DMA

Direct Memory Access

Revision 1.0.0-41621

September 11, 2015

LIST OF TABLES

6 DMA Controller

6.1 Overview

The DMA controller schedules transfers between devices in the bus structure. It is programmed by the CPU through the ARM AHB bus and it schedules transfers on a separate AHB bus which is shared with the USB bus master DMA.
For more details on the bus structure please refer to the *DICE_III_User_Guide_MemoryMap* document.

DMA Features

- 8 Independent DMA channels, each channel can perform a unidirectional transfer.
- Each channel can be programmed to handshake with one of 8 source peripherals and one of 8 destination peripherals.
- Ability to do burst or single transfers with programmable burst size based on signaling from peripheral.
- Memory-to-memory, memory-to-peripheral, peripheral-to-memory and peripheral-to-peripheral transfers.
- Link list based scatter gather support
- Prioritized transfer based on channel number, channel 0 has the highest priority.
- Incrementing or non-incrementing addressing for source and destination.
- Internal 4x32 FIFO per channel
- Separate interrupts for completion and error

6.1.1 Principle of operation

The DMA is programmed by the CPU to perform transfers between peripherals and/or memory. The memory devices are always ready as a destination or a source. The peripherals will use handshake signals to indicate when they are ready.

Usually the DMA is programmed to transfer a fixed number of words. One of the source peripherals (Ethernet MAC) can be the flow controller and determine how many words are transferred for all other peripherals the number must be programmed by the CPU.

When a transfer is complete the DMA will either disable itself or perform the next transfer in a linked list.

For each DMA transfer it is possible to select if a completion interrupt is generated. Each transfer in a linked list has an individual interrupt enable bit.

The DMA will arbitrate for the bus, read the up to the programmed burst size from the source peripheral and into the internal FIFO. It will then in turn write the data from the FIFO to the destination peripheral using the programmed burst size.

6.1.2 Programming the burst size

The peripherals must be programmed to match the burst size of the DMA. For example if the DMA is programmed to use bursts of 8 bytes when reading from a peripheral, the

peripheral must be programmed to have at least 8 bytes available when issuing a burst handshake. The documentation for each peripheral documents how to program the burst size.

6.1.2.1 Programming the transfer width

The transfer width can be programmed individually for the source and the destination. It can be programmed for 8, 16 or 32 bits. The transfer size programmed into the DMA counts source transfers.

It is important that the programmed transfer size is calculated such that the total number of destination transfers required is an integer number. This is only an issue when the destination width is larger than the source width.

6.1.2.2 Transfer priorities

Each of the 8 channels is prioritized so channel 0 has the highest priority and channel 7 the lowest. After each burst transfer the arbitration will check to see if a higher priority channel is ready.

The two lowest priority channels will insert an IDLE cycle after each 4 transfers of the programmed size. This is to prevent the DMA from starving other AHB masters from accessing the bus. For memory-to-memory transfers it is suggested that one of these two channels be used to prevent starving the CPU, assuming the CPU wants access to the same memory.

6.1.2.3 Endianess

As the CPU and the complete system are little-endian, all transfers are using little endian order. This is only relevant when the source and destination transfer width are different.

6.1.2.4 Locked transfers

Each transfer can be programmed to be locked. If a transfer is locked it means that the burst reads or writes cannot be broken by the arbitration logic even if a higher priority master is waiting for the AHB bus.

In cases where the source width and destination width are the same and the burst size is 4 or more and if the destination is ready to perform the transfer immediately the lock will encapsulate both the source and destination transfer.

6.1.2.5 Protection

The protection signals are not used by any of the DICE III peripherals. If protection is required the MMU of the CPU should be programmed to only allow authorized access to the DMA control registers and the driver should enforce protection by design.

6.2 Programming the DMA

The DMA registers should only be accessed using 32 bit writes or reads. All reserved bits in the registers should be written as zero and ignored on read.

6.2.1 Enabling the DMA Controller

Enable the DMAC by setting the DMA Enable, E, bit in the DMA Configuration Register.

6.2.2 Disabling the DMA Controller

The DMA can be disabled by clearing the DMA Enable, E bit, in the DMA Configuration Register. Before clearing this bit ensure that all the channels are disabled by reading EnabledChannels in the Enabled Channels Register. See *Disabling a DMA channel* below.

6.2.3 Enabling a DMA channel

Before enabling a channel it should be fully configured. Then the Channel Enable, E bit, in the relevant Channel Configuration Register should be set.

6.2.4 Disabling a DMA channel

There are two ways to disable a channel. The simple method is to clear the Channel Enable, E bit, in the Channel Configuration Register. In that case data in the FIFO will be lost and it is not possible to find out how much was transferred.
The other method is to do the following:

- Set the Halt, H bit, in the relevant Channel Configuration Register. This will cause subsequent DMA requests to be ignored.
- Poll the Active, A bit, in the relevant channel Configuration Register until it reaches 0.
- Clear the Channel Enable bit in the relevant channel Configuration Register.

This method assures that the FIFO is flushed and it can be determined how much was actually transferred.

6.2.5 Programming a DMA channel

1. Clear any pending interrupts on the channel you want to use by writing to the Interrupt Terminal Count Clear Register and the Interrupt Error Clear Register. The previous channel operation might have left interrupts active.
2. Write the source address into the Channel Source Address Registers.
3. Write the destination address into the Channel Destination Address Register.
4. Write the address of the next LLI into the Channel Linked List Item Register. If the transfer consists of a single packet of data, you must write 0 into this register.
5. Write the control information into the Channel Control Register.
6. Write the channel configuration information into the Channel Configuration Register. If the Enable bit is set, then the DMA channel is automatically enabled.

