes for the destination port. Software can control packing for both destination and source with the bits SRC_PACK and DST_PACK in the DMA channel source destination parameters register (DMA_CSDP).
- ❑ Split a single access, defined by its DMA Port capability, into subset access size if the address of single access is not aligned on access size. For example:
  - ■ Split a word16 transfer into several byte accesses, if access address is not aligned on word16 address.
  - ■ Split a word32 access into single word accesses, if access address is not aligned on word32 boundary.
  - ■ Split 4x32-bit burst access into several byte/word/double-word accesses, if access address is not aligned on 4x32-bit burst access boundary.

- ❑ Burst 4x32 bits if bursting is enabled and the accessed source or destination port supports it. Software can enable or disable bursting with the bit-fields SRC_BURST_EN and DST_BURST_EN in the DMA_CSDP register. These bit-fields are ignored if the accessed port does not support bursting and hence, result in single accesses.

Table 21 summarizes the possible transfer configurations and shows the cases where packing and splitting are performed.

Table 21. Packing and Splitting Summary

Data Type	Port Access Capability	Packing / Splitting
s8	s8	-
	s16	pack 2 x s8 => 16
	s32	pack 4 x s8 => 32
s16	s8	split s16 => 2 x 8
	s16	-
	s32	pack 2x s16 => 32
s32	s8	split s32 => 4 x 8
	s16	split s32 => 2 x 16
	s32	-

To compute the type of an access (8/16/32-bit or 4x32-bit burst) and decide to pack consecutive accesses, an address calculation unit checks:

- ❑ Its related DMA port capabilities:
  - ■ Can the port perform bytes, 16-bit, or 32-bit access?
  - ■ Can the port perform 4x32-bit burst access?
- ❑ What is allowed by the software in the configuration registers:
  - ■ Is packing allowed?
  - ■ Is bursting allowed?
- ❑ The last bits of the address:
  - ■ Is the address even or odd?
  - ■ Is the address word16, word32, 4x32-bit burst aligned?
- ❑ The number of elements remaining in the frame
- ❑ The synchronized transfer type for the LCh

The restrictions for packing and bursting are as follows:

- ❑ Constant address mode automatically disables packing and bursting on the DMA source or destination port where it is used. The other end of the DMA channel can still utilize packing and bursting if that DMA port is not configured to use constant address mode.

- All elements must be transferred continuously within a frame. That means that it cannot be synchronized on element boundary, and if single/dual-indexed addressing mode is used, element index (EI) must be set to 1.
- No packing or bursting can be done over frame boundary. It does not help to set frame index FI_ to 1.
- No packing or bursting can be done over block boundary.

Once the type of access is decided, the current byte address can be incremented by 1, 2, 4, or 16 to reach the next memory space location. Then, the DMA port checks its physical channel FIFO to see if there is enough data (write access) or enough space (read access) in the FIFO, before issuing the access.

It is assumed that:

- A 16-bit target memory or peripheral also has a byte-write/read capability.
- A 32-bit target memory or peripheral also has a byte and word16 write/read capability.

If this is not the case, then the software must carefully set up the transfer as follows:

- Follow the address alignment rules. See the *OMAP5910/5912 Multimedia Processor DSP Subsystem Reference Guide* (SPRU890) for information on the DSP DMA.
- Set the data_type to the same size as the peripheral register width that is the source or the destination of the DMA transfer.
- These rules apply only when the constant address mode is used (which is likely to be the case when a single register in source/destination), and no packing can be enabled.

Packing is controlled by the source and destination packing bits and packs several elements into a larger access element (word16 or word32). For instance, if the data_type is set to s16, the source port supports 32-bit accesses, the source packing is enabled, and the address is aligned on word32, then the source port makes two 16-bit accesses and packs them to a word32 before sending word32 to the destination port. If the source port uses the constant addressing mode and the packing bit is enabled, then the packing bit is ignored.

Another sort of packing groups bytes or word16 accesses within an element. This is always done if the LCh is configured as such. Packing within an element is done if the access types are smaller than the element sizes (data_type). For instance, if the data_type is set to s32 but the source target port only supports 16 bits, then two consecutive word16 source accesses are packed to build the word32 element. This element is then sent to the destination port. This is done independently of the source and destination packing bits.

Packing within an element is supported even if the constant addressing mode is used or if EI is different from 1.

Example 1 shows an example of packing enabled.

Example 1. Packing Enabled

A detailed example (see Table 22 and Table 23) of packing 2 x s16 => 32 is:

- ❑ A logical channel is set up for a transfer with the following parameters for its source:
  - Number of frames in the block: FN = 2 elements
  - Number of elements per frame: EN = 5 elements
  - Type of data: s16
  - Frame index in bytes: FI = 13 bytes
  - Element index in bytes: EI = 1 byte
  - Source start address: SA = 2 bytes
  - Source addressing mode: Double-indexed addressing mode
  - The source port is a 32-bit port with byte/word16/word32 access capability.
  - Packing is enabled but burst is disabled.
  - Memory block to transfer is: element i, j (where j is the element number of frame i).
  - i and j start with the highest number and decreases, so the first element written is element 2,5.

Table 22. Channel Data Block to Transfer

Address	Byte 0	Byte 1	Byte 2	Byte 3
0			element 2,5	
4	element 2,4		element 2,3	
8	element 2,2		element 2,1	
12				
16				
20				
24	element 1,5		element 1,4	
28	element 1,3		element 1,2	
32	element 1,1			
36				
40				

The computed addresses and access type are:

Table 23. Channel Addresses and Access Types

Access	Frame Number j	Element Number i	Address	Access Type
0	2	5	2	16 bits
1	2	4	4	32 bits
2	2	2	8	32 bits
3	1	5	24	32 bits
4	1	3	28	32 bits
5	1	1	32	16 bits
6		End of transfer		

In this example, the first word16 is not packed to word32 because the address is not aligned on word32. The next accesses are packed because packing is enabled, constant addressing is not used, EI = 1, and the address is aligned on word32.

Example 2. Packing and Burst Enabled

Example 2 shows an example (see Table 24 and Table 25) with source packing 2 x s16=> 32 plus burst 4x32-bit enabled,. This example results in exactly the same behavior as Example 1, because the burst capability is ignored. It is ignored because when the address is on a 4x32-bit aligned boundary, less than 4x32 bits are left in the frame.

If the frame size is increased by doubling the element size, EN = 10, the example is as follows (assumed that the targeted source support 4x32 bits burst):

A logical channel is set up for a transfer with the following parameters for its source:

- Number of frames in the block: FN = 2 elements.
- Number of elements per frame: EN = 10 elements.
- Type of data: s16.
- Frame index in bytes: FI = 9 bytes.
- Element index in bytes: EI = 1 byte.
- Source start address: SA = 14 bytes.
- Source addressing mode: Double indexed addressing mode.
- The source port is a 32-bit port with byte/word16/word32/4x32bit burst access capability.
- Source packing is enabled.
- Source burst is enabled.
- Memory block to transfer is: element i, j (where j is the element number of frame i).
- Note that i and j starts with the highest number and decreases, so the first element written is element 2,10.

Table 24 shows the channel blocks to transfer.

Table 24. Channel Data Block to Transfer

Address	Byte 0	Byte 1	Byte 2	Byte 3
12				Element 2,10
16		Element 2,9		Element 2,8
20		Element 2,7		Element 2,6
24		Element 2,5		Element 2,4
28		Element 2,3		Element 2,2
32		Element 2,1		
36				
40				Element 1,10
44		Element 1,9		Element 1,8
48		Element 1,7		Element 1,6
52		Element 1,5		Element 1,4
64		Element 1,3		Element 1,2
68		Element 1,1		
72				

Table 25 lists the computed addresses and access types.

Table 25. Channel Addresses and Access Types

Access	Frame Number j	Element Number i	Address [byte]	Access Type
0	2	10	10	16 bits
1	2	9-2	12	4x32 bits
2	2	1	28	16 bits
3	1	10	42	16 bits
4	1	9,8	44	32 bits
5	1	7,6	48	32 bits
6	1	5,4	52	32 bits
7	1	3,2	64	32 bits
8	1	1	68	16 bits
9	End of transfer			

In this example the first word16 is not packed to word32 because the address is not aligned on word32. The next accesses bursts 4x32 bits, because burst is enabled, constant addressing is not used, EI = 1, address is 4x32 bits burst aligned, and enough data is left in the frame (16 bytes). All other transfers are word16 single accesses or 32 bit packed accesses. There are no more bursts because the next time the address is burst aligned, there is not enough data left in the frame (less than 16 bytes).

3.1.12 Interrupt Generation

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√	√	√	√

Each generic LCh can generate six different interrupts. The following interrupt sources can be programmed:

- ❑ End of block: The last byte of the transfer has been written into destination, enabled with bit BLOCK_IE in register DMA_CICR.
- ❑ End of frame: The last byte of the current frame has been written into destination, enabled with bit FRAME_IE in register DMA_CICR.
- ❑ Half of frame: The middle byte of current frame has been written into destination, enabled with bit HALF_IE in register DMA_CICR.
- ❑ Start of last frame: The first word of the last frame has been written into destination, enabled with bit LAST_IE in register DMA_CICR.
- ❑ Request collision: Two new DMA requests occurred before the end of service of previous request, enabled with DROP_IE bit in register DMA_CICR.

- ❑ Timeout error: An access error occurred in the transfer to the source or the destination, enabled with bit TOUT_IE in register DMA_CICR. The countdown values are configured at the source and destination ports. No countdown value can be specified in the system DMA.

Note:

Configure the interrupt(s) to be generated for each generic LCh in the channel interrupt control register (DMA_CICR). The LCh-D supports only two kinds of interrupts. See Table 77, *DMA LCD Control (DMA_LCD_CTRL)* for more information.

Each DMA logical channel can generate an interrupt to the MPU to reflect the transfer status. Each system DMA logical channel has a dedicated interrupt line to the MPU. All DMA interrupts are level sensitive interrupts; that is, an interrupt line is held active-low until the MPU reads the associated logical channel status register.

No new interrupts can be generated until the status register is read and thereby cleared. For each logical channel, all the interrupt sources are connected together to generate one interrupt. When an interrupt is issued by a logical channel, its status register, DMA_CSR, is set to record the interrupt cause if its associated DMA_CICR enable bit is set. The MPU interrupt service routine (ISR) can read this channel status register to find the sources of the interrupt. The status bits are automatically cleared after they are read by MPU.

The interrupt enable bits are used to choose the events that trigger the DMA channel to send an interrupt to the processor. There are two classes of events:

- ❑ Error event: errors during the transfer. These are timeout and request collision.
- ❑ Status event: DMA transfer status during DMA channel transfers. These are start of last frame, half of frame, end of frame, and end of block.

When an error event occurs and the corresponding interrupt enable bit is enabled, the following happens:

- ❑ The status register bit is activated.
- ❑ An interrupt is generated.
- ❑ The logical channel is disabled.
- ❑ The physical channel is released.

If there is an error but error interrupt is not enabled, no event is generated, the status register bit is not set, and no interrupt is generated. However, the LCh is disabled.

When a status event occurs and the corresponding interrupt enable bit is enabled, the following happens:

- ❑ The status register bit is activated.
- ❑ An interrupt is generated.
- ❑ No new interrupts can be generated until the status register is read and thereby cleared.

Note:

- ❑ One read in the status register clears all the status bits.
- ❑ No read in the status register keeps all the status bits and DMA interrupt output stays active low.

Therefore, software must always read the associated status register for each DMA interrupt received; otherwise, interrupts can be missed.

System DMA Interrupt Mapping

System DMA interrupt mapping depends on the compatibility mode used.

If the OMAP3_1_COMPATIBLE_DISABLE bit is set to 1 in the DMA_CCR register, then the system DMA has one interrupt line per logical channel.

If this bit is set to zero, then several channels can share the same interrupt line. See 3.1.16 for more details.

Each logical channel has an associated status register, DMA_CSR, where the source of the interrupt is shown.

3.1.13 DMA IDLE Modes

The system DMA can automatically enter an idle mode dynamically as soon as it is not active, or it can be put into idle/suspend mode on request from MPU via the clock generator module.

Dynamic Idle Mode

To save power, the system DMA has a built-in dynamic idle mode that is enabled by setting the CLOCK_AUTOGATING_ON bit in the global control register, DMA_GCR.

The system DMA clock domain is split into several subdomains in this mode; each one of them is disabled if not used. This mode does not add any extra latency in the system DMA.

All internal clocks are in idle mode, disabled, when the following is fulfilled:

1) Clock_Autogating_on = 1 in DMA_GCR.
2) No nonsynchronized LChs are enabled.
3) Either no synchronized LChs are enabled, or synchronized LChs are enabled but no DMA request is received or pending.

The system DMA wakes up if software enables a new LCh or if a DMA request is received. The system DMA also wakes up temporarily if the MPU or OCP-I wants to read or write to any of the registers.

System IDLE Request

The system DMA can be put into IDLE mode, when system needs to go into power saving mode. See the *OMAP5912 Multimedia Processor OMAP3.2 Subsystem Reference Guide* (SPRU749) and the *OMAP5912 Multimedia Processor Power Management Reference Guide* (SPRU753) for more information on prerequisite steps to put the system and/or the system DMA in idle. An idle request signal is sent to the DMA when the system wants it to go idle.

As soon as the DMA detects an idle request, it enters idle mode when all PChs are free (including PCh-D). All nonsynchronized LChs, synchronized LChs without DMA request detected, and suspended LChs are unscheduled PChs. The DMA enters idle mode even if one of the LCD controllers is assigned PCh-D but is in sleep mode. The DMA temporarily wakes up if:

❏ A synchronized LChs DMA request is detected

❏ There are any writes to generic LCh registers

❏ Any of the LCD controllers gets active

The DMA goes back to idle mode when all these tasks are completed. This also means as long as the system is requesting the system DMA to be in idle, nonsynchronized LChs can be enabled, but they are never activated. Synchronized LChs can be enabled and hence, activated at DMA request during this time.

The DMA finishes pending accesses of physical channels and resumes the logical channel transfer when released from idle. Therefore, it is the responsibility of software to ensure that when software issues IDLE instruction, all logical channels have been serviced.

The power-saving difference between the dynamic idle mode and the system idle request is provided by the clock tree between the clock generation module and the system DMA clock input.

