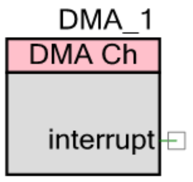
PSoC® Creator™ Component Datasheet

Direct Memory Access (DMA_PDL)

1.0



Features

- Devices support up to two DMA hardware blocks
- Each DMA block supports up to 16 DMA channels
- Supports channel descriptors in SRAM
- 4 Priority Levels for each channel
- Byte, Halfword (2 bytes), and Word (4 bytes) transfers
- Configurable source and destination addresses
- Four Transfer Modes:
- Single Data Element per Trigger
- One X (inner) loop transfer per trigger
- Entire descriptor per trigger
- Entire descriptor chain per trigger
- Configurable output trigger
- Configurable interrupt generation

General Description

The DMA Component transfers data to and from memory, Components, and registers. These transfers occur independent of the CPU. The DMA transfers can be set up in a byte, halfword (2 bytes), or word (4 bytes) wide. The DMA starts each transaction through an external trigger that can come from a DMA channel (including itself), another DMA channel, a peripheral, or the CPU. The DMA is best used to offload data transfer tasks from the CPU

When to Use a DMA

The DMA Channel Component can be used in any project that needs to transfer data without CPU intervention based on a hardware trigger signal from another Component.

A common use is transferring data from memory to a peripheral, such as a UART. The DMA can be triggered by the UART FIFO not full signal. The DMA will load data in the UART until the FIFO fills.

The DMA can also be used to take data out of the UART and place it in memory. For example, the DMA can be triggered by the FIFO not empty signal, thus the DMA will transfer data as long as the FIFO is not empty.

Another common use is transferring data from the ADC to memory. The ADC's end of conversion (eoc) signal can be used to trigger the DMA to transfer the ADC result to memory.

The DMA can also be used to move blocks of memory from one memory location to another (RAM to RAM, FLASH to RAM); DMA cannot write to FLASH.

Each DMA channel can be triggered by a hardware signal as described above, or by a firmware register write, or both.

Each DMA channel can be associated to a descriptor. User can create multiple descriptors and have them chained to each other. The user can also reconfigure a descriptor while it is not the active descriptor in the DMA.

Input/Output Connections

This section describes the various input and output connections for the DMA Component. An asterisk (*) in the following list indicates that it may not be shown on the Component symbol for the conditions listed in the description of that I/O.

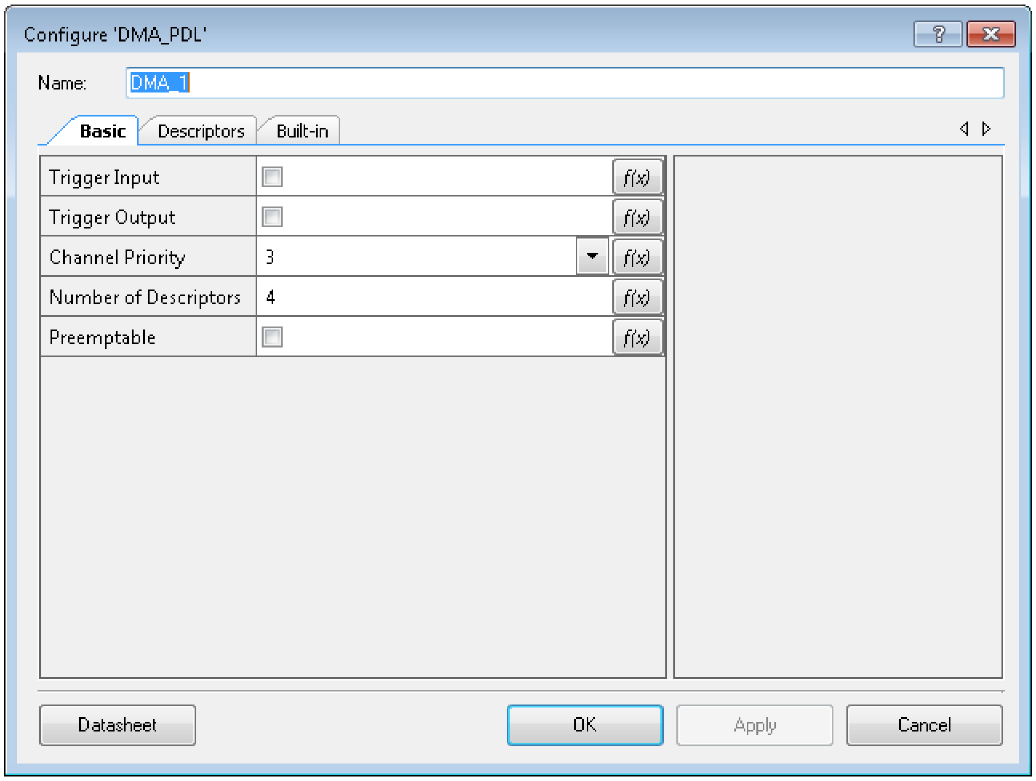
Terminal Name	I/O Type	Description
tr_in*	Digital Input	Visible only if ‘Trigger input’ is set to True. Input trigger parameter sets up the trigger input signal for the DMA Component.
tr_out*	Digital Output	Visible only if ‘Trigger output’ is set to True. Output trigger parameter sets up the trigger output signal for the DMA Component.
interrupt	Digital Output	Output terminal that allows to connect the interrupt Component.