6.3 Module Configuration

The DMA controller module is located at the CYGHWR_HAL_DICE3_DMAC base address 0x80000000.

Table 6.1 DMA register summary

Address Offset	Register	Description
0x0000	HAL_DICE3_DMAC_INTST	*Interrupt Status Register*
0x0004	HAL_DICE3_DMAC_INTTCST	*Interrupt Terminal Count Status Register*
0x0008	HAL_DICE3_DMAC_INTTCCLR	*Interrupt Terminal Count Clear Register*
0x000C	HAL_DICE3_DMAC_INTERRST	*Interrupt Error Status Register*
0x0010	HAL_DICE3_DMAC_INTERRCLR	*Interrupt Error Clear Register*
0x0014	HAL_DICE3_DMAC_INTTCRAW	*Raw Interrupt Terminal Count Status Register*
0x0018	HAL_DICE3_DMAC_INTERRRAW	*Raw Error Interrupt Status Register*
0x001C	HAL_DICE3_DMAC_ENST	*Enabled Channel Register Register*
0x0020	HAL_DICE3_DMAC_SFTBREQ	*Software Burst Request Register*
0x0024	HAL_DICE3_DMAC_SFTSREQ	*Software Single Request Register*
0x0028	HAL_DICE3_DMAC_SFTLBREQ	*Software Last Burst Request Register*
0x002C	HAL_DICE3_DMAC_SFTLSREQ	*Software Last Single Request Register*
0x0030	HAL_DICE3_DMAC_CFG	*Configuration Register*
0x0034	HAL_DICE3_DMAC_SYNC	*Synchronization Register*
0x0100	HAL_DICE3_DMAC_CH0	*Channel Registers*
0x0120	HAL_DICE3_DMAC_CH1	
0x0140	HAL_DICE3_DMAC_CH2	
0x0160	HAL_DICE3_DMAC_CH3	
0x0180	HAL_DICE3_DMAC_CH4	
0x01A0	HAL_DICE3_DMAC_CH5	
0x01C0	HAL_DICE3_DMAC_CH6	
0x01E0	HAL_DICE3_DMAC_CH7	

6.4 Interrupt Status Register

Address offset: 0x0000 HAL_DICE3_DMAC_INTST

The read-only HAL_DICE3_DMAC_INTST Register shows the status of the interrupts after masking. A HIGH bit indicates that a specific DMA channel interrupt request is active. You can generate the request from either the error or terminal count interrupt requests.

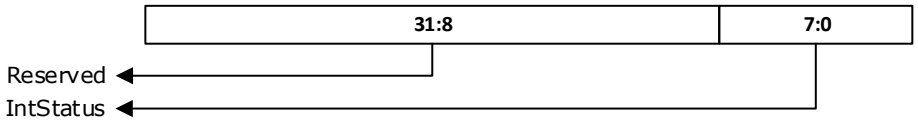

Table 6.2 Interrupt Status Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:8	-	N/A	Read undefined
InitStatus	7:0	0	R	Status of the DMA interrupts after masking

6.5 Interrupt Terminal Count Status Register

Address offset: 0x0004 HAL_DICE3_DMAC_INTCST

The read-only HAL_DICE3_DMAC_INTCST Register indicates the status of the terminal count after masking. You must use this register to ascertain the source of the interrupt request.

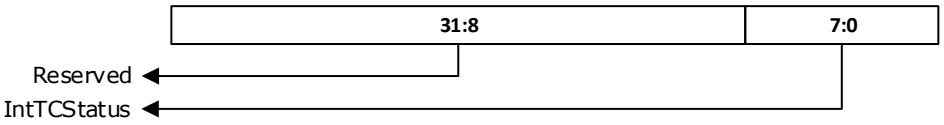

Table 6.3 Interrupt Terminal Count Status Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:8	-	N/A	Read undefined
IntTCStatus	7:0	0	RW	Interrupt terminal count register

6.6 Interrupt Terminal Count Clear Register

Address offset: 0x0008 HAL_DICE3_DMAC_INTCCLR

The write-only HAL_DICE3_DMAC_INTCCLR Register clears a terminal count interrupt request. When writing to this register, each data bit that is set HIGH causes the corresponding bit in the Interrupt Status Register to be cleared. Data bits that are LOW have no effect on the corresponding bit in the register.

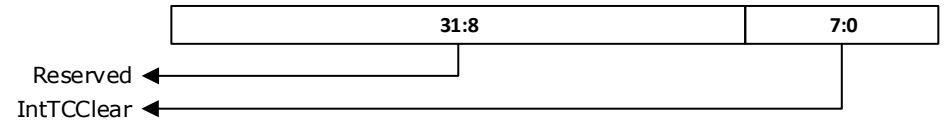

Table 6.4 Interrupt Terminal Count Clear Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:8	-	N/A	Read undefined
IntTCClear	7:0	0	W	Terminal count request clear

6.7 Interrupt Error Status Register

Address offset: 0x000C HAL_DICE3_DMAC_INTERRST
The read-only HAL_DICE3_DMAC_INTERRST Register indicates the status of the error request after masking.

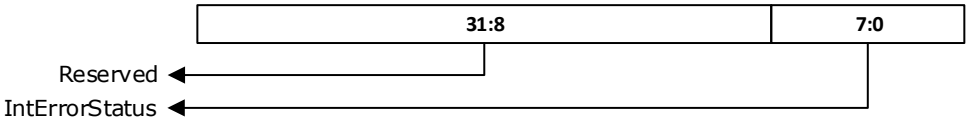

Table 6.5 Interrupt Error Status Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:8	-	N/A	Read undefined
IntErrorStatus	7:0	0	RW	Interrupt error status

6.8 Interrupt Error Clear Register

Address offset: 0x0010 HAL_DICE3_DMAC_INTERRCLR
The write-only HAL_DICE3_DMAC_INTERRCLR Register clears the error interrupt requests. When writing to this register, each data bit that is HIGH causes the corresponding bit in the Interrupt Status Register to be cleared. Data bits that are LOW have no effect on the corresponding bit in the register.