3.1.14 DMA Debug State

During debug mode the MPU can send a request to suspend the DMA. This is useful if, for instance, the MPU is halted by a breakpoint. How the system DMA responds to this request is controlled by software by configuring the FREE bit in the DMA global control register, DMA_GCR. When the FREE bit is set to 0, all current transfers are suspended when a request is received from the MPU. The transfers resume when the MPU releases the debug request signal. If the FREE bit is set to 1, the DMA continues to run as usual, even if the MPU is sending a debug request signal. The channel status of the DMA interrupts is read on the DMA_CSR register bits. In contrast with the functional mode, the DMA_CSR bits are not cleared after an emulation read.

3.1.15 Other Logical Channel Features

The LCh-G type channel not only transfers data, but also provides hardware with 2-D graphic data processing features to improve 2-D graphic processing speed and graphic quality. It supports the following graphic features:

- ❑ Transparent copy
- ❑ Constant fill (or constant solid color fill)
- ❑ Rotation

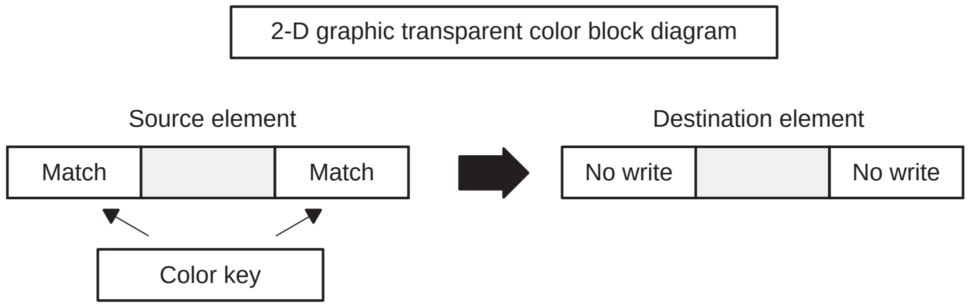
LCh-2D also supports constant fill and rotation. Rotation is supported by all LCh types supporting separate element and frame indexed addressing (2 dimension addressing).

Transparent Copy

LCh Types Supporting this Feature	2D	P	PD	G	D
				√	

It is often desirable to transfer irregular shapes, especially in software for games. The system DMA supports the COLOR KEY feature for 8 BPP, 16 BPP, and 32 BPP from source to destination; that is, each element of channel source is compared to a color key and those data bits (pixels) that match the color key are not written to the destination.

Figure 9. 2-D Transparent Color Block Diagram



This feature is enabled by setting the TRANSPARENT_COPY_ENABLE bit in the channel control 2 register, DMA_CCR2. If this bit is enabled, the DMA color parameter registers, DMA_COLOR_L and DMA_COLOR_U, are used to specify the color key.

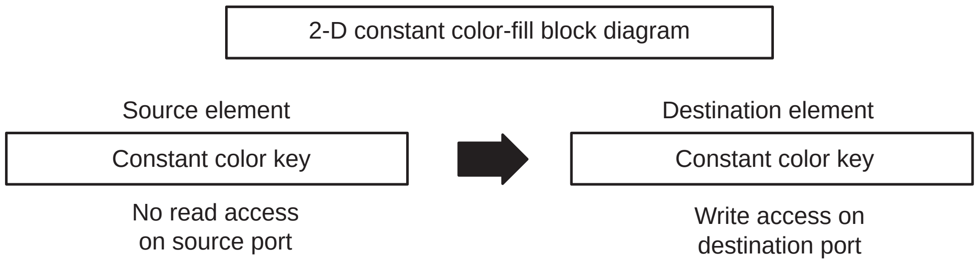
If the data type, ES, is 8 or 16 bits, only the DMA_COLOR_L register is used; if the data type is 32 bits, the DMA_COLOR_U register is also used. The system DMA always compares all bits (8, 16, or 32 bits). If other color depths (other than 8, 16, or 32) are used, it is the responsibility of the software to ensure that unused upper bits are constant. See the register description for more detailed information.

Constant Fill

LCh Types Supporting this Feature	2D	P	PD	G	D
	√			√	

This feature allows filling a region with a constant (solid color) or a pattern by repeating the data horizontally and/or vertically in the region. The system DMA writes to the destination a constant color/value defined in a register. There is no read access on any system DMA source port.

Figure 10. 2-D Constant Color Fill Block Diagram


2-D constant color-fill block diagram
Source element
Constant color key
No read access on source port
Destination element
Constant color key
Write access on destination port

This feature is enabled by setting the bit Constant_Fill_Enable in the channel control register 2, DMA_CCR2. If this bit is enabled, the register DMA color parameter, DMA_COLOR_L and DMA_COLOR_U, is used to specify the color key.

If the data type, ES, is 8 or 16 bits, only the DMA_COLOR_L register is used; if the data type is 32 bits, register DMA_COLOR_U is also used. See the register description for more detailed information.

Rotation

LCh Types Supporting this Feature	2D	P	PD	G	D
	√	√	√	√	√

All LCh types with independent source and destination indexing support four types of rotation: 0°, 90°, 180°, and 270°. This can be established with the help of double indexed addressing; that is, separate element/frame indexing. See section 3.1.10 for details of this addressing capability.

3.1.16 Compatibility with OMAP 3.0 and 3.1

After reset, the DMA and all LChs default to the OMAP 3.0/3.1 programming model. This is characterized by a reduced feature set, no LCH chaining, fewer LChs, program and active register sets per LCh, and a restricted number of interrupts (see Table 26).

New code for the DMA must use the programming model presented in previous sections of this document. Future versions of the DMA will be optimized for this programming model and new features will be added to it rather than the 3.0/1 compatible mode.

The DMA described here does not exhibit the same cycle-by-cycle behavior as previous implementations in OMAP 3.0/3.1 and hence time critical applications must be reverified when upgrading.

Two orthogonal mechanisms control the compatibility modes of the system DMA:

- A global register bit controls the DMA register and interrupt mapping.
- A register bit per logical channel controls the disabling/enabling of the OMAP 3.0/1 programmers model.

Note:

OMAP 3.2 is configured from reset to be OMAP 3.1 binary compatible.

When in OMAP 3.1, compatible mode PCh-0 and 1 are dynamically shared to serve the generic LChs.

To enable the new features introduced beyond OMAP 3.1, the following configuration register bits must be configured:

- ❑ OMAP 3_1_MAPPING_DISABLE in register DMA_GSCR
- ❑ OMAP3_1_COMPATIBLE_DISABLE in register DMA_CCR/DMA_LCD_CCR
- ❑ LCh register bit that enables/disables the OMAP 3.2 programming model.

Table 26. Summary Table of Different Compatibility Modes

Global Register Bit OMAP3_1_Mapping_Disable	Global Effects	LCh Register Bit OMAP3_1compatible_disable	Per-LCH Effects
0	❑ Reset values ❑ 9 LCh + 1 LCh-D available ❑ OMAP 3.1 configuration register mapping ❑ OMAP 3.1 interrupt line mapping ❑ 7 interrupt lines	0	Only OMAP 3.1 programming model/feature set is supported. PCh-2 cannot be used, so only the LCD channel and two generic logical physical channels are available to these LChs.
		1	All features are available as described in the remainder of this document. All four physical channels are available to these LChs.
1	❑ 16LCh + 1 LCh-D available ❑ OMAP3.2 configuration register mapping ❑ OMAP3.2 interrupt line mapping ❑ 17 interrupt lines	0	Only OMAP 3.1 .programming model/feature set is supported. PCh-2 cannot be used, so only the LCD channel and two generic physical channels are available to these LChs.
		1	All features are available as earlier described in this document. All four physical channels are available to these LChs.

Note: The LCh compatibility bit, OMAP3_1_compatible_disable, is configured per logical channel, so it is possible to mix the programmer model for different logical channels.

The OMAP 3_1_MAPPING_DISABLE bit in register DMA_GSCR is a global register bit that enables/disables OMAP 3.1 register mapping and interrupt line mapping. DMA_GSCR is a global register and must therefore be configured before any LCh is enabled. The DMA_CCR/DMA_LCD_CCR registers are configured per LCh, which enables having different programming models for the LChs. The register and interrupt line mapping must be configured first and affect all the LChs, but the OMAP 3.0/3.1 and OMAP 3.2 programming models, respectively, can be mixed among the LChs.

Autoinitialization of Logical Channels

In OMAP 3.0/3.1 compatible mode, autoinitialization is supported.

If a logical channel is autoinitialized, then the logical channel is automatically enabled itself again when a transfer is ended. A new or old logical channel configuration register set is loaded, and a new block of data is transferred.

Autoinitialization means that the logical channel gets reenabled. It does not mean that the physical channel is kept for the logical channel. The logical channel must return to the same scheduling schemes as normal before it gets access to a physical channel.

The autoinitialization bit is used to set the DMA into autoinitialization mode while END_PROG and REPEAT bits are used to control DMA behavior while in autoinitialization mode. The bits AUTO_INIT, END_PROG, and REPEAT are all located in the channel control register, DMA_CCR. Table 27 summarizes the autoinitialization mode.

Table 27. Autoinitialization Configuration Bits Summary

Auto_init	Repeat	End_prog	Autoinitialization Behavior
0	Don't care	Don't care	No autoinitialization. Waits until software sets enable = 1 to enable the LCh and loads the PCh with its programming register set.
1	0	0	As both repeat and end_prog bits equal 0, the LCh is still enabled, but pending. At the end of the current transfer, the LCh channel waits until repeat or end_prog = 1 to re-activate itself again. When end_prog = 1, the programming register set is copied to the active register set: a new context is programmed.
1	0	1	At the end of the current transfer, the LCh immediately loads the PCh with its programming register set when PCh is granted (end_prog = 1 allows loading the new LCh context, disregarding the repeat bit). The channel reinitializes itself and starts a new transfer with the new context.
1	1	Don't care	The LCh reinitializes itself at the end of the current transfer and starts a new transfer with the previous channel context (active register set).

This table differs slightly from what is implemented in OMAP 3.0/3.1 hardware, where the programming register set is always loaded when END_PROG or REPEAT is set to 1.

OMAP3.0/3.1 System DMA Interrupt Mapping Rule

The OMAP 3.1 system DMA has nine channels plus one LCD channel, which share seven interrupt lines; the OMAP 3.2 system DMA has one interrupt line per channel. In order to be backward compatible, the below interrupt mapping modes have been implemented.

To control the mappings:

OMAP 3.2 Mapping: DMA_CCR.OMAP3_1_Mapping_Disable = 1

OMAP 3.1 Mapping: DMA_CCR.OMAP3_1_Mapping_Disable = 0

See the *OMAP5912 Multimedia Processor Interrupts Reference Guide* (SPRU757) for more details about interrupts.

Table 28. Interrupt Mapping per LCh for Both Compatible Modes

Interrupt Line	OMAP 3.2 Mapping	OMAP 3.1 Mapping
MPU level 1 IRQ 19	LCH_0	LCH_0 and LCH_6
MPU level 1 IRQ 20	LCH_1	LCH_1 and LCH_7
MPU level 1 IRQ 21	LCH_2	LCH_2 and LCH_8
MPU level 1 IRQ 22	LCH_3	LCH_3
MPU level 1 IRQ 23	LCH_4	LCH_4
MPU level 1 IRQ 24	LCH_5	LCH_5
MPU level 2 IRQ 53	LCH_6	N/A
MPU level 2 IRQ 54	LCH_7	N/A
MPU level 2 IRQ 55	LCH_8	N/A
MPU level 2 IRQ 56	LCH_9	N/A
MPU level 2 IRQ 57	LCH_10	N/A
MPU level 2 IRQ 58	LCH_11	N/A
MPU level 2 IRQ 59	LCH_12	N/A
MPU level 2 IRQ 60	LCH_13	N/A
MPU level 2 IRQ 61	LCH_14	N/A
MPU level 2 IRQ 62	LCH_15	N/A
MPU level 1 IRQ 25	LCH_D	LCH_12

In case of simultaneous events in two physical channels that share the same interrupt line, only one interrupt is generated and all the relevant status bits are set.

Each physical channel has a seven-bit status register. When an interrupt is shared by two logical channels, the MPU can read the status from the two channels in one TIPB access. The data read has the format shown in Figure 11.

Figure 11. DMA Packed Channel Status Register for Compatible Mode

	[13:7]	[6:0]
00	DMA_CSR [LCH_6] [6:0]	DMA_CSR [LCH_0] [6:0]

3.2 LCD Channel

3.2.1 Display Logical Channel

The display logical channel, LCh-D, transfers data to the LCD controller from a video block buffer stored in memory. In the OMAP 3.2 system, the memory source for the transfer can be Test RAM (OCP-T1), or EMIFF. These transfers can be arranged to have the source in one or two blocks.

Table 29 gives a summary of features for the LCD LCh type.

Table 29. Features Summary for LCh-D

Supported Transfer Features		**LCh-D (D)**
LCh Control	Transfer type	
	❑ Memory to external LCD controller	√
	❑ Memory to OMAP LCD controller	√
	❑ Nonsynchronized	√
	❑ Synchronized	See Note 1
	Two levels of priority	
	Preemption at element boundary	
	LCh interleaving on DMA request	
	Linking logical channel capability	
	Automatic initialization	√
Addressing	Types of addressing modes for src and dst	
	❑ Constant mode	
	❑ Post increment mode	See Note 2
	❑ Single-indexed mode, with separate SRC/DST index	See Note 2
	❑ Double-indexed mode, with separate SRC/DST index	See Note 2
	❑ Supports single and dual block modes	See Note 2
	❑ Supports separate element/frame index and element/frame numbers for dual block mode	See Note 2

Notes: 1) LCh-D can be synchronized with the external LCD controller but not with the OMAP internal LCD controller.

2) The LCh-D destination port is the LCD port. The access address to the display is controlled by the LCD controller. Hence, these features only apply on the source.

The dual-block mode allows concurrent transfer and image processing (reloading of one block while a second is being processed). Switching from one block to another is achieved by loading the configuration register of the second block buffer and then starting the second block transfer after the first block buffer has been fully transferred. This is automatic, enabling the dual-block feature by setting the BLOCK_MODE bit in the DMA LCD control register (DMA_LCD CTRL).

Separate element/frame index and number of elements between the two buffers are also supported in the dual-block mode.

The LCD channel sends the read request to the relevant port, defined by the lcd_source_port bit in the DMA LCD control register (DMA_LCD CTRL) just as the destination port is selected by the lcd_destination_port in the same register. The destination port can be either the OMAP internal LCD controller or the external LCD controller port.

The dedicated DMA channel contains a FIFO used as an elastic buffer to prevent underflow or overrun to/of the LCD.