Component Parameters

Drag a DMA Component onto your design and double click it to open the Configure dialog. This dialog has the following tabs with different parameters.

Basic Tab

This tab contains DMA channel settings and number of descriptors.



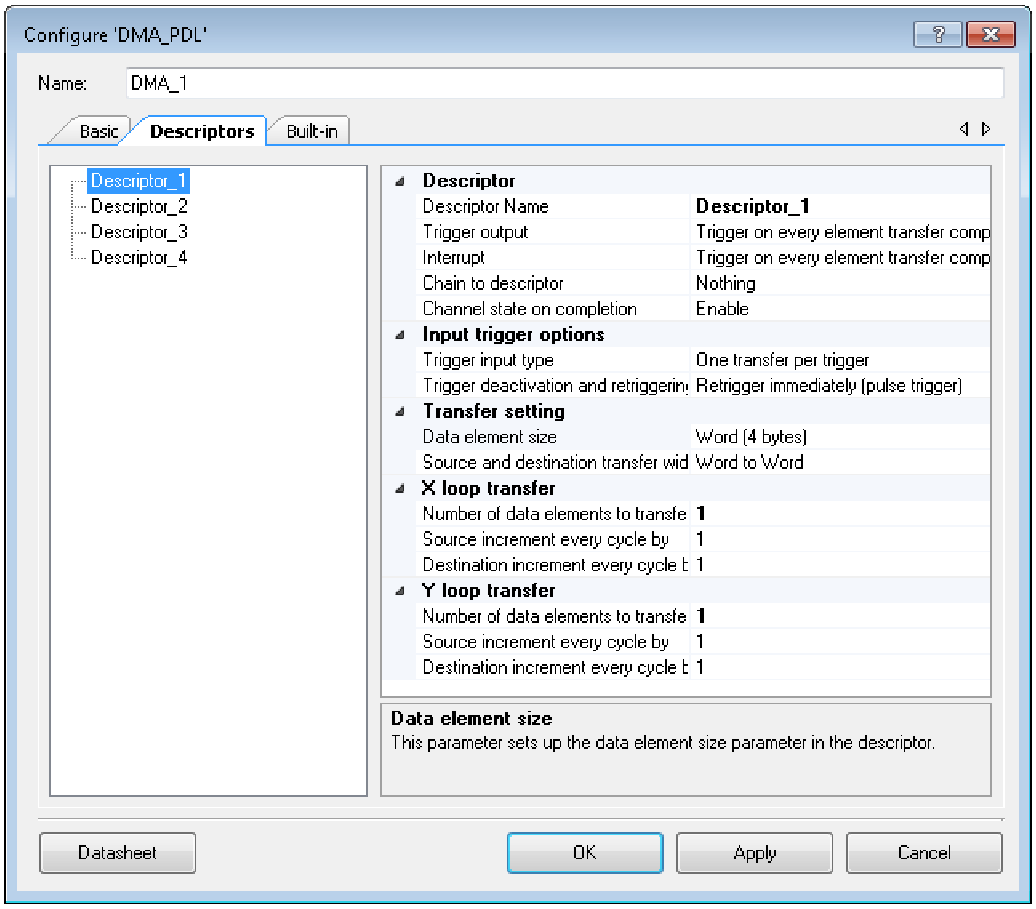
Name: DMA_1

Trigger Input	☐
Trigger Output	☐
Channel Priority	3
Number of Descriptors	4
Preemptable	☐