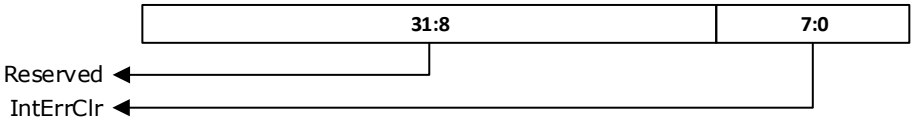

Table 6.6 Interrupt Error Clear Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:8	-	N/A	Read undefined. Write as zero.
IntErrClr	7:0	0	W	Interrupt error clear

6.9 Raw Interrupt Terminal Count Status Register

Address offset: 0x0014 HAL_DICE3_DMAC_INTTCRAW
The read-only HAL_DICE3_DMAC_INTTCRAW Register indicates the DMA channels that are requesting a transfer complete, terminal count interrupt, prior to masking. A HIGH bit indicates that the terminal count interrupt request is active prior to masking.

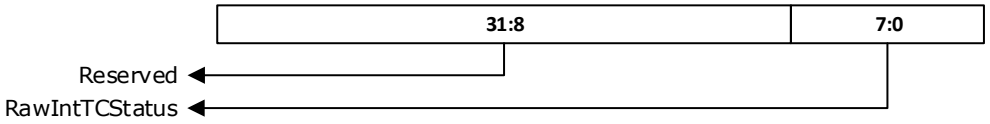

Table 6.7 Raw Interrupt Terminal Count Status Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:8	-	N/A	Read undefined
RawIntTCStatus	7:0	0	RW	Status of the terminal count interrupt prior to masking

6.10 Raw Error Interrupt Status Register

Address offset: 0x0018 HAL_DICE3_DMAC_INTERRRAW

The read-only HAL_DICE3_DMAC_INTERRRAW Register indicates the DMA channels that are requesting an error interrupt prior to masking. A HIGH bit indicates that the error interrupt request is active prior to masking.

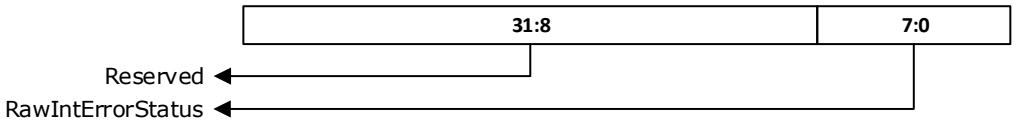

Table 6.8 Raw Error Interrupt Status Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:8	-	N/A	Read undefined
RawIntErrorStatus	7:0	0	RW	Status of the error interrupt prior to masking

6.11 Enabled Channels Register

Address offset: 0x001C HAL_DICE3_DMAC_ENST

The read-only HAL_DICE3_DMAC_ENST Register indicates the DMA channels that are enabled, as indicated by the Enable, E bit, in the relevant Channel Configuration Register. A HIGH bit indicates that a DMA channel is enabled. A bit is cleared on completion of the DMA transfer.

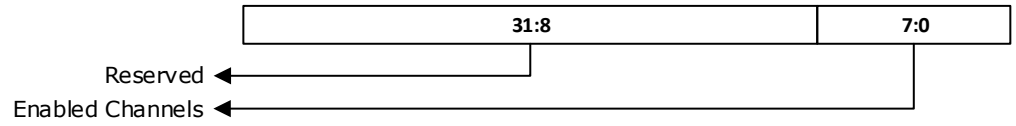

Table 6.9 Enabled Channels Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:8	-	N/A	Read undefined
EnabledChannels	17:0	0	R	Channel enable status

6.12 Software Burst Request Register

Address offset: 0x0020 HAL_DICE3_DMAC_SFTBREQ

The HAL_DICE3_DMAC_SFTBREQ Register enables DMA burst requests to be generated by software. You can generate a DMA request for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect. Reading the register indicates the sources that are requesting DMA burst transfers. You can generate a request from either a peripheral or the software request register.

Note

It is recommended not to use software and hardware peripheral requests at the same time.

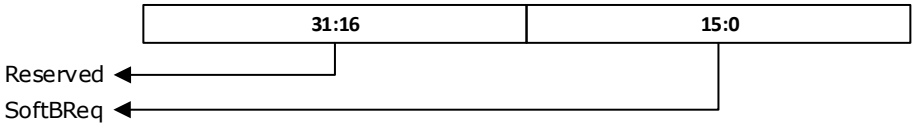

Table 6.10 Software Burst Request Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:16	-	N/A	Read undefined. Write as zero.
SoftBReq	15:0	0	RW	Software burst request

6.13 Software Single Request Register

Address offset: 0x0024 HAL_DICE3_DMAC_SFTSREQ

The HAL_DICE3_DMAC_SFTSREQ Register enables DMA single requests to be generated by software. You can generate a DMA request for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect. Reading the register indicates the sources that are requesting single DMA transfers. You can generate a request from either a peripheral or the software request register.

Note

It is recommended not to use software and hardware peripheral requests the same time.