Hardware ignores the bottom address register of blocks for dual- and single-block access mode. Each block size is defined by

Block_Size = ES x EN x FN in bytes

where:

ES = Element Size: number of bytes within an element. Bit-field data_type is configured in register DMA LCD channel source destination parameters (DMA_LCD_CSDP).

EN = Element number (within one frame).

FN = Frame number.

In order to support graphic functionality, this DMA LCD channel also supports rotation capability at 0°, 90°, 180°, and 270°, by utilizing its source double index mode.

3.2.2 LCD Channel Addressing Modes

Post-incremented, single-indexed (element index), and double indexed (element and frame indexes) addressing modes are supported from the source port side. However, note that there is no write address to compute in the destination for the display channel. The read FIFO address is controlled by the LCD controller, so there is no write address to a port.

Source Address and Block Size Alignment

On the destination side, the accesses are fixed to 32 bits for external LCD controller and 16 bits for OMAP LCD controller. There is no address alignment because the LCD controller does not use addresses to access data from the LCD channel.

Source memory accesses can be 32/16/8-bit or 4x32-bit burst accesses. The LCD channel is compliant with the generic logical channel constraint; that is, address must be aligned on data_type.

Data block size must be a multiple of the 32-bit for external LCD controller, and a multiple of the 16-bit for OMAP LCD controller.

However, the data_type can still be 32/16/8-bit for both the OMAP and external LCD controllers.

Source Address Modes

The four different LCD channel source address algorithms use the following notations. The address mode descriptions are as follows:

A(n): Byte address of the element n within the transfer.

ES_B1: Element size in block 1 (in bytes): ES $\in$ {1, 2, 4}.

ES_B2: Element size in block 2 (in bytes): ES $\in$ {1, 2, 4}.

BS_B1: Block size of block 1 (in bytes).

BS_B1 = BB1 – TB1 = ES_B1 x EN_B1 x FN_B1

BS_B2: Block size of block 2 (in bytes).

BS_B2 = BB2 – TB2= ES_B2 x EN_B2 x FN_B2

BB1: Bottom address of block 1.

BB2: Bottom address of block 2.

TB1: Top address of block 1.

TB2: Top address of block 2.

DBM: Dual-block mode.

EN_B1: Number of elements within a frame of block 1.

EN_B2: Number of elements within a frame of block 2.

Element_counter_B1: Counter that is (re)initiated with the number of elements per frame or per transfer inside block 1. Decreased by one at each element transferred. Initial value EN_B1 is configured in register DMA channel element number, DMA_CEN.

Element_counter_B2: Counter that is (re)initiated with the number of elements per frame inside block 2. Decreased by one at each element transferred. Initial value EN_B2 is configured in register DMA channel element number, DMA_CEN.

Frame_counter_B1: Counter that is (re)initiated with the number of frames inside block 1. Decreased by one at each frame transferred. Initial value FN_B1 is configured in register DMA channel frame number, DMA_CFN.

Frame_counter_B2: Counter that is (re)initiated with the number of frames inside block 2. Decreased by one at each frame transferred. Initial value FN_B2 is configured in register DMA channel frame number, DMA_CFN.

EI_B1: Block 1 element index in bytes $-32768 \le \text{EI\_B1} \le 32767$.

EI_B2: Block 2 element index in bytes $-32768 \le \text{EI\_B1} \le 32767$.

Stride_EI_B1: Number of elements between the beginning of the current element (n) and the beginning of the next one (n+1) in block 1.

Stride_EI_B2: Number of elements between the beginning of the current element (n) and the beginning of the next one (n+1) in block 2.

Stride_FI_B1: Number of elements between the beginning of the last element of the current frame and the beginning of the first element of the next frame in block 1.

Stride_FI_B2: Number of elements between the beginning of the last element of the current frame and the beginning of the first element of the next frame in block 2.

FI_B1: block 1 frame index in bytes:

$-2147483648 < \text{FI\_B1} < 2147483647$.

FI_B2: block 2 frame index in bytes:

$-2147483648 < \text{FI\_B2} < 2147483647$.

Post-Incremented Addressing Mode

Address is post-incremented by element size (ES_B1 or ES_B2 depending on which block is active if in dual-block mode).

DBM = 0 (one block mode)

Only block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 until the end of block 1.
Then, when A(n) reaches BB1, A(n+1) = TB1 again

DBM = 1 (dual block mode)

When block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 until the end of block 1.
Then, when A(n) reaches BB1, A(n+1) = TB2: block 2 becomes active.

When block 2 is active:

A(n) = TB2.
A(n+1) = A(n) + ES_B2 until the end of block 2.
Then, when A(n) reaches BB2, A(n+1) = TB1: block 1 becomes active again.

Single-Indexed Addressing Mode

Address is incremented by element size and element index. All is expressed in bytes.

DBM = 0 (one block mode)

Only block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 + (EI_B1 – 1)
where EI_B1 = ((Stride_EI_B1 – 1) * ES_B1) + 1

A(n) is incremented in this way until the end of block 1. Then, when A(n) reaches BB1, A(n+1) = TB1 again.

Note:

Stride_EI_B1 = 1 (equivalent to EI_B1 = 1) gives consecutive element accesses, hence the same behavior as with the post-incremented addressing mode.

DBM = 1 (dual block mode)

When block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 + (EI_B1 – 1)
where EI_B1 = ((Stride_EI_B1 – 1) * ES_B1) + 1

A(n) is incremented in this way until the end of block 1. Then, when A(n) reaches BB1, A(n+1) = TB2: block 2 becomes active.

When block 2 is active:

A(n) = TB2.
A(n+1) = A(n) + ES_B2 + (EI_B2 – 1)
where EI_B2 = ((Stride_EI_B2 – 1) * ES_B2) + 1

A(n) is incremented in this way until the end of block 2. Then, when A(n) reaches BB2, A(n+1) = TB1: block 1 becomes active again.

Note:

Stride_EI_B1/2 = 1 (equivalent to EI_B1/2 = 1) give consecutive element accesses, hence the same behavior as with the Post_Incremented addressing mode.

In dual-block mode, block 2 can be active first.

Double-Indexed Addressing Mode

Address is incremented by element size and element index if the end of the current frame is not reached.

Address is incremented by element size and frame index if the end of the current frame is reached.

All is expressed in bytes.

DBM = 0 (one block mode)

Only block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 + (EI_B1 – 1)
where EI_B1 = ((Stride_EI_B1 – 1) * ES_B1) + 1

A(n) is incremented in this way until the end of the current frame, that is: as long as Element_counter_B1 ≠ 0.

When end of frame (but not end of block 1) is reached, that is:
Element_counter_B1 = 0 and Frame_counter_B1 ≠ 0:

A(n+1) = A(n) + ES_B1 + (FI_B1 – 1)
where FI_B1 = ((Stride_FI_B1 – 1) * ES_B1) + 1

When A(n) reaches BB1, A(n+1) = TB1 again

DBM = 1 (dual block mode)

When block 1 is active:

A(0) = TB1.
A(n+1) = A(n) + ES_B1 + (EI_B1 – 1)
where EI_B1 = ((Stride_EI_B1 – 1) * ES_B1) + 1

A(n) is incremented in this way until the end of the current frame, that is: as long as Element_counter_B1 ≠ 0.

When end of frame (but not end of block 1) is reached, that is:
Element_counter_B1 = 0 and Frame_counter_B1 ≠ 0:

A(n+1) = A(n) + ES_B1 + (FI_B1 – 1)
where FI_B1 = ((Stride_FI_B1 – 1) * ES_B1) + 1

When A(n) reaches BB1, A(n+1) = TB2: block 2 becomes active.

When block 2 is active:

A(0) = TB2A.
A(n+1) = A(n) + ES_B2 + (EI_B2 – 1)
where EI_B2 = ((Stride_EI_B2 – 1) * ES_B2) + 1

A(n) is incremented in this way until the end of the current frame, that is: as long as Element_counter_B2 $\neq$ 0.

When end of frame (but not end of block 2) is reached, that is:
Element_counter_B2 = 0 and Frame_counter_B2 $\neq$ 0:

A(n+1) = A(n) + ES_B2 + (FI_B2 – 1)
where FI_B2 = ((Stride_FI_B2 – 1) * ES_B2) + 1

When A(n) reaches BB2, A(n+1) = TB1: block 1 becomes active again.

Note:

Both Stride_EI_B1/2= 1 (equivalent to EI_B1/2 = 1) and Stride_FI_B1/2 = 1 (equivalent to EI_B1/2 = 1) give consecutive element accesses, hence the same behavior as with the post-incremented addressing mode.

In dual-block mode, block 2 can be active first.

3.2.3 DMA LCD Channel Sharing Feature

The LCD channel in the OMAP 3.2 system DMA supports the LCD controller. The lcd_destination_port bit in the DMA LCD control register (DMA_LCD_CTRL) sets the DMA to use the LCD controller. The OMAP LCD controller supports 16-bit single accesses. See the *OMAP5912 Multimedia Processor Display Interface Reference Guide* (SPRU764) for detailed information about the LCD controller.

LCh-D is enabled differently depending on which LCD controller used:

❑ In OMAP LCD controller mode, the LCD channel is enabled when the OMAP LCD controller is enabled; that is, when bit LCDEN = 1 in the LCD control register (LCDCONTROL). For more information, see the LCD control register description in the *OMAP5912 Multimedia Processor Display Interface Reference Guide* (SPRU764).

3.2.4 DMA LCD Channel Rotation

This DMA LCD channel supports four types of rotation: 0°, 90°, 180°, and 270°, by using the double indexed addressing mode in LCD channel source port. This feature allows the use of display screens that are not oriented as they were designed. For example, a 320x240 screen can be used in a 240x320 application.

3.2.5 DMA LCD Channel Autoinitialization Feature

In order to support special requirements for LCD display, the detailed autoinitialization feature for the LCD channel is described below.

❑ If the DMA LCD channel is connected to the OMAP external LCD controller:

  When software sets the DMA_LCD_CCR.ENABLE bit to 1, then the logical LCD channel is enabled. It loads LCD channel programming register set to its channel active register set. Then the logical LCD channel (LCh_D) is active, and data starts transferring.

❑ If the DMA LCD channel is connected to the OMAP LCD controller:

  When software enables the OMAP LCD controller, bit LCDCONTROL. LCDEN is set to 1 and the DMA LCD channel is automatically enabled at the same time. It loads LCD channel programming register set to channel active register set. The logical LCD channel is active, and data starts transferring.

At the end of the transfer, the LCD logical channel is enabled again, when auto-init is set to 1, and it loads LCD physical channel with:

❑ Channel programming set, if end_prog = 1 and repeat = 1
❑ Channel active set, if end_prog = 0 and repeat = 1

Table 30 provides a bit summary.

Table 30. *Autoinitialization Bits Summary for LCD Channel in Noncompatible Mode*

Auto_init	Repeat	end_prog	Autoinitialization Behavior
0	Don't care	Don't care	No autoinitialization. It waits until enable = 1 to enable the LCD logical channel, and loads the physical LCD channel with its programming register set. However, the LCD logical channel is active only when the LCD controller enables it.
1	0	0	As both repeat and end_prog bits equal 0, the channel is still enabled but pending. At the end of the current transfer, *the logical LCD channel waits until repeat or end_prog = 1 to reactivate itself again*. When end_prog = 1 the programming register set is copied to the active register set: *a new context is programmed.*
1	0	1	At the end of the current transfer, the logical LCD channel immediately loads the physical LCD channel with its programming register set, when physical LCD channel is granted (end_prog = 1 allows loading the new context, disregarding the repeat bit). *The channel reinitializes itself and starts a new transfer with the new context*.
1	1	Don't care	The channel reinitializes itself at the end of the current transfer *and starts a new transfer with the previous context* (active register set).

3.2.6 DMA_LCD_Disable/Bus Error Feature

Software can disable the LCD channel on the fly. If the LCD channel is disabled, then transfer stops immediately.

During LCD channel transfer, if an underflow occurs, the DMA LCD channel resets its enable bit and the LCD channel stops immediately.

LCD PCh underflow is possible when the destination is the OMAP LCD controller (reads from external controller are stalled until data is ready).

If underflow occurs, a signal is sent to the OMAP LCD controller, and the FUF bit is set in the OMAP LCD controller status register (LCSR). An interrupt is generated when a LCSR bit is set. See the *OMAP5912 Multimedia Processor Display Interface Reference Guide* (SPRU764).

If one hardware request is currently being serviced, and other hardware requests are triggered, then the DMA_LCD controller will stop and signal an event drop interrupt on the second new hardware request (bus error).

3.2.7 LCD Channel Usage Restrictions

Exclusive Blocks

The hardware design does not detect any overlap between two block buffers; that is, the start and stop addresses of each buffer must represent two different physical parts into the memory. In dual-block mode, the top address of the second block must be greater (and not equal to) than the bottom address of the first block.

Both Blocks Must Belong to a Single Source

In case of dual-block mode operation, it is not possible to have one block read from one source and one block read from a second source. *To change from a source to another, the LCD_LCD_EN (enable transfer) signal must not be asserted and all pending LCD interrupts must be processed.*

LCD Registers Can Be Configured During a Transfer

There is a shadow register set for the LCD channel, which allows the software to change the LCD channel context on the fly after the LCD channel enabled. This is not supported in the OMAP 3.0/3.1 system DMA.

Example 3 shows a transfer from a video block, located in SDRAM, to the OMAP LCD controller.

Example 3. LCD Transfer: EMIFF (SDRAM) → LCD, One Block

The size for the LCD display is 6 x 16 pixels with 16 bits per pixels. So the length of the video frame is 6x16x2 (in bytes) + 32 bytes for the palette = 224 bytes. If the video block starts at address 0x0B0000, the bottom address of the video block is 0x0B00DE.