Parameter Name	Description
Trigger input	Input trigger parameter sets up the trigger input signal for the DMA Component.
Trigger Output	Output trigger parameter sets up the trigger output signal for the DMA Component.
Channel priority	Priority of the channel in the DMA block.
Number of Descriptors	The number of descriptors sets up rest of the customizer. Based on the number set in this parameter as many instances of Descriptors are created in the customizer.
Preemtable	Preemptable setting when enabled makes the channel preemptable by another higher priority channel.

Descriptors Tab

This tab contains the Descriptors configuration settings. Same set of parameters are created for each of descriptors.



Name: DMA_1

Descriptor_1
Descriptor_2
Descriptor_3
Descriptor_4

Descriptor	
Descriptor Name	**Descriptor_1**
Trigger output	Trigger on every element transfer comp
Interrupt	Trigger on every element transfer comp
Chain to descriptor	Nothing
Channel state on completion	Enable
Input trigger options	
Trigger input type	One transfer per trigger
Trigger deactivation and retriggerin	Retrigger immediately (pulse trigger)
Transfer setting	
Data element size	Word (4 bytes)
Source and destination transfer wid	Word to Word
X loop transfer	
Number of data elements to transfe	**1**
Source increment every cycle by	1
Destination increment every cycle b	1
Y loop transfer	
Number of data elements to transfe	**1**
Source increment every cycle by	1
Destination increment every cycle b	1

Data element size
This parameter sets up the data element size parameter in the descriptor.

Parameter Name	Description
Descriptor	
Descriptor Name	This parameter sets up the instance name for the Descriptor.
Trigger output	This is the selection for what event would trigger the DMA output.
Interrupt	This is the selection for what event would trigger the DMA interrupt.
Channel state on completion	This is the state of the channel when the descriptor is completed.
Chain to descriptor	This parameter allows to configure what will be executed after current descriptor.
Input trigger options	
Trigger input type	Trigger input type will set up the character of each trigger.
Trigger deactivation and retriggering	This parameter sets up the trigger deactivation options for the descriptor.

Transfer settings	
Data element size	This parameter sets up the data element size parameter in the descriptor.
Source and destination width	This sets up the source and destination transfer size parameters.
X loop transfer	
Number of data elements to transfer	This parameter configures how many transfers are effected in the X loops.
Source increment every cycle by	This integer determines the source address increment after each transfer. When this value is set to zero, it is equivalent to disabling the source increment feature.
Destination increment every cycle by	This integer determines the destination address increment after each transfer. When this value is set to zero, it is equivalent to disabling the destination increment feature.
Y loop transfer	
Number of data elements to transfer	This parameter configures how many transfers are effected in the Y loops.
Source increment every cycle by	This integer determines the source address increment after each transfer. When this value is set to zero, it is equivalent to disabling the source increment feature.
Destination increment every cycle by	This integer determines the destination address increment after each transfer. When this value is set to zero, it is equivalent to disabling the destination increment feature.

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the Component using software. The following sections list and describe Component functions and dependencies.

This Component uses firmware drivers from the cy_dma Peripheral Driver Library (PDL) module, which is automatically added to your project after a successful build in the pdl3.0/cy_dma folder. Pass the generated data structures to the associated PDL functions in your application initialization code to configure the peripheral.

Functions

API functions allow configuration of the Component using the CPU.

By default, PSoC Creator assigns the instance name "DMA_1" to the first instance of a Component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol.

Function	Description
DMA_1_Start	Based on the settings for descriptor in the customizer this function runs the DMA_Descr_Init() and then initializes the channel using DMA_Chnl_Init().