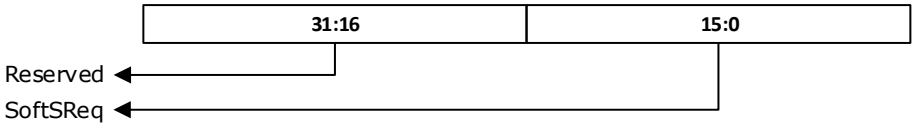

Table 6.11 Software Single Request Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:16	-	N/A	Read undefined. Write as zero.
SoftSReq	15:0	0	RW	Software single request

6.14 Software Last Burst Request Register

Address offset: 0x0028 HAL_DICE3_DMAC_SFTLBREQ

The HAL_DICE3_DMAC_SFTLBREQ Register enables software to generate DMA last burst requests. You can generate a DMA request for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect. Reading the register indicates the sources that are requesting last burst DMA transfers. You can generate a request from either a peripheral or the software request register.

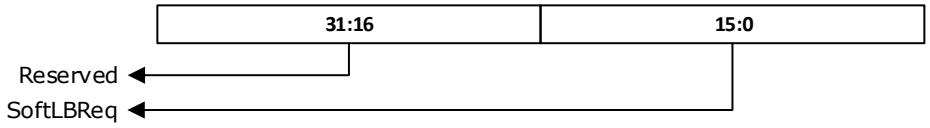

Table 6.12 Software Last Burst Request Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:16	-	N/A	Read undefined. Write as zero.
SoftLBReq	15:0	0	RW	Software last burst request

6.15 Software Last Single Request Register

Address offset: 0x002C HAL_DICE3_DMAC_SFTLSREQ

The HAL_DICE3_DMAC_SFTLSREQ Register enables software to generate DMA last single requests. You can generate a DMA request for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect. Reading the register indicates the sources that are requesting last single DMA transfers. You can generate a request from either a peripheral or the software request register.

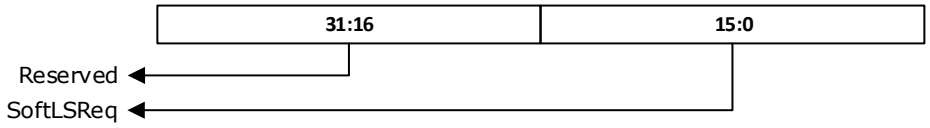

Table 6.13 Software Last Single Request bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:16	-	N/A	Read undefined. Write as zero.
SoftLSReq	15:0	0	RW	Software last single request

6.16 Configuration Register

Address offset: 0x0030 HAL_DICE3_DMAC_CFG

The HAL_DICE3_DMAC_CFG Register configures the operation of the DMAC. You can alter the endianness of the AHB master interfaces by writing to the M1 bit of this register. The AHB master interfaces are set to little-endian mode on reset. Please note that changing the endianness will affect all DMA channels. While it might be feasible in some applications to use this functionality to do endian swapping, care should be taken not to affect other channels which expect the native little-endian mode.

Note

The AHB master interfaces are not required to have the same endianness.

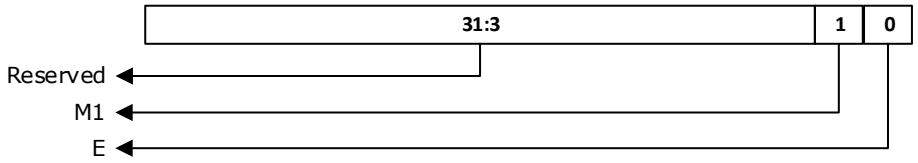

Table 6.14 Configuration Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:2	-	N/A	Read undefined. Write as zero.
M1	1	0	RW	AHB Master 1 endianness configuration: 0 = little-endian mode 1 = big-endian mode.
E	0	0	RW	DMAC enable: 0 = disabled 1 = enabled. Disabling the DMAC reduces power consumption.

6.17 Synchronization Register

Address offset: 0x0034 HAL_DICE3_DMAC_SYNC

The HAL_DICE3_DMAC_SYNC Register enables or disables synchronization logic for the DMA request signals.

The DMA request signals consist of:

DMACBREQ[15:0]
DMACSREQ[15:0]
DMACLBREQ[15:0]
DMACLSREQ[15:0]

A bit set to 0 enables the synchronization logic for a particular group of DMA requests. A bit set to 1 disables the synchronization logic for a particular group of DMA requests. This register is reset to 0, and synchronization logic is enabled.

Note
You must use synchronization logic when the peripheral generating the DMA request runs on a different clock to the DMAC. For peripherals running on the same clock as the DMAC, disabling the synchronization logic improves the DMA request response time. DICE III uses synchronous clocks for all peripherals and it should not be necessary to use synchronization.

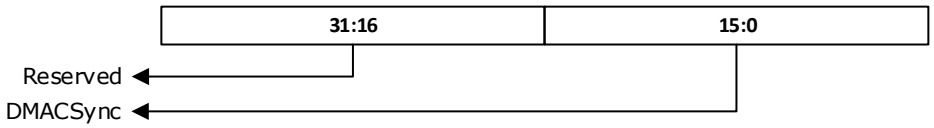

Table 6.15 Synchronization Register bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:16	-	N/A	Read undefined
DMACSync	15:0	0	RW	DMA synchronization logic for DMA request signals enabled or disabled. A LOW bit indicates that the synchronization logic for the request signals is enabled. A HIGH bit indicates that the synchronization logic is disabled.

6.18 Channel Registers

Address offset: 0x0100, 0x0120, 0x0140, 0x0160, 0x0180, 0x01A0, 0x01C0, 0x01E0
HAL_DICE3_DMA_CH<n>

The channel registers are for programming a DMA channel. These registers consist of:

- eight Source Address Registers
- eight Destination Address Registers
- eight Linked List Item Registers
- eight Control Registers
- eight Configuration Registers

When performing scatter/gather DMA, the first four registers are automatically updated.