Step 1: Registers are set as:

DMA_LCD_CTRL

BLOCK_MODE = 0 (one block)
BLOCK_IT_IE = 1
BUS_ERROR_IT_IE = 1
LCD_SOURCE_PORT = 0 (SDRAM)

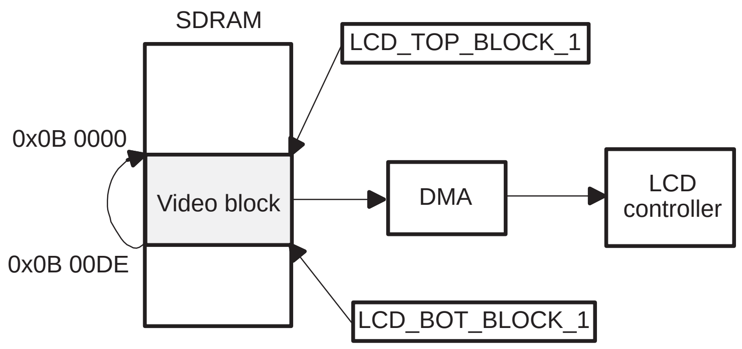
DMA_LCD_TOP_B1_U = 0x000B
DMA_LCD_TOP_B1_L = 0x0000
DMA_LCD_BOT_B1_U = 0x000B
DMA_LCD_BOT_B1_L = 0x00DE

DMA_LCD_TOP_B2_U = irrelevant
DMA_LCD_TOP_B2_L = irrelevant
DMA_LCD_BOT_B2_U = irrelevant
DMA_LCD_BOT_B2_L = irrelevant

Step 2: The transfer starts when the enable (hardware) signal from the OMAP LCD controller is asserted high.

The transfer runs, and an interrupt is generated at the end of the block.

Figure 12. LCD One-Block Mode Transfer Scheme



Step 3: When an interrupt occurs, read the DMA_LCD_CTRL register, to know the source of the interrupt.

If DMA_LCD_CTRL[3] = 1 (that is to say, block_1_it_cond = 1), end of block 1 interrupt is detected.

If end of block is reached, the DMA restarts at the top of the block.

Step 4: Reset DMA_LCD_CTRL [3] and wait for another interrupt.

Example 4. LCD Transfer: OCP_T1 (Test RAM) → LCD, Two Blocks

Example 4 shows a transfer from two video blocks located in memory connected to the OCP_T1 port to the LCD controller.

The size for the LCD display is 6 x 16 pixels with 16 bits per pixels. So the length of one video block is 6 x 16 x 2 (in bytes) + 32 bytes for the palette = 224 bytes. If the video block 1 starts at address 0x0B0000, the bottom address of the video block is 0x0B00DE. If the video block 2 starts at address 0x0C0000, the bottom address of the video block is 0x0C00DE.

Step 1: Registers are set as:

DMA_LCD_CTRL

BLOCK_MODE = 1 (two blocks)
BLOCK_IT_IE = 1
BUS_ERROR_IT_IE = 1
LCD_SOURCE_PORT = 1 (IMIF)

DMA_LCD_TOP_B1_U = 0x000B
DMA_LCD_TOP_B1_L = 0x0000
DMA_LCD_BOT_B1_U = 0x000B
DMA_LCD_BOT_B1_L = 0x00DE

DMA_LCD_TOP_B2_U = 0x000C
DMA_LCD_TOP_B2_L = 0x00000
DMA_LCD_BOT_B2_U = 0x000C
DMA_LCD_BOT_B2_L = 0x00DE

Step 2: The transfer starts when the enable (hardware) signal from the OMAP LCD controller is asserted high.

The transfer runs, and the interrupts are generated at the end of block 1 and 2, respectively.

DMA_LCD_CTRL [3] = block_1_it_cond = 1. This bit is a status bit, which detects the interrupt.

When end of block 1 is reached, the DMA restarts at the top address of block 2 and DMA_LCD_CTRL [3] is reset to be able to detect a next interrupt.

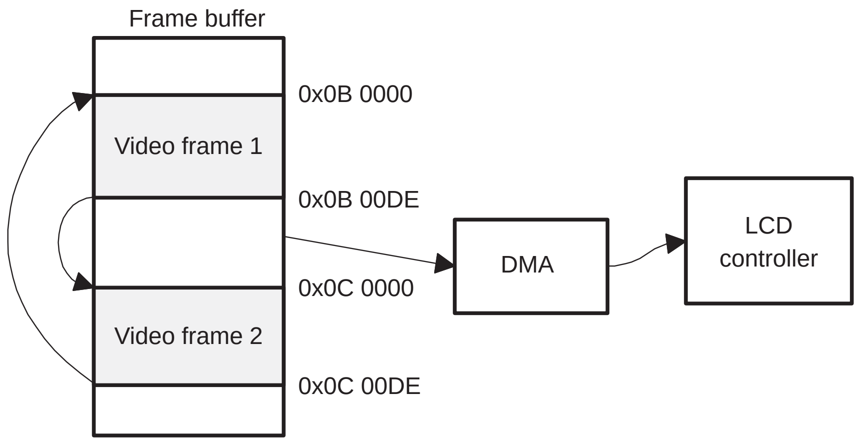


Figure 13. LCD Dual-Block Mode Transfer Scheme

DMA_LCD_CTRL [4] = BLOCK_2_IT_COND = 1: end of block 2 is reached. The DMA restarts at the top address of block 1, and DMA_LCD_CTRL [4] is reset to be able to detect a next interrupt.

Switching between the two frames is automatic; it does not need any reconfiguration of the channel.

3.2.8 LCD Channel OMAP 3.0/3.1 Compatible Mode Programming

Users have the option of having new LCD channel features and maintaining compatibility with the previous LCD channel. To keep compatible with the OMAP3.1 programming mode, care must be taken on the following points. Everywhere else, there is no compatibility obstacle.

Configuration Registers

The DMA_LCD_CTRL register is one of the LCD configuration registers. This register manages the operation of dual or single block mode, the interrupt enable bits, and the source port for the next transfer. Then it returns information by setting the status bits in its register. An interrupt can be sent at the end of the transfer of each block; this interrupt line is connected to the LCD interrupt line of the DMA.

Addressing Mode

In OMAP 3.1 compatible mode, the LCD channel only supports the *post-increment* addressing mode on the source side.

Address is post-incremented by element size (ES_B1 or ES_B2 depending on which block is active if in dual-block mode)

A(n+1) = A(n) + ES_B1/2 if TB1/2 ≤ A(n+1) ≤ BB1/2

where:

A(n) is the byte address of element n within the transfer.

ES_B1/2 is block1/2 element size.

TB1/2 is top address for block1/2 and BB1/2 is bottom address for block1/2.

See post-increment details in section 3.2.2.

Hardware rotation is not supported in compatible mode due to restrictions on addressing modes.

DMA LCD Channel Sharing Feature

A second external LCD controller is not supported in compatible mode. Only the OMAP3.2 embedded LCD controller is available.

Disabling Feature

The LCD channel cannot be disabled by software. However, it stops transferring as soon as the OMAP LCD controller disables the LCD channel. Then it restarts from the beginning again, after the LCD controller enables the LCD channel.

LCD Channel Restriction

In OMAP 3.1 compatible mode, it is not possible to change any bit of any LCD channel register until a transfer has been fully completed, because no shadow registers exist in LCD channel as in regular channels. To update registers, the LCD_Controller must not be enabled and all pending LCD interrupts must be processed.

The LCD channel source can only have 32-bit access from L3_OCP_T1 (Test RAM) or EMIFF (SDRAM).

Selecting the lcd_source_port bit-field (in the DMA_LCD_CTRL register) to binary 10 or 11 causes undefined effects.

DMA LCD Channel Autoinitialization Feature

At the end of transfer, the LCD logical channel is enabled again if LCD_LCD_EN is set to 1 in the OMAP LCD controller. Table 31 summarizes the autoinitialization behavior in compatible mode.

Table 31. Autoinitialization Configuration Bits Summary for LCD Channel in Compatible Mode

Auto_init	Repeat	End_prog	Autoinitialization Behavior
Hardwired 1	Hardwired 1	Hardwired 1	At the end of the current transfer, the logical LCD channel immediately loads the physical LCD channel with its programming register set, when physical LCD channel is granted (end_prog = 1 allows loading the new context, disregarding the repeat bit). *The channel reinitializes itself and starts a new transfer with the new context.* However, the LCD logical channel is active only when the LCD controller enables it.
0 (Software)	X (Software)	X (Software)	Not supported
1 (Software)	X (Software)	X (Software)	Not supported

DMA Configuration Registers I/O Space

Each channel has its own address space. The configuration address is split into several fields. Each field is decoded to generate a channel select and a register select within the channel.

More configuration registers were added to the LCD channel OMAP 3.2. This results in two different LCD channel register mappings which are dependent on the mode selected, which is controlled by bit OMAP3_1_MAPPING_DISABLE in register DMA_GSCR.

Table 32 shows the register mapping in the two different modes (OMAP 3.0/3.1 compatible or not).

Table 32. LCD Channel Register Mapping for OMAP 3.2 Respectively OMAP 3.0/3.1 Compatible Modes

Offset	Register Mapping in OMAP 3.2 Mapping Mode	Register Mapping in OMAP 3.0/3.1 Mapping Mode
E3C0	DMA_LCD_CSDP	DMA_LCD_CTRL
E3C2	DMA_LCD_CCR	DMA_LCD_TOP_B1_L
E3C4	DMA_LCD_CTRL	DMA_LCD_TOP_B1_U
E3C6	Reserved	DMA_LCD_BOT_B1_L

Table 32. LCD Channel Register Mapping for OMAP 3.2 Respectively OMAP 3.0/3.1 Compatible Modes (Continued)

Offset	Register Mapping in OMAP 3.2 Mapping Mode	Register Mapping in OMAP 3.0/3.1 Mapping Mode
E3C8	DMA_LCD_TOP_B1_L	DMA_LCD_BOT_B1_U
E3CA	DMA_LCD_TOP_B1_U	DMA_LCD_TOP_B2_L
E3CC	DMA_LCD_BOT_B1_L	DMA_LCD_TOP_B2_U
E3CE	DMA_LCD_BOT_B1_U	DMA_LCD_BOT_B2_L
E3D0	DMA_LCD_TOP_B2_L	DMA_LCD_BOT_B2_U
E3D2	DMA_LCD_TOP_B2_U	N/A
E3D4	DMA_LCD_BOT_B2_L	N/A
E3D6	DMA_LCD_BOT_B2_U	N/A
E3D8	DMA_LCD_SRC_EI_B1	N/A
E3DA	DMA_LCD_SRC_FI_B1_L	N/A
E3DC	DMA_LCD_SRC_EI_B2	N/A
E3DE	DMA_LCD_SRC_FI_B2_L	N/A
E3E0	DMA_LCD_SRC_EN_B1	N/A
E3E2	DMA_LCD_SRC_EN_B2	N/A
E3E4	DMA_LCD_SRC_FN_B1	N/A
E3E6	DMA_LCD_SRC_FN_B2	N/A
E3EA	DMA_LCH_CTRL	N/A
E3F4	DMA_LCD_SRC_FI_B1_U	N/A
E3F6	DMA_LCD_SRC_FI_B2_U	N/A

3.3 System DMA Registers

❑ All reserved bits and all reserved registers must be written to as 0x0000.

❑ All reserved bits are read as 0.

❑ The configuration and status registers are part of a superset that is designed for a 16-bit port and a 16-bit channels DMA. This superset enables optimal design reuse in hardware and software, and this design reuse capability requires generic register mapping. These requirements cause some registers to seem at times almost empty.

3.3.1 Summary DMA Global Registers

Table 33 lists the DMA global control registers. Table 34 through Table 50 describe the register bits. You must configure global registers before any LCh is enabled, to avoid undefined effects.

Table 33. DMA Global Control Registers

Base Address = 0xFFFE DC00			
Name	**Description**	**R/W**	**Offset**
DMA_GCR	DMA global control	R/W	0x00
DMA_GSCR	DMA software compatible	R/W	0x04
DMA_GRST	Software reset control	R/W	0x08
DMA_HW_ID	DMA version ID	R/W	0x42
DMA_PCh2_ID	Physical channel 2 version ID	R/W	0x44
DMA_PCh0_ID	Physical channel 0 version ID	R/W	0x46
DMA_PCh1_ID	Physical channel 1 version ID	R/W	0x48
DMA_PChG_ID	Physical channel G version ID	R/W	0x4A
DMA_PChD_ID	Physical channel D version ID	R/W	0x4C
DMA_CAPS_0_U	Global DMA capability 0 upper	R/W	0x4E
DMA_CAPS_0_L	Global DMA capability 0 lower	R/W	0x50
DMA_CAPS_1_U	Global DMA capability 1 upper	R/W	0x52
DMA_CAPS_1_L	Global DMA capability 1 lower	R/W	0x54
DMA_CAPS_2	Global DMA capability 2	R/W	0x56
DMA_CAPS_3	Global DMA capability 3	R/W	0x58
DMA_CAPS_4	Global DMA capability 4	R/W	0x5A
DMA_PCh2_SR	Physical channel 2 status	R/W	0x60
DMA_PCh0_SR	Physical channel 0 status	R/W	0x80
DMA_PCh1_SR	Physical channel 1 status	R/W	0x82
DMA_PChD_SR_0	Physical channel D status	R/W	0xC0

Table 34. DMA Global Control Register (DMA_GCR)

Base Address = 0xFFFE DC00, Offset = 0x00

Bit	Name	Function	R/W	Reset
15:5	RESERVED	Reserved	R/W	ND
4	ROUND_ROBIN_DISABLE	DMA physical channel scheduler round robin scheduling disable: 0: DMA physical channel scheduler uses round robin scheduling scheme to schedule next available logical channel. 1: DMA physical channel scheduler uses fixed weighted scheduling scheme (from LCH_0 to LCH_i) to schedule next available logical channel.	R/W	0
3	CLK_AUTOGATING_ON	DMA Clock autogating on: 0 = DMA clocks are always on. 1 = DMA cuts off its clocks, according to its activity. Clock_Autogating_on must always be enabled but can be disabled for silicon debug reasons.	R/W	1
2	FREE	DMA reaction to the suspend signal: 0 = DMA suspends all the current transfers when it receives the suspend signal from the processor. Transfers resume when the MPU releases the suspend signal. 1 = DMA continues running when it receives the suspend signal from the processor (For example: when the processor is halted for debug by a breakpoint).	R/W	0
1:0	RESERVED	Reserved	R/W	0

Table 35. DMA Software Compatible Register (DMA_GSCR)

Base Address = 0xFFFE DC00, Offset = 0x04				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	RESERVED	Reserved	R/W	ND
3	OMAP3_1_MAPPING DISABLE	OMAP3.1 mapping disable 0 = DMA compatible with OMAP 3.0/3.1 system DMA interrupt line mapping and logical channel configuration register address mapping (system DMA I/O space). 1 = DMA compatible with OMAP_3.2 DMA. Maps interrupt lines and logical channel configuration register address mapping as defined for OMAP 3.2. Compatibility is also dependent on the OMAP3_1_Compatible_Disable bit in register DMA_LCH_CCR.	R/W	0
2:0	RESERVED	Reserved	R/W	ND