Function	Description
DMA_1_Init	Based on the settings for descriptor in the customizer this function runs the DMA_Descr_Init() and then initializes the channel using DMA_Chnl_Init().
DMA_1_Stop	Disables the channel and clears any DMA transfers pending in the channel.
DMA_1_ChEnable	Enables DMA_1 channel.
DMA_1_ChDisable	Disables DMA_1 channel.
DMA_1_SetDescriptor	Sets a Descriptor as the current descriptor for the DMA_1 channel.
DMA_1_GetDescriptor	Returns the pointer to the structure that is the descriptor for the ongoing transfer in the DMA_1.
DMA_1_GetNextDescriptor	Returns the next Descriptor for the current descriptor of the DMA channel.
DMA_1_SetNextDescriptor	Sets the next Descriptor for the current descriptor of the DMA channel.
DMA_1_SetPriority	Sets priority for the channel.
DMA_1_GetPriority	Gets the priority of the channel.
DMA_1_GetSrcDstTransferWidth	Gets the transfer width for source and destination transfer widths of the Descriptor
DMA_1_SetSrcDstTransferWidth	Sets the transfer width for source and destination transfer widths of the Descriptor
DMA_1_SetSrcAddress	Sets source address for the channel.
DMA_1_GetSrcAddress	Gets source address for the channel.
DMA_1_SetDataElementSize	Sets the data element size for the Descriptor.
DMA_1_GetDataElementSize	Gets the data element size for the Descriptor.
DMA_1_SetXloopNumDataElements	Sets the number of data elements for the X loop of the descriptor
DMA_1_GetXloopNumDataElements	Gets the number of data elements for the X loop of the descriptor
DMA_1_SetXloopSrcIncrement	Sets the source increment parameter for the X loop of the descriptor
DMA_1_GetXloopSrcIncrement	Gets the source increment parameter for the X loop of the descriptor
DMA_1_SetXloopDstIncrement	Sets the destination increment parameter for the X loop of the descriptor
DMA_1_GetXloopDstIncrement	Sets the destination increment parameter for the X loop of the descriptor
DMA_1_SetYloopNumDataElements	Sets the number of data elements for the Y loop of the descriptor
DMA_1_GetYloopNumDataElements	Gets the number of data elements for the Y loop of the descriptor
DMA_1_SetYloopSrcIncrement	Sets the source increment parameter for the Y loop of the descriptor
DMA_1_GetYloopSrcIncrement	Gets the source increment parameter for the Y loop of the descriptor
DMA_1_SetYloopDstIncrement	Sets the destination increment parameter for the Y loop of the descriptor

Function	Description
DMA_1_GetYloopDstIncrement	Sets the destination increment parameter for the Y loop of the descriptor
DMA_1_GetInterruptStatus	This function can be used to determine the interrupt status of the DMA channel.
DMA_1_GetInterruptCause	This function can be used to determine the interrupt reason of the DMA channel.
DMA_1_ClearInterrupt	This function clears the interrupt status.
DMA_1_SetInterrupt	This function sets the interrupt.
DMA_1_GetInterruptMask	This function gets interrupt mask value.
DMA_1_SetInterruptMask	This function sets interrupt mask value.
DMA_1_GetInterruptStatusMasked	This function returns logical and of corresponding INTR and INTR_MASK fields in a single load operation.

void DMA_1_Start(void)

Description: Based on the settings for descriptor in the customizer this function runs the DMA_Descr_Init() and then initializes the channel using DMA_Chnl_Init(). Enables the DMA_1 block using the DMA_Chnl_Enable().

void DMA_1_Init(void)

Description: Based on the settings for descriptor in the customizer this function runs the DMA_Descr_Init() and then initializes the channel using DMA_Chnl_Init().

void DMA_1_Stop(void)

Description: Disables the channel and clears any DMA transfers pending in the channel.

void DMA_1_ChEnable(void)

Description: Enables DMA_1 channel.

void DMA_1_ChDisable(void)

Description: Disables DMA_1 channel.

void DMA_1_SetDescriptor(cy_stc_dma_descr_t* descriptor)

Description: Sets a Descriptor as the current descriptor for the DMA_1 channel.

Parameters: descriptor - Pointer to a Descriptor structure.

cy_stc_dma_descr_t* DMA_1_GetDescriptor(void)

Description: Returns the pointer to the structure that is the descriptor for the ongoing transfer in the DMA_1. DMA_1 channel is disabled while this action is processed.

Return Value: Pointer to a Descriptor structure.

void DMA_1_SetNextDescriptor(cy_stc_dma_descr_t* descriptor)

Description: Sets the next Descriptor for the current descriptor of the DMA channel. The DMA channel is disabled while this action is being processed.