Note
Unpredictable behavior can result if you update the channel registers when a transfer is taking place. If you want to change the channel configurations, you must disable the channel first and then reconfigure the relevant register.

Each channel register contains the registers shown below.

Table 6.16 DMA Channel register summary

Address Offset	Register	Description
0x0000	HAL_DICE3_DMAC_CHn_SRCADR	*Channel Source Address Registers*
0x0004	HAL_DICE3_DMAC_CHn_DSTADR	*Channel Destination Address Registers*
0x0008	HAL_DICE3_DMAC_CHn_LLI	*Channel Linked List Item Registers*
0x000C	HAL_DICE3_DMAC_CHn_CTRL	*Channel Control Registers*
0x0010	HAL_DICE3_DMAC_CHn_CFG	*Channel Configuration Registers*

6.18.1 Channel Source Address Registers

The eight Source Address Registers, with address offsets of 0x100, 0x120, 0x140, 0x160, 0x180, 0x1A0, 0x1C0, and 0x1E0 respectively, contain the current source address, byte-aligned, of the data to be transferred. Software programs each register directly before the appropriate channel is enabled.

When the DMA channel is enabled, this register is updated:

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

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

Note

You must align source and destination addresses to the source and destination widths.

Table 6.17 Source Address Register bit assignments

Name	Bit	Reset	Dir	Description
SrcAddr	31:0	0	RW	DMA Source Address

6.18.2 Channel Destination Address Registers

The eight Destination Address Registers, with address offsets of 0x104, 0x124, 0x144, 0x164, 0x184, 0x1A4, 0x1C4, and 0x1E4 respectively, contain the current destination address, byte-aligned, of the data to be transferred.

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

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

Table 6.18 Destination Address Register bit assignments

Name	Bit	Reset	Dir	Description
DestAddr	31:0	0	RW	DMA Destination Address

6.18.3 Channel Linked List Item Registers

The eight read/write Channel Linked List Item Registers, with address offsets of 0x108, 0x128, 0x148, 0x168, 0x188, 0x1A8, 0x1C8, and 0x1E8 respectively, contain a word-aligned address of the next LLI. If the LLI is 0, then the current LLI is the last in the chain, and the DMA channel is disabled after all DMA transfers associated with it are completed.

Note
Programming this register when the DMA channel is enabled has unpredictable results.
Note
To make loading the LLIs more efficient, you can make the LLI data structures 4-word aligned.

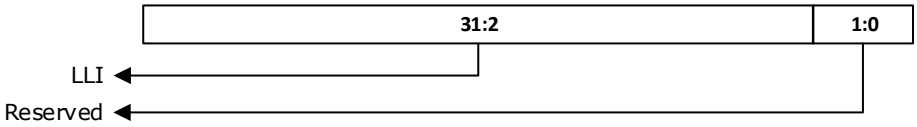

Table 6.19 Destination Address Register bit assignments

Name	Bit	Reset	Dir	Description
LLI	31:2	0	RW	Linked list item. Bits [31:2] of the address for the next LLI. Address bits [1:0] are 0.
Reserved	1:0	0	N/A	Read undefined. Write as zero.

6.18.4 Channel Control Registers

The eight Channel Control Registers, with address offsets of 0x010C, 0x12C, 0x14C, 0x16C, 0x18C, 0x1AC, 0x1CC, and 0x1EC respectively, contain DMA channel control information such as the transfer size, burst size, and transfer width. Software programs each register directly before the DMA channel is enabled.

When the channel is enabled, the register is updated by following the linked list when a complete packet of data has been transferred. Reading the register while the channel is active does not give useful information. This is because by the time that software has processed the value read, the channel might have progressed. It is intended to be read-only when a channel has stopped.

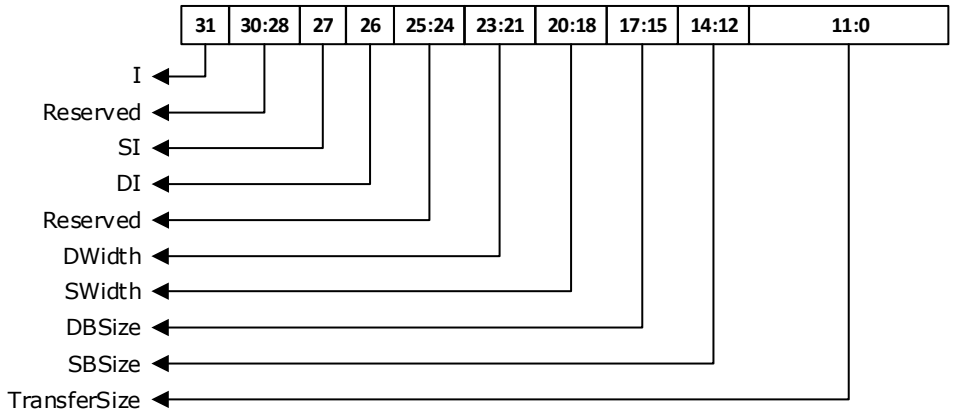

Table 6.20 Channel Control Register bit assignments

Name	Bit	Reset	Dir	Description
I	31	0	RW	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
Reserved	30:28	0	N/A	Reserved.
DI	27	0	RW	Destination increment. When set, the destination address is incremented after each transfer.
SI	26	0	RW	Source increment. When set, the source address is incremented after each transfer.
Reserved	25:24	0	N/A	Reserved.
DWidth	23:21	0	RW	Destination transfer width. Transfers wider than the AHB master bus width are illegal. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data when required.
SWidth	20:18	0	RW	Source transfer width. Transfers wider than the AHB master bus width are illegal. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data when required.