Table 36. DMA Software Reset Control Register (DMA_GRST)

Base Address = 0xFFFE DC00, Offset = 0x08				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1	RESERVED	Reserved	R/W	ND
0	SW_RESET	DMA software reset control: 0: SW will always read this bit as 0. 1: Resets the whole DMA when software writes 1 to this bit (automatically clears to 0).	R/W	0

Table 37. DMA Hardware Version ID Register (DMA_HW_ID)

Base Address = 0xFFFE DC00, Offset = 0x42				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		DMA version ID number	R	0x0001

Table 38. PCh-2 Version ID Register (DMA_PCh2_ID)

Base Address = 0xFFFE DC00, Offset = 0x44				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	PCH2_ID	DMA PCh-2 version ID number	R	0x0001

Table 39. PCh-0 Version ID Register (DMA_PCh0_ID)

Base Address = 0xFFFE DC00, Offset = 0x46				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	PCH0_ID	DMA PCh-0 version ID number	R	0x0001

Table 40. PCh-1 Version ID Register (DMA_PCh1_ID)

Base Address = 0xFFFE DC00, Offset = 0x48				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	PCH1_ID	DMA PCh-1 version ID number	R	0x0001

Table 41. PCh-G Version ID Register (DMA_PChG_ID)

Base Address = 0xFFFE DC00, Offset = 0x4A				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	PCHG_ID	DMA PCh-G version ID number	R	0x0001

Table 42. PCh-D Version ID Register (DMA_PChD_ID)

Base Address = 0xFFFE DC00, Offset = 0x4C				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	PCHD_ID	DMA PCh-D version ID number	R	0x0001

Table 43. Global DMA Capability U Register 0 (DMA_ CAPS_0_U)

Base Address = 0xFFFE DC00, Offset = 0x4E				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	RESERVED	Reserved	R	ND
3	CFC	Constant fill capacity: 0: PCh-G/PCh-0, or -1 cannot do constant fill copy. 1: PCh-G/PCh-0, or -1 can do constant fill copy.	R	1
2	TBLTC	Transparent BLT capability: 0: PCh-G/PCh-0, or -1 cannot do transparent BLT copy. 1: PCh-G/PCh-0, or -1 can do transparent BLT copy.	R	1

Table 43. Global DMA Capability U Register 0 (DMA_ CAPS_0_U) (Continued)

Base Address = 0xFFFE DC00, Offset = 0x4E				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	OD	Overlap detection capability (not available in OMAP3.2): 0: PCh-G cannot do overlap detection. 1: PCh-G can do overlap detection.	R	0
0	DBLTC	Directional BLT capability (not available in OMAP3.2): 0: PCh-G cannot do directional BLT copy. 1: PCh-G can do directional BLT copy.	R	0

Table 44. Global DMA Capability L Register 0 (DMA_ CAPS_0_L)

Base Address = 0xFFFE DC00, Offset = 0x50				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:3	RESERVED	Reserved	R	ND
2		Sub-byte destination capability (not available in OMAP 3.2); 0: PCh-G cannot do bub-byte adjust for expansion. 1: PCh-G can do bub-byte adjust for expansion.	R	0
1	RESERVED	Reserved	R	ND
0		Origin coordinate capability (not available in OMAP 3.2): 0: PCh-G cannot do origin coordinate calculation. 1: PCh-G can do origin coordinate calculation.	R	0

Table 45. Global DMA Capability U Register 1 (DMA_CAPS_1_U)

Base Address = 0xFFFE DC00, Offset = 0x52				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	RESERVED	Reserved	R	ND

Table 46. Global DMA Capability L Register 1 (DMA_CAPS_1_L)

Base Address = 0xFFFE DC00, Offset = 0x54				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:2	RESERVED	Reserved	R	ND
1	CE	1-bit palletized capability (not available in OMAP 3.2): 0: PCh-G cannot do 1-bit color expansion. 1: PCh-G can do 1-bit color expansion.		
0	RESERVED	Reserved	R	ND

Table 47. Global DMA Capability Register 2 (DMA_CAPS_2)

Base Address = 0xFFFE DC00, Offset = 0x56				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:9	RESERVED	Reserved	R	ND
8	SSDIC	Separate source/double-index capability: 0: Does not support separate src/dst index for 2-D addressing. 1: Supports separate src/dst index for 2-D addressing.	R	1
7	DDIAC	Destination double-index address capability: 0: Does not support double-index address mode in destination port. 1: Supports double-index address mode in destination port.	R	1
6	DSIAC	Destination single-index address capability: 0: Does not support single-index address mode in destination port. 1: Supports single-index address mode in destination port.	R	1
5	DPIAC	Destination post-increment address capability: 0: Does not support post-increment address mode in destination port. 1: Supports post-increment address mode in destination port.	R	1
4	DCAC	Destination constant address capability: 0: Does not support constant address mode in destination port. 1: Supports constant address mode in destination port.	R	1
3	SDIAC	Source double-index address capability: 0: Does not support double-index address mode in source port. 1: Supports double-index address mode in source port.	R	1

Table 47. Global DMA Capability Register 2 (DMA_CAPS_2) (Continued)

Base Address = 0xFFFE DC00, Offset = 0x56				
Bit	**Name**	**Function**	**R/W**	**Reset**
2	SSIAC	Source single-index address capability: 0: Does not support single-index address mode in source port. 1: Supports single-index address mode in source port.	R	1
1	SPIAC	Source post-increment address capability: 0: Does not support post-increment address mode in source port. 1: Supports post-increment address mode in source port.	R	1
0	SCAC	Source constant address capability: 0: Does not support constant address mode in source port. 1: Supports constant address mode in source port.	R	1

Table 48. Global DMA Capability Register 3 (DMA_CAPS_3)

Base Address = 0xFFFE DC00, Offset = 0x58				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:6	RESERVED	Reserved	R	ND
5	CCC	Channel chaining capability: 0: Does not support logical channel chaining capability. 1: Supports logical channel chaining capability.	R	1
4	IC	LCh interleave capability: 0: Does not support logical channel interleave capability. 1: Supports logical channel interleave capability.	R	1
3	ARC	Autoinit and repeat capability: 0: Does not support repeat feature in autoinitialization mode. 1: Supports repeat feature in autoinitialization mode (supported only in OMAP 3.0/1 compatible mode).	R	1
2	AEC	Autoinit and End_prog capability: 0: Does not support End_Prog feature in autoinit mode. 1: Supports End_Prog feature in autoinit mode (supported only in OMAP 3.0/1 compatible mode).	R	1

Table 48. Global DMA Capability Register 3 (DMA_CAPS_3) (Continued)

Base Address = 0xFFFE DC00, Offset = 0x58				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	FSC	Frame synchronization capability: 0: Does not support synchronization transfer on frame boundary. 1: Supports synchronization transfer on frame boundary.	R	1
0	ESC	Element synchronization capability: 0: Does not support synchronization transfer on element boundary. 1: Supports synchronization transfer on element boundary.	R	1

Table 49. Global DMA Capability Register 4 (DMA_CAPS_4)

Base Address = 0xFFFE DC00, Offset = 0x5A				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:7	RESERVED	Reserved	R	ND
6	SSC	Sync status capability: 0: Does not support synchronization transfer status bit generation. 1: Supports synchronization transfer status bit generation.	R	1
5	BIC	Block interrupt capability (end of block): 0: Does not support block interrupt generation capability. 1: Supports block interrupt generation capability.	R	1
4	LFIC	Last frame interrupt capability (start of last frame): 0: Does not support last frame interrupt generation capability. 1: Supports last frame interrupt generation capability.	R	1
3	FIC	Frame interrupt capability (end of frame): 0: Does not support frame interrupt generation capability. 1: Supports frame interrupt generation capability.	R	1
2	HFIC	Half frame interrupt capability (half of frame): 0: Does not support half frame interrupt generation capability. 1: Supports half frame interrupt generation capability.	R	1

Table 49. Global DMA Capability Register 4 (DMA_CAPS_4) (Continued)

Base Address = 0xFFFE DC00, Offset = 0x5A				
Bit	**Name**	**Function**	**R/W**	**Reset**
1	EDIC	Event drop interrupt capability (request collision): 0: Does not support event drop interrupt generation capability. 1: Supports event drop interrupt generation capability.	R	1
0	TOIC	Timeout interrupt capability (timeout error): 0: Does not support timeout interrupt generation capability. 1: Supports timeout interrupt generation capability.	R	1

Table 50. Physical Channel-x Status Registers (DMA_PCh2_SR,..., DMA_PHhD_SR_0)

Base Address = 0xFFFE DC00, Offset = 0x60, 0x80. 0x82, 0xC0				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:8	RESERVED	Reserved	R	ND
7:0	ALCN	Active logical channel number for associated physical channel.	R	0xFF

Table 33 specifies which physical channels match each offset. These return the logical channel ID, which is active in the associated physical channel. These four registers can be used to monitor the progress of a DMA physical channel transfer.

Each register is a snapshot of the current logical channel number, which is active on the physical channel:

FF: DMA Physical_Channel_x is IDLE.

Others: DMA Logical_Channel_Number is active on the physical channel.

If PchD is active, DMA_PChD_SR_0 will read as 0x1F if DMA is in OMAP3.2 compatible mode and 0xC in OMAP3.0/3.1 compatible mode.

3.3.2 Logical Channel Registers

This set of registers is instantiated within each logical channel to the DMA. The registers for the display channel, LCh-D are specific for that channel. Hence, these registers are collected in a dedicated section for the LCh-D, see Section 3.3.3.

All registers for a specific LCh must be configured before the LCh is enabled; if not, it will cause undefined effects. There are some exceptions to this rule when in OMAP3.0/3.1 compatible mode.

Table 51 lists the logical channel configuration registers. Table 52 through Table 96 describe the register bits.

Table 51. DMA Logical Channel Configuration Registers

Base Address = FFFE D800			
Name	**Description**	**R/W**	**Offset** †
DMA_CSDP	Channel source destination parameters	R/W	0x00 + (n*0x40)
DMA_CCR	Channel control register	R/W	0x02 + (n*0x40)
DMA_CICR	Channel interrupt control register	R/W	0x04 + (n*0x40)
DMA_CSR	Channel status register	R	0x06 + (n*0x40)
DMA_CSSA_L	Channel source start addr, lower bits	R/W	0x08 + (n*0x40)
DMA_CSSA_U	Channel source start addr, upper bits	R/W	0x0A + (n*0x40)
DMA_CDSA_L	Channel destination start addr, lower bits	R/W	0x0C + (n*0x40)
DMA_CDSA_U	Channel destination start addr, upper bits	R/W	0x0E + (n*0x40)
DMA_CEN	Channel element number	R/W	0x10 + (n*0x40)
DMA_CFN	Channel frame number	R/W	0x12 + (n*0x40)
DMA_CSFI	Channel source frame index	R/W	0x14 + (n*0x40)
DMA_CSEI	Channel source element index	R/W	0x16 + (n*0x40)
DMA_CSAC	Channel source addr counter	R	0x18 + (n*0x40)
DMA_CDAC	Channel destination addr counter	R	0x1A + (n*0x40)
DMA_CDEI	Channel destination element index	R/W	0x1C + (n*0x40)
DMA_CDFI	Channel destination frame index	R/W	0x1E + (n*0x40)
DMA_COLOR_L	Color parameter register, lower bits	R/W	0x20 + (n*0x40)
DMA_COLOR_U	Color parameter register, upper bits	R/W	0x22 + (n*0x40)
DMA_CCR2	Channel control register 2	R/W	0x24 + (n*0x40)
DMA_CLNK_CTRL	Channel link control register	R/W	0x28 + (n*0x40)
DMA_LCH_CTRL	Logical channel control register	R/W	0x2A + (n*0x40)

† n is the logical channel numbered 0x0 through 0xF.

Table 52. Channel Source Destination Parameters Register (DMA_CSDP)

Base Address = 0xFFFE D800, Offset Address = 0x00 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:14	DST_BURST_EN	Destination burst enable	R/W	00
13	DST_PACK	Destination packing	R/W	0
12:9	DST	Transfer destination	R/W	0000
8:7	SRC_BURST_EN	Source burst enable	R/W	00
6	SRC_PACK	Source packing	R/W	0
5:2	SRC	Transfer source	R/W	0000
1:0	DATA_TYPE	Defines the type of the data moved in the channel (see Table 53)	R/W	00

Table 53. Data Types

Data_type	Type
00	s8: 8 bits scalar
01	s16: 16 bits scalar
10	s32: 32 bits scalar
11	Illegal (cause undefined effects)

Start_Address must be aligned on the boundary of the type of data moved. For example, if data_type is s32, the source and destination start address must be aligned on a word32. If data_type is s8, source and destination start_address can have any value. Otherwise, the DMA is forced to transfer a different type of data to the PORT because of the unaligned start address value, which can cause undefined effects.

It is the software's responsibility to make sure that start address is aligned with channel data type.

- src: Transfer source

  A unique identifier is given to each port. This field identifies the port originator of the transfer.

src	OMAP Source Port
0000	EMIFF
0001	EMIFS
0010	OCP_T1
0011	TIPB
0100	OCP_T2
0101	MPUI
Others	Illegal (cause undefined effects)

- SRC_PACK: source packing

  The DMA ports can have a data bus width different from the type of data moved by the DMA channel. For example, we can read s8 data_type on a 32-bit DMA port. The DMA channel has the capacity to pack four consecutive s8 data reads in a single 32-bit read access, to increase transfer bandwidth. The packing is made in respect of endianism.

  SRC PACK = 1: the source port makes packed accesses;
  SRC PACK = 0: the source port never makes packed accesses.

- SRC_BURST_EN: source burst enable

  Used to enable bursting on the source port. When bursting is enabled, the source port performs bursts 4 x src_width access. When bursting is disabled, the source port performs single accesses of SRC_WIDTH access.

SRC_BURST_EN	Burst Type
00	Single access
01	Single access
10	Burst 4x Src_width (Only support 4x32bit)
11	Burst 8x Src_width (Not supported)

If the source port of the channel has no burst access capability, this field is ignored.