Parameters: descriptor - Pointer to a Descriptor structure.

cy_stc_dma_descr_t* DMA_1_GetNextDescriptor(void)

Description: Gets the next Descriptor for the current descriptor of the DMA channel. The DMA channel is disabled while this action is being processed.

Return Value: Pointer to a Descriptor structure.

void DMA_1_SetPriority(uint32_t Priority)

Description: Sets priority for the channel.

Parameters: Priority – Priority of the channel

uint32_t DMA_1_GetPriority(void)

Description: Gets the priority of the channel.

Return Value: Priority of the channel.

void DMA_1_SetSrcDstTransferWidth(cy_stc_dma_descr_t* descriptor, uint32_t options)

Description: Sets the transfer width for source and destination transfer widths of the Descriptor

Parameters: descriptor - Descriptor for which the data element size is set

options - Macros set up for the four options

uint32_t DMA_1_GetSrcDstTransferWidth(cy_stc_dma_descr_t* descriptor)

Description: Gets the transfer width for source and destination transfer widths of the Descriptor

Parameters: Descriptor - Descriptor for which the data element size is set

Return Value: Macros set up for the four options

void DMA_1_SetSrcAddress(cy_stc_dma_descr_t* descriptor, uint32_t* SrcAddress)

Description: Sets source address for the channel.

Parameters: descriptor – Descriptor pointer

SrcAddress – Source address

uint32_t* DMA_1_GetSrcAddress(cy_stc_dma_descr_t* descriptor)

Description: Gets source address for the channel.

Parameters: descriptor - Descriptor pointer.

Return Value: Source address of the channel.

void DMA_1_SetDataElementSize(cy_stc_dma_descr_t* descriptor, uint32_t DataSize)

Description: Sets the data element size for the Descriptor.

Parameters: descriptor - Descriptor for which the data element size is set

DataSize - Data element size

Constant	Descritpion
DMA_BYTE	One byte
DMA_HALFWORD	Half word (two bytes)
DMA_WORD	Full word (four bytes)

uint32_t DMA_1_GetDataElementSize(cy_stc_dma_descr_t* descriptor)

Description: Gets the data element size for the Descriptor

Parameters: Descriptor - Descriptor for which the data element size is set

Return Value: Data element size

void DMA_1_SetXloopNumDataElements (cy_stc_dma_descr_t* descriptor, uint32_t number)

Description: Sets the number of data elements for the X loop of the descriptor.

Parameters: descriptor - Descriptor for which the data element size is set

number - Number of data elements

uint32_t DMA_1_GetXloopNumDataElements(cy_stc_dma_descr_t* descriptor)

Description: Gets the number of data elements for the X loop of the descriptor.

Parameters: descriptor - Descriptor for which the data element size is set

Return Value: Number of data elements

void DMA_1_SetXloopSrcIncrement(cy_stc_dma_descr_t* descriptor, uint32_t increment)

Description: Sets the source increment parameter for the X loop of the descriptor.

Parameters: descriptor - Descriptor for which the data element size is set

increment - value of the source increment

uint32_t DMA_1_GetXloopSrcIncrement(cy_stc_dma_descr_t* descriptor)

Description: Gets the source increment parameter for the X loop of the descriptor

Parameters: descriptor
* Descriptor for which the data element size is set

Return Value: value of the source increment

void DMA_1_SetXloopDstIncrement(cy_stc_dma_descr_t* descriptor, uint32_t increment)

Description: Sets the destination increment parameter for the X loop of the descriptor.

Parameters: descriptor - Descriptor for which the data element size is set

increment - value of the destination increment

uint32_t DMA_1_GetXloopDstIncrement(cy_stc_dma_descr_t* descriptor)

Description: Gets the destination increment parameter for the X loop of the descriptor

Parameters: descriptor - Descriptor for which the data element size is set

Return Value: value of the destination increment

void DMA_1_SetYloopNumDataElements (cy_stc_dma_descr_t* descriptor, uint32_t number)

Description: Sets the number of data elements for the Y loop of the descriptor.

Parameters: descriptor - Descriptor for which the data element size is set

number - Number of data elements

uint32_t DMA_1_GetYloopNumDataElements(cy_stc_dma_descr_t* descriptor)

Description: Gets the number of data elements for the Y loop of the descriptor.