Name	Bit	Reset	Dir	Description
DBSize	17:15	0	RW	Destination burst size. Indicates the number of transfers that make up a destination burst transfer request. You must set this value to the burst size of the destination peripheral, or if the destination is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the DMACxBREQ signal goes active in the destination peripheral. The burst size is not related to the AHB HBURST signal.
SBSize	14:12	0	RW	Source burst size. Indicates the number of transfers that make up a source burst. You must set this value to the burst size of the source peripheral, or if the source is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the DMACxBREQ signal goes active in the source peripheral. The burst size is not related to the AHB HBURST signal.SWidth
TransferSize	11:0	0	RW	Transfer size. A write to this field sets the size of the transfer when the DMAC is the flow controller. This value counts down from the original value to zero, and so its value indicates the number of transfers left to complete. A read from this field provides the number of transfers still to be completed on the destination bus. Reading the register when the channel is active does not give useful information because by the time the software has processed the value read, the channel might have progressed. Only use it when a channel is enabled, and then disabled. Program the transfer size value to zero if the DMAC is not the flow controller. If you program the TransferSize to a non-zero value, the DMAC might attempt to use this value instead of ignoring the TransferSize.

The table below lists the values of the DBSize or SBSsize bits and their corresponding burst sizes.

Table 6.21 Source or destination burst size

Bit value of DBSize or SBSize	Source or destination burst transfer request size
0b000	1
0b001	4
0b010	8
0b011	16
0b100	32
0b101	64
0b110	128
0b111	256

The table below lists the value of the SWidth or DWidth bits and their corresponding widths.

Table 6.22 Source or destination transfer width

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

6.18.5 Channel Configuration Registers

The eight Channel Configuration Registers, with address offsets of 0x110, 0x130, 0x150, 0x170, 0x190, 0x1B0, 0x1D0, and 0x1F0 respectively, are read/write and configure the DMA channel. The registers are not updated when a new LLI is requested.

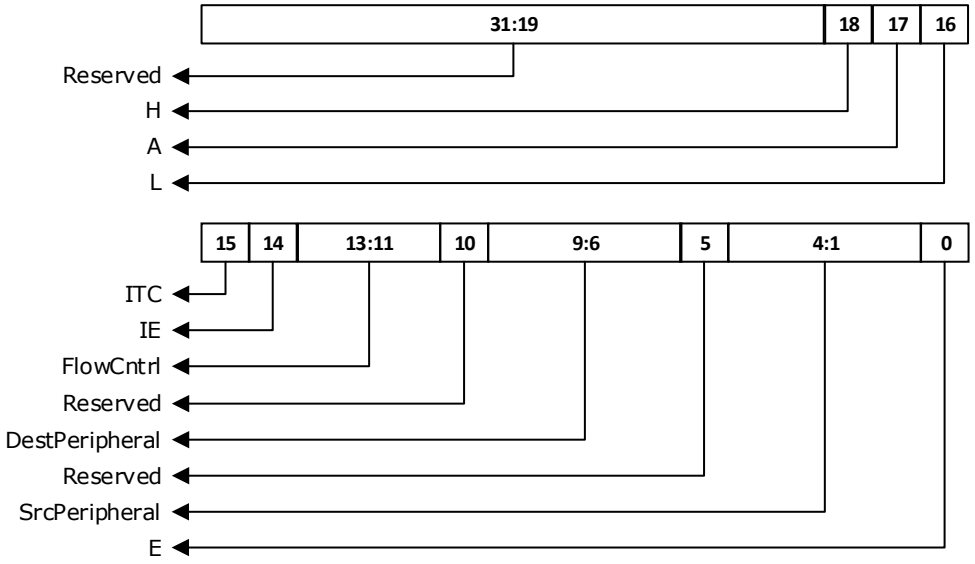

Table 6.23 Software Last Single Request bit assignments

Name	Bit	Reset	Dir	Description
Reserved	31:19	0	N/A	Read undefined. Write as zero.
H	18	0	RW	Halt: 0 = enable DMA requests 1 = ignore extra source DMA requests. The contents of the channels FIFO are drained. You can use this value with the Active and Channel Enable bits to cleanly disable a DMA channel.
A	17	0	R	Active: 0 = there is no data in the FIFO of the channel 1 = the FIFO of the channel has data. You can use this value with the Halt and Channel Enable bits to cleanly disable a DMA channel.
L	16	0	RW	Lock. When set, this bit enables locked transfers. For details of how lock control works, see Locked transfers.
ITC	15	0	RW	Terminal count interrupt mask. When cleared, this bit masks out the terminal count interrupt of the relevant channel.
IE	14	0	RW	Interrupt error mask. When cleared, this bit masks out the error interrupt of the relevant channel.