- destination fields: (dst, dst_pack, dst_burst_en) contains the same parameters as the source fields, but for the destination port

Table 54. DMA Channel Control Register (DMA_CCR)

Base Address = 0xFFFE D800, Offset Address = 0x02 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:14	DST_AMODE	Destination addressing mode	R/W	00
13:12	SRC_AMODE	Source addressing mode	R/W	00
11	END_PROG	End of programming status bit	R/W	0
10	OMAP_3_1_COMPATIBLE_DISABLE	OMAP 3.0/3.1 channel compatibility control	R/W	0
9	REPEAT	Repetitive operation	R/W	0
8	AUTO_INIT	Autoinitialization at the end of the transfer	R/W	0
7	ENABLE	Logical channel enable	R/W	0
6	PRIO	Channel priority	R/W	0
5	FS	Frame synchronization	R/W	0
4:0	SYNC	Synchronization control	R/W	00000

- SYNC: Synchronization control

  This field is used to specify the external DMA request, which can trigger the transfer for the LCh. There are 31 possible choices for the system DMA. Each LCh can be triggered by one of the DMA request inputs. A hardware DMA request cannot be shared between several concurrent channels (enabled and active). However, a hardware DMA request can be shared between different channels if they are part of a chain. Therefore, the user must carefully generate DMA requests if a sharing strategy is chosen. Otherwise, event drop of channel transfer may occur when the next DMA request is issued to trigger another channel transfer while the current channel transfer is still going on. For nonsynchronized LCh transfers, this field must be set to binary 00000.

- FS: Frame synchronization

  This bit and the bs bit in the DMA_CCR2 register are used to program the way that a DMA_request is serviced in a synchronized transfer.

  fs = 1 and bs = 0: An entire frame is transferred each time a DMA_request is made. This frame can be interleaved on the DMA ports with other channel requests.

fs = 0 and bs = 1: An entire block is transferred each time a DMA_request is made. This block can be interleaved on the DMA port with other channel requests.

fs = 0 and bs = 0: An element is transferred each time a DMA_request is made. This element can be interleaved on the DMA port with other channel requests.

fs = 1 and bs = 1: This is not allowed and causes undefined results.

❑ PRIO: Channel priority

Prio = 1: The logical channel has the highest priority level.

Prio = 0: The logical channel has the lowest priority level.

❑ ENABLE: Logical channel enable

This bit is used to enable/disable the transfer in the DMA channel.

Enable = 1: The transfer starts.
Enable = 0: The transfer stops, and it is reset.

This bit is automatically cleared by the DMA hardware once the transfer is accomplished. Clearing of this bit by the DMA has the priority over write by the processor. If both occur simultaneously, the configuration write is discarded.

❑ AUTO_INIT: Autoinitialization at the end of the transfer

Only supported in OMAP 3.0/1 compatible mode. In OMAP 3.2 mode, when bit OMAP3_1_COMPATIBLE_DISABLE = 1, this bit must always be set to 0.

Auto_init = 1: DMA is in autoinitialization mode. Once the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer if repeat or end_prog bits is set.

Auto_init = 0: DMA is in non-autoinitialization mode. The DMA channel stops at the end of the current transfer.

There are two ways to stop a channel while it is in auto-init mode:

- Write 0 in DMA_CCR.ENABLE bit: the channel immediately stops.
- Write 0 in DMA_CCR.AUTO-INIT: the channel completes the current transfer and stops.

- ❑ REPEAT: Repetitive operation

  Only supported in OMAP 3.0/3.1 compatible mode. In OMAP 3.2 mode, when bit OMAP3_1_Compatible_Disable = 1, this bit must always be set to 0. The functionality of repeat and end_prog are dependent on each other. See Table 27 for more details (following descriptions do not cover all possibilities).

  Repeat = 1: When DMA channel is in autoinitialization mode and end_prog = 0, once the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer.

  Repeat = 0: When DMA channel is in autoinitialization mode and end_prog = 0, once the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer.

  Note that even if REPEAT bit is only supported in OMAP3.0/3.1 compatible mode, its behavior with respect to END_PROG bit is slightly different than in OMAP3.1.

- ❑ OMAP3_1_COMPATIBLE_DISABLE: OMAP 3.0/3.1 channel compatibility control

  This bit is used to set the DMA logical channel programming model to be used. By default, all DMA logical channels are in omap_3.1 compatible mode**.**

  omap3_1_compatible_disable = 0: The logical channel is in OMAP_3.0/3.1 compatible mode.

  omap3_1_compatible_disable = 1: The logical channel is in OMAP_3.2 compatible mode.

- ❑ END_PROG: End of programming status bit

  Only supported in OMAP 3.0/1 compatible mode. In OMAP 3.2 mode, when bit OMAP3_1_COMPATIBLE_DISABLE = 1, this bit must always be set to 0.

  End_prog = 1: When DMA channel is in autoinitialization mode, it allows the channel to reinitialize itself after the current DMA channel transfer has been finished.

  End_prog = 0: When DMA channel is in autoinitialization mode and REPEAT bit is 0, DMA delays the channel autoinitialization until end_prog bit is set to 1 after the current DMA channel transfer has been finished.

  Note that even if END_PROG bit is only supported in OMAP3.0/3.1 compatible mode, its behavior with respect to REPEAT bit is slightly different than in OMAP3.1.

- SRC_AMODE: Source addressing mode

  This field is used to choose the addressing mode on the source port of a channel.

  src_amode = 00: Constant address
  src_amode = 01: Post-incremented address
  src_amode = 10: Single index (element index)
  src_amode = 11: Double index (element index and frame index)

- DST_AMODE: Destination addressing mode

  This field is used to choose the addressing mode on the destination port of a channel.

  dst_amode = 00: Constant address
  dst_amode = 01: Post-incremented address
  dst_amode = 10: Single index (element index)
  dst_amode = 11: Double index (element index and frame index)

Table 55. DMA Channel Interrupt Control Register(DMA_CICR)

Base Address = 0xFFFE D800, Offset Address = 0x04 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:6	RESERVED	Reserved	R/W	N/A
5	BLOCK_IE	End block interrupt enable (end of block)	R/W	0
4	LAST_IE	Last frame interrupt enable (start of last frame)	R/W	0
3	FRAME_IE	Frame interrupt enable (end of frame)	R/W	0
2	HALF_IE	Half-frame interrupt enable (half of frame)	R/W	0
1	DROP_IE	Synchronization event drop interrupt enable (request collision)	R/W	1
0	TOUT_IE	Timeout interrupt enable (timeout error)	R/W	1

The interrupt enable bits are used to choose the events that cause the DMA channel send an interrupt to the processor. There are two classes of events:

- Error event: Errors during the transfer; for example timeout and event drop.

- Status event : DMA transfer status, during DMA channel transfers; for example new frame starts, end of data block to transfer is reached.

Each time an event occurs, if the corresponding interrupt enable bit is set, the channel sends an interrupt to the processor. At the same time, the corresponding status bit is set in DMA_CSR (DMA channel status register). A status bit is not set, if the corresponding interrupt enable bit in the DMA_CICR equals 0.

- TOUT_IE: Timeout interrupt enable (timeout error)

  tout_ie = 1: The DMA sends an interrupt to the processor if a timeout error occurs either in the source or in the destination port of the channel.

  tout_ie = 0: The DMA does not send an interrupt to the processor if a timeout error occurs.

- DROP_IE: Synchronization event drop interrupt enable (request collision)

  drop_ie = 1: The channel sends an interrupt to the processor if the channel transfer is synchronized on DMA requests and two successive DMA requests drop. This occurs when a new DMA request is made while the service of the previous one has not been finished yet.

  drop_ie = 0: The channel does not interrupt the processor when a synchronization event drop occurs.

- HALF_IE: Half-frame interrupt enable (half of frame)

  half_ie = 1: The channel sends an interrupt to the processor when the transfer of the first half of the current frame completes.

  half_ie = 0: The channel does not interrupt the processor when the transfer of the first half of the current frame completes.

- FRAME_IE: Frame interrupt enable (end of frame)

  frame_ie = 1: The channel sends an interrupt to the processor when the transfer of the current frame completes.

  frame_ie = 0: The channel does not interrupt the processor when the transfer of the current frame completes.

- LAST_IE: Last frame interrupt enable (start of last frame)

  last_ie = 1: The channel sends an interrupt to the processor when the transfer of the last frame starts.

  last_ie = 0: The channel does not interrupt the processor when the transfer of the last frame starts.

- BLOCK_IE: End block interrupt enable (end of block)

  block_ie = 1: The channel sends an interrupt to the processor when the transfer of the block completes.

  block_ie = 0: The channel does not interrupt the processor when the transfer of the block completes.

The DMA channel status register is written by the DMA to reflect the channel status. It can be read by the processor by polling or after an interrupt. After a functional read, all the DMA_CSR bits are automatically cleared. However, DMA_CSR bits are not cleared after emulation read.

Table 56. DMA Channel Status Register (DMA_CSR)

Base Address = 0xFFFE D800, Offset Address = 0x06 + n*0x40

Bit	Name	Function	R/W	Reset
15:7	RESERVED	Reserved	R	N/A
6	SYNC	Synchronization status	R	0
5	BLOCK	End block (end of block)	R	0
4	LAST	Last frame (start of last frame)	R	0
3	FRAME	Frame (end of frame)	R	0
2	HALF	Half frame (half of frame)	R	0
1	DROP	Event drop (request collision)	R	0
0	TOUT	Timeout in the channel (timeout error)	R	0

A status bit is not set if the corresponding interrupt enable bit in the DMA_CICR register = 0.

- TOUT: Timeout in the channel (timeout error)

  tout = 1: Timeout occurred in channel.
  tout = 0: No timeout error occurred in channel.

- DROP: Event drop (request collision)

  drop = 1: Synchronization event drop occurred during the transfer.
  drop = 0: No event drop occurred during the transfer.

- HALF: Half frame (half of frame)

  half = 1: First half of the current frame has been transferred.
  half = 0: First half of the current frame has not finished transferring yet.

- FRAME (end of frame)

  frame = 1: A complete frame has been transferred.
  frame = 0: Transfer of the current frame is still in progress.

- LAST: Last frame (start of last frame)

  last = 1: The transfer of the last frame has started.
  last = 0: Last frame has not started yet.

- ❑ BLOCK: End block (end of block)

  block = 1: The current transfer in the channel has been finished, but another one may have started if DMA_CCR2[AUTO_INIT] = 1.

  block = 0: Current transfer has not finished yet.

- ❑ SYNC: Synchronization status

  Set to 1 when a DMA request is made in a synchronized channel. When there is a TIPB read access to DMA_CSR register, this bit returns to zero.

  Sync = 1: Logical channel is servicing synchronized DMA request.

  Sync = 0: Logical channel is not servicing a synchronized channel, or DMA request has not been scheduled.

Table 57. DMA Channel Source Start Address Lower Bits Register (DMA_CSSA_L)

Base Address = 0xFFFE D800, Offset Address = 0x08 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SSAL	Source start address, lower bits	R/W	N/A

- ❑ Source start address, lower bits

  Lower bits of the source start address, expressed in bytes. The source start address generated by the DMA is up to a 32-bit address, made of the concatenation of DMA_CSSA_U and DMA_CSSA_L.

Table 58. DMA Channel Source Start Address Upper Bits Register (DMA_CSSA_U)

Base Address = 0xFFFE D800, Offset Address = 0x0A + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SSAU	Source start address, upper bits	R/W	N/A

- ❑ Source start address, upper bits

  Upper bits of the source start address, expressed in bytes. The source start address generated by the DMA is a byte address, made of concatenation of DMA_CSSA_U and DMA_CSSA_L.

Table 59. DMA Channel Destination Start Address Lower Bits Register (DMA_CDSA_L)

Base Address = 0xFFFE D800, Offset Address = 0x0C + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSAL	Destination start address, lower bits	R/W	N/A

- ❑ Destination start address, lower bits

  Lower bits for the destination start address, expressed in bytes. The destination start address is up to a 32-bit address, made of the concatenation of DMA_CDSA_U and DMA_CDSA_L.

Table 60. DMA Channel Destination Start Address Upper Bits Register (DMA_CDSA_U)

Base Address = 0xFFFE D800, Offset Address = 0x0E + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DSAU	Destination start address, upper bits	R/W	N/A

- ❑ Destination start address, upper bits

  Upper bits for the source start address, expressed in bytes. The destination start address is made of the concatenation of DMA_CDSA_U and DMA_CDSA_L.

Table 61. DMA Channel Element Number Register (DMA_CEN)

Base Address = 0xFFFE D800, Offset Address = 0x10 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	EN	Channel element number	R/W	N/A

- ❑ Element number

  Number of elements within a frame (unsigned). The maximum element number is 65535.

Table 62. DMA Channel Frame Number Register (DMA_CFN)

Base Address = 0xFFFE D800, Offset Address = 0x12 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	FN	Channel frame number	R/W	N/A

❑ Frame number

Number of frames within the block to be transferred (unsigned). The maximum frame number is 65535.

The size in bytes of the data block to transfer is data_block_in_bytes = DMA_CFN x DMA_CEN x DMA_CSDP[data_type].

Table 63. DMA Channel Source Frame Index Register (DMA_CSFI)

Base Address = 0xFFFE D800, Offset Address = 0x14 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SFI	Channel source frame index	R/W	N/A

❑ Channel source frame index

Contains the channel source frame index, expressed as signed value in bytes, which is used to compute addresses when double indexed addressing mode is used in DMA source port.

Table 64. DMA Channel Source Element Index Register (DMA_CSEI)

Base Address = 0xFFFE D800, Offset Address = 0x16 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SEI	Channel source element index	R/W	N/A

❑ Channel source element index

Contains the channel source element index, expressed as signed value in bytes, which is used to compute the addresses when single index addressing mode is used.

Table 65. DMA Channel Source Element Index Register (DMA_CSAC)

Base Address = 0xFFFE D800, Offset Address = 0x18 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	SAC	Source element/frame address 16 LSB	R	N/A

This register can be used to monitor the progress of a DMA transfer on channel source port:

❑ It is a snapshot of the source address generated by the channel source address counter, which is scheduled in the channel source port.

❑ It is incremented on each access made on the channel source port (S8, S16 or S32).

Note that in OMAP3.0/3.1 the address used by this register was dedicated to DMA_CPC register, which does not exist anymore.

Table 66. DMA Channel Destination Address Counter Register (DMA_CDAC)

Base Address = 0xFFFE D800, Offset Address = 0x1A + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DAC	Destination element/frame address 16 LSB	R	N/A

This register can be used to monitor the progress of a DMA transfer on channel destination port:

❑ It is a snapshot of the destination address generated by the channel destination address counter, which is scheduled in the channel destination port.

❑ It is incremented on each access made on the channel destination port (S8, S16 or S32).

Table 67. DMA Channel Destination Element Index Register (DMA_CDEI)

Base Address = 0xFFFE D800, Offset Address = 0x1C + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DEI	Channel destination element index	R/W	N/A

❑ Channel destination element index

Contains the channel destination element index, expressed as a signed value in bytes, which is used to compute the addresses, when single/double indexed addressing mode is used.

Note:

When DMA_ CCR[10] = 1, then destination_element_index = DMA_CDEI.

When DMA_ CCR[10] = 0, then destination_element_index = DMA_CSEI.

Table 68. DMA Channel Destination Frame Index Register (DMA_CDFI)

Base Address = 0xFFFE D800, Offset Address = 0x1E + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	DFI	Channel destination frame index	R/W	N/A

- ❑ Channel destination frame index

  Contains the channel destination frame index, expressed as a signed value in bytes, which is used to compute the addresses when double indexed addressing mode is used.

Note:

When DMA_CCR[10] = 1, then destination_frame_index = DMA_CDFI
When DMA_CCR[10] = 0, then destination_frame_index = DMA_CSFI

Table 69. DMA COLOR Parameter Register (DMA_COLOR_L)

Base Address = 0xFFFE D800, Offset Address = 0x20 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	FCL	Channel BLT foreground color (LSW)	R/W	N/A

This register can be used to provide parameter for DMA constant fill and transparent copy features. It must be configured in big endian format.

- ❑ If DMA_CCR2[Constant_Fill_Enable] = 1, then it defines the parameter for constant filling.

  If data_type = 8 bit, then

  DMA_COLOR_L[7:0]: it defines parameter for constant filling

  If data_type = 16 bit, then

  DMA_COLOR_L[15:0]: it defines parameter for constant filling

  If data_type = 32 bit, then

  DMA_COLOR_L[15:0]: it defines parameter[15:0] (LSW) for constant filling

  DMA_COLOR_U[15:0]: it defines parameter[31:16] (MSW) for constant filling

- ❏ If DMA_CCR2[Transparent_Copy_Enable] = 1, then it defines color key parameter for transparent copy.

  If data_type = 8 bit, then

  DMA_COLOR_L[7:0]: it defines color key for transparent copy

  If data_type = 16 bit, then

  DMA_COLOR_L[15:0]: it defines color key for transparent copy

  If data_type = 32 bit, then

  DMA_COLOR_L[15:0]: it defines color key[15:0] (LSW) for transparent copy

  DMA_COLOR_U[15:0]: it defines color key[31:16] (MSW) for transparent copy

Table 70. DMA COLOR Parameter Register (DMA_COLOR_U)

Base Address = 0xFFFE D800, Offset Address = 0x22 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0	FCU	Channel BLT foreground color (MSW)	R/W	N/A

For more details, see Table 69, *DMA COLOR Parameter Register.*

Table 71. DMA Channel Control Register 2 (DMA_CCR2)

Base Address = 0xFFFE D800, Offset Address = 0x24 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:3	RESERVED	Reserved	R/W	N/A
2	BS	Block Synchronization	R/W	N/A
1	TCE	Transparent copy enable	R/W	N/A
0	CFE	Constant fill enable	R/W	N/A

- ❏ BS**:** Block synchronization

  This bit and the fs bit in the DMA_CCR register are used to program the way that a DMA_request is serviced in a synchronized transfer.

- ❏ Transparent copy enable

  1: Transparent copy operation is enabled. During transparent copy operation, any source data type that matches the DMA_COLOR_U/L registers is not written to the destination.

  0: Transparent copy operation is disabled. During transparent copy operation, any source pixel is written to the destination (No mask, brush are supported).

- ❑ Constant fill enable

  1: Constant fill operation is enabled. During constant fill operation, it writes destination with DMA_COLOR_U/L, instead of data from source.

  0: Constant fill operation is disabled. During BLT operation, any source data is written to the destination without constant fill.

Table 72. DMA Logical Channel Link Control Register (DMA_CLNK_CTRL)

Base Address = 0xFFFE D800, Offset Address = 0x28 + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15	EL	Enable_Lnk	R/W	0
14	SL	Stop_Lnk	R/W	0
13:5	RESERVED	Reserved	R/W	N/A
4	NID	Next LCh_ID	R/W (read as 0)	0
3:0	NID	Next LCh_ID	R/W	0000

Used to control logical channel linked queue.

- ❑ NEXTLCH_ID**:** Defines the NEXTLCH_ID, which is used to build logical channel chaining queue. The LOGICAL_CHANNEL_I is enabled**,** after the current logical channel finishes transfer (i = 0-15).

- ❑ STOP_LNK: Disables the logical channel on channel linked queue.

  1: The logical channel, defined by NEXTLCH_ID, is disabled, and ENABLE_LNK is disabled.

  0: No logical channel in chained is disabled.

- ❑ ENABLE_LNK: Defines the logical channel that is on channel linked queue.

  1: The logical channel, defined by NEXTLCH_ID, is enabled after the current channel finishes transferring.

  0: No logical channel is chained after the current logical channel.

Table 73. DMA Logical Channel Control Register (DMA_LCH_CTRL)

Base Address = 0xFFFE D800, Offset Address = 0x2A + n*0x40				
Bit	**Name**	**Function**	**R/W**	**Reset**
15	LID	LCH Interleave Disable	R/W	0
14:4	RESERVED	Reserved	R/W	N/A
3:0	LT	LCH Type	R/W	0000

Used to control logical channel access.

- ❑ LCH_Type: Defines the logical channel assignment relationship to physical channel and associated features.
  - ❑ 0000: LCh-2D dynamically shares PCh-0 and -1
  - ❑ 0001: LCh-G dynamically shares PCh-0 and -1
  - ❑ 0010: LCh-P dynamically shares PCh-0 and -1
  - ❑ 0111: LCh-PD PCh-2

  All other configurations result in undefined effects.

- ❑ LCH_Interleave_Disable: Defines synchronized logical channel interleave mode enable.

  0: Synchronized logical channel interleave mode enabled. The logical channel transfer can be interleaved at the end of DMA request transfers.

  1: A synchronized logical channel interleave mode is disabled. The logical channel takes control of the PCh until the entire DMA data has been transferred, regardless of DMA requests. The LCh/PCh cannot be preempted when interleaving is disabled, regardless of the LCh priority setting.

3.3.3 LCD Channel Dedicated Registers

Table 74 lists the DMA channel dedicated registers. Table 75 through Table 96 describe the register bits.

Table 74. DMA Logical Channel Configuration Registers

Base Address = FFFE EC00			
Name	**Description**	**R/W**	**Offset**†
DMA_LCD_CSDP	DMA LCD channel source destination parameters	R/W	0xC0
DMA_LCD_CCR	DMA LCD channel control	R/W	0xC2

† Some offsets are not in numerical order to facilitate the relationship between the upper and lower words of some registers.

Table 74. DMA Logical Channel Configuration Registers (Continued)

Base Address = FFFE EC00			
Name	**Description**	**R/W**	**Offset**[†]
DMA_LCD_CTRL	DMA LCD control	R/W	0xC4
TOP_B1_L	DMA LCD top address B1 L	R/W	0xC8
TOP_B1_U	DMA LCD top address B1 U	R/W	0xCA
BOT_B1_L	DMA LCD bottom address B1 L	R/W	0xCC
BOT_B1_U	DMA LCD bottom address B1 U	R/W	0xCE
TOP_B2_L	DMA LCD top address B2 L	R/W	0xD0
TOP_B2_U	DMA LCD top address B2 U	R/W	0xD2
BOT_B2_L	DMA LCD bottom address B2 L	R/W	0xD4
BOT_B2_U	DMA LCD bottom address B2 U	R/W	0xD6
DMA_LCD_SRC_EI_B1	DMA LCD source element index B1	R/W	0xD8
DMA_LCD_SRC_FI_B1_L	DMA LCD source frame index B1 L	R/W	0xDA
DMA_LCD_SRC_FI_B1_U	DMA LCD source frame index B1 U	R/W	0xF4
DMA_LCD_SRC_EI_B2	DMA LCD source element index B2	R/W	0xDC
DMA_LCD_SRC_FI_B2_L	DMA LCD source frame index B2 L	R/W	0xDE
DMA_LCD_SRC_FI_B2_U	DMA LCD source frame index B2 U	R/W	0xF6
DMA_LCD_SRC_EN_B1	DMA LCD source element number B1	R/W	0xE0
DMA_LCD_SRC_FN_B1	DMA LCD source frame number B1	R/W	0xE4
DMA_LCD_SRC_EN_B2	DMA LCD source element number B2	R/W	0xE2
DMA_LCD_SRC_FN_B2	DMA LCD source frame number B2	R/W	0xE6
DMA_LCD_LCH_CTRL	DMA LCD logical channel control	R/W	0xEA

[†] Some offsets are not in numerical order to facilitate the relationship between the upper and lower words of some registers.

Table 75. DMA LCD Channel Source Destination Parameters Register (DMA_LCD_CSDP)

Base Address = 0xFFFE E300, Offset Address = 0xC0				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:14	BURST_EN_B2	Burst enable for block 2. (Tie off = 10)	R/W	00
13	PACK_EN_B2	Pack enable for block 2	R/W	0
12:11	DATA_TYPE_B2	Data type for block 2. (Tie off = 10)	R/W	00
10:9	RESERVED	Reserved	R/W	ND
8:7	BURST_EN_B1	Burst enable for block 1. (Tie off = 10)	R/W	00
6	PACK_EN_B1	Pack enable for block 1	R/W	0
5:2	RESERVED	Reserved	R/W	ND
1:0	DATA_TYPE_B1	Data type for block 1. (Tie off = 10)	R/W	00

The OMAP_3.1_mode tie-off values are the values given to the bits in OMAP3.1 compatible mode, since the DMA_LCD_CCR, DMA_LCH_CTRL, and DMA_LCD_CSDP registers do not exist in the compatible mode. These values are tied off/on in hardware.

Used to control the LCD channel dual/single block transferring.

❑ data_type_b1/data_type_b2: Defines the type of data moved in the LCD channel from source port for block_1/block_2

DATA_TYPE [15:14] or [8:7]	type
00	s8: 8 bits scalar
01	s16: 16 bits scalar
10	s32: 32 bits scalar
11	illegal value

Start_Address must be aligned on the boundary of the type of data moved. For example, if data_type is s32 the source start address must be aligned on a word32. If data_type is s8, source start_address can have any value.

It is the software’s responsibility to make sure that start address is aligned with channel data_type.

- ❑ PACK_EN_B1/PACK_EN_B2: packing enable for block_1/block_2 transfer

  The DMA ports can have a data bus width different from the type of data moved by the DMA channel. For example, s8 data_type can be read on a 32-bit DMA port. The DMA channel has the capacity to pack four consecutive s8 data reads in a single 32-bit read access in order to increase transfer bandwidth.

  Pack_en_b1/Pack_en_b2 = 1: The source port makes packed accesses.

  Pack_en_b1/Pack_en_b2 = 0: The source port never makes packed accesses.

- ❑ BURST_EN_B1 / BURST_EN_B2: burst enable for block_1 / block_2 transfer

  Used to enable bursting on the source port. When bursting is enabled, the source port performs bursts 4 x src_width access. When bursting is disabled, the source port performs single accesses of src_width access.

SRC_BURST_EN[1:0]	Burst Type
00	Single access
01	Reserved
10	Burst 4x port_width (Support 4x32bit accesses)
11	Reserved

If the source port of the channel has no burst access capability, this field is ignored.

Table 76. DMA LCD Channel Control Register (DMA_LCD_CCR)

Base Address = 0xFFFE E300, Offset Address =0xC2				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:14	SRC_AMODE_B2	Source addressing mode for block 2. (Tie off: 01)	R/W	00
13:12	SRC_AMODE_B1	Source addressing mode for block 2. (Tie off 01)	R/W	00
11	end_prog	End of programming status bit. (Tie off = 0)	R/W rst	00
10	OMAP3_1_Compatible_ disable	OMAP3.1 channel compatibility control.	R/W	0
9	REPEAT	Repetitive operation. (Tie off = 1)	R/W	0
8	AUTOINIT	Autoinitialize at the end of the transfer. (Tie off = 1)	R/W	0
7	ENABLE	Enable transfer. (Tie off = 1)	R/W rst	0

Table 76. DMA LCD Channel Control Register (DMA_LCD_CCR) (Continued)

Bit	Name	Function	R/W	Reset
6	PRIO	Channel priority. (Tie off = 1)	R/W	0
5	RESERVED	Reserved	R/W	ND
4	BS	Block synchronize	R/W	ND
3:0	RESERVED	Reserved	R/W	ND

The OMAP_3.1_mode tie-off values are the values given to the bits in OMAP3.1 compatible mode, since the DMA_LCD_CCR, DMA_LCH_CTRL, and DMA_LCD_CSDP registers do not exist in the compatible mode. These values are tied off/on in hardware.

❑ bs: block synchronize

If the external LCD controller is the destination and bs = 1:

Then the DMA LCD channel is synchronized on blocks. This means a block transfer is started each time the LCh is enabled and a hardware synchronization signal is received from the external LCD controller.

- One DMA LCD channel request triggers one block transfer, in both single and dual block modes.
- Two DMA LCD channel requests are required to trigger two block transfers, even if it is in dual block mode.

If the external LCD controller is enabled and bs = 0:

The LCD channel is a software-triggered channel. This means a transfer can start as soon as the LCh is enabled.

If the OMAP LCD controller is the destination, the bs bit is ignored.

If auto init and repeat and/or end prog are set, a hardware request for successive block transfers is required.

❑ Prio_0 : Channel priority

Prio_0 = 1: The channel has the high-priority level.

Prio_0 = 0: The channel has the low-priority level.

❑ Enable:

This bit is used to enable/disable the transfer in the DMA LCD channel when OMAP external LCD controller is selected.

Enable = 1: The transfer starts.

Enable = 0: The transfer stops and is reset.

This bit is automatically cleared by the DMA hardware once the transfer is completed. However, if AUTO_INIT = 1 and REPEAT = 1, the channel continues without being disabled. Clearing of this bit by the DMA has priority over a write by the processor. If both occur simultaneously, the configuration write is discarded.

Note:

If the software must use this bit as a flag, it must do a back to back read to make sure that correct value of this enable bit is latched. This is because the enable bit can be asynchronously reset at the end of channel transfer, and a wrong value may send back when this risk condition happens.