Name	Bit	Reset	Dir	Description
FlowCntrl	13:11	0	RW	Flow control and transfer type. This value indicates the flow controller and transfer type. The flow controller can be the DMAC, the source peripheral, or the destination peripheral. The transfer type can be memory-to-memory, memory-to-peripheral, peripheral-to-memory, or peripheral-to-peripheral. See *Table 6.25* below for bit value assignments.
Reserved	10	0	N/A	Read undefined. Write as zero.
DestPeripheral (see note below)	9:6	0	RW	Destination peripheral. This value selects the DMA destination request peripheral. This field is ignored if the destination of the transfer is to memory.
Reserved	5	0	N/A	Read undefined. Write as zero.
SrcPeripheral (see note below)	4:1	0	RW	Source peripheral. This value selects the DMA source request peripheral. This field is ignored if the source of the transfer is from memory.
E	0	0	RW	Channel enable. Reading this bit indicates whether a channel is currently enabled or disabled: 0 = channel disabled 1 = channel enabled. You can also determine the Channel Enable bit status by reading the Enabled Channels Register. You enable a channel by setting this bit. You can disable a channel by clearing the Enable bit. This causes the current AHB transfer, if one is in progress, to complete, and the channel is then disabled. Any data in the channel's FIFO is lost. Restarting the channel by setting the Channel Enable bit has unpredictable effects and you must fully re-initialize the channel. The channel is also disabled, and the Channel Enable bit cleared, when the last LLI is reached, or if a channel error is encountered. If a channel has to be disabled without losing data in a channel's FIFO, you must set the Halt bit so that subsequent DMA requests are ignored. The Active bit must then be polled until it reaches 0, indicating that there is no data left in the channel's FIFO. Finally, you can clear the Channel Enable bit.

The DestPeripheral and SrcPeripheral bits are programmed with the binary value of the request line and not a mask value. For example, if the request is number 7, set the register bits to 4'b0111. The ID's of the peripherals can be found in the table below.

Table 6.24 DMA handshake mapping

DMA Request#	Component name	Flow Ctrl
0	I2C TX	
1	I2C RX	
2	SPIO Master TX	
3	SPI0 Master RX	
4	SPI1 Master TX	
5	SPI1 Master RX	
6	SPI2 Slave TX	
7	SPI2 Slave RX	
* 8	ETH_FIFO_RX	√
9	PWM_TX	
10	SDIO_RX/TX	
11	Reserved	
12	UART0 TX	
13	UART0 RX	
14	UART1 TX	
15	UART1 RX	

The table below lists the bit values of the three flow control and transfer type bits.

Table 6.25 Flow control and transfer type bits

Bit value	Transfer type	Controller
0b000	Memory-to-memory	DMA
0b001	Memory-to-peripheral	DMA
0b010	Peripheral-to-memory	DMA
0b011	Source peripheral-to-destination peripheral	DMA
0b100	Source peripheral-to-destination peripheral	N/A
0b101	Memory-to-peripheral	N/A
0b110	Peripheral-to-memory	Peripheral – Ethernet MAC only.
0b111	Source peripheral-to-destination peripheral	Source peripheral – Ethernet MAC only.

6.19 Address generation

Address generation can be either incrementing or non-incrementing.

Note
Address wrapping is not supported.
Bursts do not cross the 1KB address boundary.

6.20 Scatter/gather

Scatter/gather is supported through the use of linked lists. This means that the source and destination areas do not have to occupy contiguous areas in memory. You must set the Channel Linked List Item Register to 0 if you do not require scatter/gather.

6.20.1 Linked list items

An LLI consists of four words. These words are organized in the following order:

1. Channel n Source Address
2. Channel n Destination Address
3. Channel n Linked List Item
4. Channel n Control

Note
The Channel Configuration Register is not part of the LLI.

6.20.2 Programming the DMAC for scatter/gather DMA

To program the DMAC for scatter/gather DMA:

1. Write the LLIs for the complete DMA transfer to memory. Each LLI contains four words:
   - source address
   - destination address
   - pointer to next LLI
   - control word.

   The last LLI has its linked list word pointer set to 0.
2. Choose a free DMA channel with the required priority.
   DMA channel 0 has the highest priority and DMA channel 7 the lowest priority.
3. Write the first LLI, previously written to memory, to the relevant channel in the DMAC.
4. Write the channel configuration information to the channel configuration register and set the Channel Enable bit.
   The DMAC then transfers the first and then subsequent packets of data as each LLI is loaded.
5. An interrupt can be generated at the end of each LLI depending on the Terminal Count bit in the Control Register. If this bit is set, an interrupt is generated at the end of the relevant LLI. You must then service the interrupt request, and you must

set the relevant bit in the Interrupt Terminal Count Clear Register to clear the interrupt.
If so, you must service this interrupt request and you must set the relevant IntTCClear bit in the Interrupt Terminal Count Clear Register to clear the interrupt request interrupt.

6.21 Interrupt requests

Interrupt requests can be generated when an AHB error is encountered, or at the end of a transfer, terminal count, after all the data corresponding to the current LLI has been transferred to the destination. The interrupts can be masked by programming the relevant bits on the relevant Channel Control and Channel Configuration Registers.

Interrupt Status Registers are provided. They group the interrupt requests from all the DMA channels prior to interrupt masking, DMACRawIntTCStatus, DMACRawIntErrorStatus, and after interrupt masking, DMACIntTCStatus, DMACIntErrorStatus.

The Interrupt Status Register combines both the DMACIntTCStatus and DMACIntErrorStatus requests into a single register to enable the source of an interrupt to be found quickly. Writing to the Interrupt Terminal Count Clear or the Interrupt Error Clear Registers with a bit set HIGH enables selective clearing of interrupts.

The DMAC uses separate interrupt requests for the error and transfer complete requests. Read either the Interrupt Terminal Count Status or Interrupt Error Status Registers to find the source of an interrupt.

6.21.1 Terminal count interrupt sequence flow

1. You must wait until the terminal count DMA interrupt request goes active.
   Assuming the interrupt is enabled in the interrupt controller and in the processor, the processor branches to the interrupt vector address and enters the interrupt service routine.
2. You must read the interrupt controller Status Register to determine if the source of the interrupt request was the DMAC asserting the DMACINTTC signal.
3. You must read the Interrupt Terminal Count Status Register to determine the channel that generated the interrupt.
   If more than one request is active, it is recommended that you service the highest priority channel first.
4. You must service the interrupt request.
5. You must write a 1 to the relevant bit in the Interrupt Terminal Count Clear Register to clear the interrupt request.

6.21.2 Error interrupt sequence flow

1. You must wait until the interrupt request goes active because of a DMA channel error.
   Assuming the interrupt is enabled in the interrupt controller and in the processor, the processor branches to the interrupt vector address and enters the interrupt service routine.

2. You must read the Interrupt Controllers Status Register to determine if the source of the request was the DMAC asserting the DMACINTERR signal.
3. You must read the Interrupt Error Status Register to determine the channel that generated the interrupt.
   If more than one request is active it is recommended that you check the highest priority channels first.
4. You must service the interrupt request.
5. You must write a 1 to the relevant bit in the Interrupt Error Clear Register to clear the interrupt request.

6.21.3 3.6.4 Interrupt polling sequence flow

The DMAC interrupt request signal is masked out, disabled in the interrupt controller, or disabled in the processor. When polling the DMAC, you must:

1. Read the Interrupt Status Register.
   If none of the bits are HIGH repeat this step, otherwise, go to step 2.
   If more than one request is active, it is recommended that you check the highest priority channels first.
2. Read the Interrupt Terminal Count Status Register to determine if the interrupt was generated because of the end of the transfer, terminal count, or because of error occurred.
   A HIGH bit indicates that the transfer completed.
3. Service the interrupt request.
4. For an error interrupt, write a 1 to the relevant bit of the Interrupt Error Clear Register to clear the interrupt request.
   For a terminal count interrupt, write a 1 to the relevant bit of the Interrupt Terminal Count Clear Register.

6.22 DMAC data flow

This section describes the DMAC data flow sequences for:

- Memory-to-memory DMA flow
- Memory-to-peripheral, or peripheral-to-memory DMA flow
- Peripheral-to-peripheral DMA flow

6.22.1 Memory-to-memory DMA flow

For a memory-to-memory DMA flow:

1. Program and enable the DMA channel.
2. Transfer data whenever the DMA channel has the highest pending priority and the DMAC gains bus master ship of the AHB bus.
3. If an error occurs while transferring the data, generate an error interrupt and disable the DMA stream.
4. Decrement the transfer count.
5. If the count has reached zero:
   a. Generate a terminal count interrupt. You can mask the interrupt.

b. If the Channel Linked List Item Register is not 0, then reload the following registers and go to back to step 2:
Channel Source Address
Channel Destination Address
Channel Linked List Item
Channel Control
However, if the Channel Linked List Item Register is 0, the DMA stream is disabled and the flow sequence ends.

6.22.2 Memory-to-peripheral, or peripheral-to-memory DMA flow

For a peripheral-to-memory or memory-to-peripheral DMA flow:

1. Program and enable the DMA channel.
2. Wait for a DMA request.
3. The DMAC then starts transferring data when:
   a. The DMA request goes active.
   b. The DMA stream has the highest pending priority.
   c. The DMAC is the bus master of the AHB bus.
4. If an error occurs while transferring the data, an error interrupt is generated and the DMA stream is disabled, and the flow sequence ends.
5. Decrement the transfer count if the DMAC is controlling the flow control.
6. If the transfer has completed, indicated by the transfer count reaching 0 if the DMAC is performing flow control, or by the peripheral setting the DMACLBREQ or DMACLSREQ signals if the peripheral is performing flow control:
   a. The DMAC asserts the DMACTC signal.
   b. The terminal count interrupt is generated. You can mask this interrupt.
   c. If the Channel Linked List Item Register is not 0, then reload the following registers and go to back to step 2:
   Channel Source Address
   Channel Destination Address
   Channel Linked List Item
   Channel Control
   However, if the Channel Linked List Item Register is 0, the DMA stream is disabled and the flow sequence ends.

6.22.3 Peripheral-to-peripheral DMA flow

For a peripheral-to-peripheral DMA flow:

1. Program and enable the DMA channel.
2. Wait for a source DMA request.
3. The DMAC then starts transferring data when:
   a. The DMA request goes active.
   b. The DMA stream has the highest pending priority.
   c. The DMAC is the bus master of the AHB bus.
1. If an error occurs while transferring the data, an error interrupt is generated, then finishes.
2. Decrement the transfer count if the DMAC is controlling the flow control.

3. If the transfer has completed, indicated by the transfer count reaching 0 if the DMAC is performing flow control, or by the peripheral setting the DMACLBREQ or DMACLSREQ signals if the peripheral is performing flow control:
   a. The DMAC asserts the DMACTC signal to the source peripheral.
   b. Subsequent source DMA requests are ignored.
   c. When the destination DMA request goes active and there is data in the DMAC FIFO, transfer data into the destination peripheral.

1. If an error occurs while transferring the data, an error interrupt is generated and the DMA stream is disabled, and the flow sequence ends.
2. If the transfer has completed, it is indicated by the transfer count reaching 0 if the DMAC is performing flow control, or by the peripheral setting the DMACLBREQ or DMACLSREQ signals if the peripheral is performing flow control. The following happens:
   a. The DMAC asserts the DMACTC signal to the destination peripheral.
   b. The terminal count interrupt is generated. You can mask this interrupt.
   c. If the Channel Linked List Item Register is not 0, then reload the following registers and go to back to step 2:
      Channel Source Address
      Channel Destination Address
      Channel Linked List Item
      Channel Control
      However, if Channel Linked List Item Register is 0, the DMA stream is disabled and the flow sequence ends.

6.23 Revisions

Table 6.26 Document revision history

Date	Rev.	By	Change
Sept. 11, 2015	1.0.0-41621	BK	Initial publication