- AUTO_INIT: Autoinitialization at the end of the transfer

  auto_init = 1: LCD channel is in autoinitialization mode. Once the current transfer is complete, the channel automatically reinitializes itself and starts a new transfer, if REPEAT or END_PROG bits is set. Else, if neither REPEAT nor END_PROG equals 1, wait for the END_PROG to be set to 1 to reload the channel with a new context and let the transfer restart.

  auto_init = 0: LCD channel is in non-autoinitialization mode. The DMA channel stops at the end of the current transfer.

- REPEAT: repetitive operation

  Repeat = 1: When LCD channel is in autoinitialization mode, once the current transfer is complete the channel automatically reinitializes itself and starts a new transfer with the previous or a new context depending on the end_prog bit (0/1).

  Repeat = 0: When LCD channel is in autoinitialization mode, once the current transfer is complete the channel automatically reinitializes itself and starts a new transfer only if END_PROG = 1.

  The repeat bit is not considered if AUTO_INIT = 0.

- OMAP3_1_COMPATIBLE_DISABLE: Omap3.1 channel compatibility control.

  This bit is used to set DMA LCD channel programming model.

  omap3_1_compatible_disable = 1: The LCD channel is in OMAP3.2 compatible mode.

  omap3_1_compatible_disable = 0: The LCD channel is in OMAP3.1 compatible mode.

- ❑ END_PROG: end of programming status bit

  end_prog = 1: When LCD channel is in autoinitialization mode, it allows the channel to reinitialize itself with a new channel context (the program register set is copied to the active register set) after the current DMA channel transfer has been finished. The end_prog bit automatically resets itself when the new context has been loaded.

  end_prog = 0: When LCD channel is in autoinitialization mode, and REPEAT bit is 1, DMA continues LCD next transfer with the same channel context. If REPEAT = 0, the channel does not reinitialize itself and does not start a new transfer.

  The end_prog bit is not considered if AUTO_INIT = 0.

- ❑ SRC_AMODE_B1/SRC_AMODE_B2: Source addressing mode for block1 or block2 transfer. This field is used to choose the addressing mode on the source port of the channel.

  src_amode_b1/b2 = 00: Reserved

  src_amode_b1/b2 = 01: Post-incremented address

  src_amode_b1/b2 = 10: Single index (element index)

  src_amode_b1/b2 = 11: Double index (element and frame indexes)

There is no need for a dst_amode_b1/dst_amode_b2 because there is no write address to a destination port in the LCD channel case.

Table 77. DMA LCD Control Register (DMA_LCD_CTRL)

Base Address = 0xFFFE E300, Offset Address = 0xC4

Bit	Name	Function	R/W	Reset
15:9	RESERVED	Reserved.	R/W	ND
8	LDP	LCD destination port (OMAP or external LCD controller).	R/W	0
7:6	LSP	LCD source port. Memory source for the LCD channel.	R/W	00
5	BUS_ERROR_IT_ COND	Bus error interrupt condition.	R/W (read as 0)	0
4	BLOCK_2_IT_COND	Block 2 interrupt condition.	R/W	0
3	BLOCK_1_IT_COND	Block 1 interrupt condition.	R/W	0
2	BUS_ERROR_IT_IE	Buss error interrupt enable.	R/W	0
1	BLOCK_IT_IE	Block interrupt enable.	R/W	0
0	BLOCK_MODE	Type of block mode used for the LCD transfer.	R/W	0

The DMA LCD control register contains nine bit-fields, which control the LCD channel operation. There are two cases of interrupt: end block buffer or abort on the bus (bus error). Bits block_it_ie and bus_error_iie (interrupt enable) enable the generation of the interrupt.

If the status bits (xxx_cond bits and the corresponding interrupt enable bits xxx_ie bits) are all set, an interrupt signal is sent from the DMA LCD channel to the CPU. The CPU reads this register to find the cause of the interrupt.

Users must write to this register to clear the status bits.

- BLOCK_MODE: Type of block mode used for the LCD transfer

  0: One block buffer; only registers relative to block 1 are used.

  1: Two block buffers; the LCD channel reads alternatively top_block_1 and top_block_2.

- BLOCK_IT_IE: end block interrupt enable

  0: Interrupt disabled.

  1: Interrupt enabled.

  This bit enables an end of block interrupt for either block 1 or 2 when dual block mode is selected.

- BUS_ERROR_IT_IE: bus error interrupt enable

  0: Interrupt disabled.

  1: Interrupt enabled.

- BLOCK_1_IT_COND: status LCD channel bit. Users must write to this register to clear the status bits.

  0: No end of block 1 interrupt detected.

  1: End of block 1 interrupt detected.

- BLOCK_2_IT_COND: status LCD channel bit. Users must write to this register to clear the status bits.

  0: No end of block 2 interrupt detected.

  1: End of block 2 interrupt detected.

- BUS_ERROR_IT_COND: status LCD channel bit. Users must write to this register to clear the status bits.

  0: No bus error interrupt detected.

  1: Bus error interrupt detected.

- ❑ LCD_SOURCE_PORT: memory source for the LCD channel

  This bit indicates which memory source is selected for the next LCD transfer.

  00: Memory source is SDRAM.

  01: Memory source is L3_OCP_T1 (Camera).

  10: Memory source is L3_OCP_T2 (Test SRAM).

  11: Reserved.

- ❑ LCD_DESTINATION_PORT: LCD controller for the LCD channel

  This bit indicates which LCD controller is selected for the next LCD transfer.

  0: OMAP controller is connected to the DMA LCD channel.

  1: External LCD controller is connected to the DMA LCD channel.

Note:

To enable the external LCD controller, users must have the external LCD clock active (even if it is not ready to send an image). The following must be set to accomplish this:

DMA_GSCR.OMAP31_MAPPING_DISABLE = 1

DMA_LCD_CTRL.ICD_DESTINATION_PORT = 1

DMA_LCD_CCR.OMAP31_COMPATIBLE_DISABLE = 1

In order to enter deep idle mode, users must set DMA_LCD_CTRL.LCD_DESTINATION_PORT = 0. Indeed, DMA_LCD_CTRL.LCD_DESTINATION_PORT = 1 is a condition that prevents OMAP3.2 from going to idle state because the clock request corresponding to external LCD controller clock is kept active. Consequently, deep idle mode cannot be initiated.

DMA LCD Top Address B1 Registers

The LCD top address B1 registers are two 16-bit registers, which contain the start address for the video RAM buffer 1. The 32-bit address is obtained by the concatenation of the two word16 as described here:

LCD_TOP_B1 = DMA_LCD_TOP_B1_U & DMA_LCD_TOP_B1_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

Table 78. DMA LCD Top Address B1 L Register (TOP_B1_L)

Base Address = 0xFFFE E300, Offset Address = 0xC8				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1		LCD TOP address for block buffer 1 lower bits	R/W	ND
0		0 (always tied to 0)	R	0

Table 79. DMA LCD Top Address B1 U Register (TOP_B1_U)

Base Address = 0xFFFE E300, Offset Address = 0xCA				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD TOP address for block buffer 1 upper bits	R/W	ND

DMA LCD Bottom Address B1 Registers

The LCD bottom address B1 registers are two 16-bit registers that contain the bottom address for the video RAM buffer 1. The 32-bit address is obtained by the concatenation of the two word16 as described here:

LCD_BOTTOM_B1 = DMA_LCD_BOT_B1_U & DMA_LCD_BOT_B1_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

Table 80. DMA LCD Bottom Address B1 L Register (BOT_B1_L)

Base Address = 0xFFFE E300, Offset Address = 0xCC				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1		LCD BOT address for block buffer 1 lower bits	R/W	ND
0		0 (always tied to 0)	R	0

Table 81. DMA LCD Bottom Address B1 U Register (BOT_B1_U)

Base Address = 0xFFFE E300, Offset Address = 0xCE				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD BOT address for block buffer 1 upper bits	R/W	ND

DMA LCD Top Address B2 Registers

The LCD top address B2 registers are two 16-bit registers that contain the start address for the video RAM buffer 2. The 32-bit address is obtained by the concatenation of the two word16 as described here:

LCD_TOP_B2 = DMA_LCD_TOP_B2_U & DMA_LCD_TOP_B2_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

Table 82. DMA LCD Top Address B2 L Register (TOP_B2_L)

Base Address = 0xFFFE E300, Offset Address = 0xD0				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1		LCD TOP address for block buffer 2 lower bits	R/W	ND
0		0 (always tied to 0)	R	0

Table 83. DMA LCD Top Address B2 U Register (TOP_B2_U)

Base Address = 0xFFFE E300, Offset Address = 0xD2				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD TOP address for block buffer 2 upper bits	R/W	ND

DMA LCD Bottom Address B2 Registers

The LCD bottom B2 address registers are two 16-bit registers that contain the bottom address for the video RAM buffer 2. The 32-bit address is obtained by the concatenation of the two word16 as described here:

LCD_BOTTOM_B2 = DMA_LCD_BOT_B2_U & DMA_LCD_BOT_B2_L

Note:

The LSB of the word32 is equal to zero. Address of video buffer must always be even.

Table 84. DMA LCD Bottom Address B2 L Register (BOT_B2_L)

Base Address = 0xFFFE E300, Offset Address = 0xD4				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:1		LCD BOT address for block buffer 2 lower bits	R/W	ND
0		0 (always tied to 0)	R	0

Table 85. DMA LCD Bottom Address B2 U Register (BOT_B2_U)

Base Address = 0xFFFE E300, Offset Address = 0xD6				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD BOT address for block buffer 2 upper bits	R/W	ND

Table 86. DMA LCD Source Element Index B1 Register (DMA_LCD_SRC_EI_B1)

Base Address = 0xFFFE E300, Offset Address = 0xD8				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source element index for block 1	R/W	ND

❑ LCD channel source element index for block 1

Contains the channel source element index for LCD video RAM buffer 1, expressed as signed value in bytes, which is used to compute the addresses when single or double-indexed addressing mode is used.

Table 87. DMA LCD Source Frame Index B1 Register L (DMA_LCD_SRC_FI_B1_L)

Base Address = 0xFFFE E300, Offset Address = 0xDA				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source frame index for block 1 L	R/W	ND

Table 88. DMA LCD Source Frame Index B1 Register U (DMA_LCD_SRC_FI_B1_U)

Base Address = 0xFFFE E300, Offset Address = 0xF4				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source frame index for block 1 U	R/W	ND

❑ LCD channel source frameindex for block 1

These registers contain the channel source frame index for video RAM buffer 1, expressed as a signed value in bytes, which is used to compute addresses when double-indexed addressing mode is used.

Table 89. DMA LCD Source Element Index B2 Register (DMA_LCD_SRC_EI_B2)

Base Address = 0xFFFE E300, Offset Address = 0xDC				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source element index for block 2	R/W	ND

❑ LCD channel source element index for block 2

Contains the channel source element index for LCD video RAM buffer 2, expressed as signed value in bytes, which is used to compute the addresses when single or double-indexed addressing mode is used.

Table 90. DMA LCD Source Frame Index B2 Register L (DMA_LCD_SRC_FI_B2_L)

Base Address = 0xFFFE E300, Offset Address = 0xDE				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source frame index for block 2 L	R/W	ND

Table 91. DMA LCD Source Frame Index B2 Register U (DMA_LCD_SRC_FI_B2_U)

Base Address = 0xFFFE E300, Offset Address = 0xF6				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD source frame index for block 2 U	R/W	ND

❑ LCD channel source frame index for block 2

Contains the channel source frame index for video RAM buffer 2, expressed as signed value in bytes, which is used to compute addresses when double-indexed addressing mode is used.

Table 92. DMA LCD Source Element Number B1 Register (DMA_LCD_SRC_EN_B1)

Base Address = 0xFFFE E300, Offset Address = 0xE0				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD channel source element number for block 1	R/W	ND

❑ LCD channel source element number for block 1

Number of elements within a frame (unsigned) for the video RAM buffer 1. The maximum element number is 65535.

Table 93. DMA LCD Source Frame Number B1 Register (DMA_LCD_SRC_FN_B1)

Base Address = 0xFFFE E300, Offset Address = 0xE4				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD channel source frame number for block 1	R/W	ND

❑ LCD channel source frame number for block 1

Number of frames within a block (unsigned) for the video RAM buffer 1. The maximum frame number is 65535.

The size in bytes of the data block to transfer is:

data_block_in_bytes = DMA_LCD_SRC_FN_B1 x
DMA_LCD_SRC_EN_B1 x
DMA_LCD_CDSP[DATA_TYPE_B1]

Table 94. DMA LCD Source Element Number B2 Register (DMA_LCD_SRC_EN_B2)

Base Address = 0xFFFE E300, Offset Address = 0xE2				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD channel source element number for block 2	R/W	ND

❑ LCD channel source element number for block 2

Number of elements within a frame (unsigned) for the video RAM buffer 2. The maximum element number is 65535.

Table 95. DMA LCD Source Frame Number B2 Register (DMA_LCD_SRC_FN_B2)

Base Address = 0xFFFE E300, Offset Address = 0xE6				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:0		LCD channel source frame number for block 2	R/W	ND

❑ LCD channel source frame number for block 2

Number of frames within a block (unsigned) for the video RAM buffer 2. The maximum frame number is 65535. The size in bytes of the data block to transfer is:

data_block_in_bytes = DMA_LCD_SRC_FN_B2 x
DMA_LCD_SRC_EN_B2 x
DMA_LCD_CSDP[DATA_TYPE_B2]

Table 96. DMA LCD Logical Channel Control Register (DMA_LCD_LCH_CTRL)

Base Address = 0xFFFE E300, Offset Address = 0xEA				
Bit	**Name**	**Function**	**R/W**	**Reset**
15:4	RESERVED	Reserved	R/W	ND
3:0	LCH_TYPE	Logical channel type.	R/W	0000

Used to control logical channel access. The OMAP_3.1_mode tie-off values are those values given to the bits in OMAP 3.1 compatible mode, because the DMA_LCD_CCR, DMA_LCH_CTRL, and DMA_LCD_CSDP registers do not exist in the compatible mode. These values are tied off/on in hardware. (For LCH_TYPE, the tie-off value is 0000).

- ❑ LCH_Type:

  Defines logical channel assignment relationship to associated features and physical channel.

  The only possible configuration is: 0100: LCh-D PCh-D. This bit field must be configured as above to secure forward compatibility.

This page is intentionally left blank.

Revision History

Table 97 lists the changes made since the previous version of this document.

Table 97. Document Revision History

Page	Additions/Modifications/Deletions
Global	DSP DMA section was removed, see updated information in the *OMAP5910/5912 Multimedia Processor DSP Subsystem Reference Guide* (SPRU890).

Index