Parameters: descriptor - Descriptor for which the data element size is set

Return Value: Number of data elements

void DMA_1_SetYloopSrcIncrement(cy_stc_dma_descr_t* descriptor, uint32_t increment)

Description: Sets the source increment parameter for the Y loop of the descriptor.

Parameters: descriptor - Descriptor for which the data element size is set

increment - value of the source increment

uint32_t DMA_1_GetYloopSrcIncrement(cy_stc_dma_descr_t* descriptor)

Description: Gets the source increment parameter for the Y loop of the descriptor

Parameters: descriptor - Descriptor for which the data element size is set

Return Value: value of the source increment

void DMA_1_SeYloopDstIncrement(cy_stc_dma_descr_t* descriptor, uint32_t increment)

Description: Sets the destination increment parameter for the Y loop of the descriptor.

Parameters: descriptor - Descriptor for which the data element size is set

increment - value of the destination increment

uint32_t DMA_1_GetYloopDstIncrement(cy_stc_dma_descr_t* descriptor)

Description: Gets the destination increment parameter for the Y loop of the descriptor

Parameters: descriptor - Descriptor for which the data element size is set

Return Value: value of the destination increment

uint32_t DMA_1_GetInterruptCause(void)

Description: This function can be used to determine the interrupt reason of the DMA channel.

Return Value: Returns a status of an interrupt for specified channel.

Constant	Descritpion
CY_DMA_INTRCAUSE_NO_INTR	No interrupt
CY_DMA_INTRCAUSE_COMPLETION	Completion
CY_DMA_INTRCAUSE_SRC_BUS_ERROR	Source bus error
CY_DMA_INTRCAUSE_DST_BUS_ERROR	Destination bus error
CY_DMA_INTRCAUSE_SRC_MISAL	Source address is not aligned
CY_DMA_INTRCAUSE_DST_MISAL	Destination address is not aligned
CY_DMA_INTRCAUSE_CURR_PTR_NULL	Current descriptor pointer is NULL
CY_DMA_INTRCAUSE_ACTIVE_CH_DISABLED	Active channel is disabled
CY_DMA_INTRCAUSE_DESCR_BUS_ERROR	Descriptor bus error

void DMA_1_ClearInterrupt(void)

Description: This function clears the interrupt status.

void DMA_1_SetInterrupt(void)

Description: This function sets the interrupt.

uint32_t DMA_1_GetInterruptMask(void)

Description: This function gets interrupt mask value.

Return Value: Returns an interrupt mask value.

void DMA_1_SetInterruptMask(void)

Description: This function sets interrupt mask value.

uint32_t DMA_1_GetInterruptStatusMasked(void)

Description: This function returns logical and of corresponding INTR and INTR_MASK fields in a single load operation.

Return Value: logical and of corresponding INTR and INTR_MASK fields

Global Variables

Variable	Description
DMA_1_initVar(static)	The initVar variable is used to indicate initial configuration of this Component. This variable is prepended with the Component name. The variable is initialized to zero and set to 1 the first time DMA_1_Start() is called. This allows for Component initialization without reinitialization in all subsequent calls to the DMA_1_Start() routine.
DMA_1_Descriptor_X	Each descriptor is created in RAM and is initialized after DMA_1_Init() or DMA_1_Start() are called.
DMA_1_Descriptor_X_SETTINGS	Stores descriptor settings that are used by DMA_1_Init() or DMA_1_Start() to initialize descriptor.

Interrupt Service Routine

Each instance of a Component can have an interrupt. To enable and configure it:

1. Drag the Interrupt Component to TopDesign and connect it to the DMA Component terminal called Interrupt.
2. Add DMA_1_SetInterruptMask() function call to enable interrupt from DMA_1 Component.
3. Optionally, you can have DMA_1_GetInterruptCause() function call in the interrupt handler to get the interrupt cause.

API Memory Usage

The Component is designed to use API from the dma Peripheral Driver Library (PDL) module. That is why the Component itself only consumes resources necessary to allocate structures for driver operation and start the Component.

Functional Description

The DMA Component represents a DMA channel that can have up to 4096 descriptors that are pre-configured in the customizer. Each descriptor can be chained to other descriptor. That means that on descriptor completion DMA channel will automatically be switched to the next descriptor. Depending on Trigger input type configured in customizer DMA Component can wait for the next trigger or execute the next descriptor.

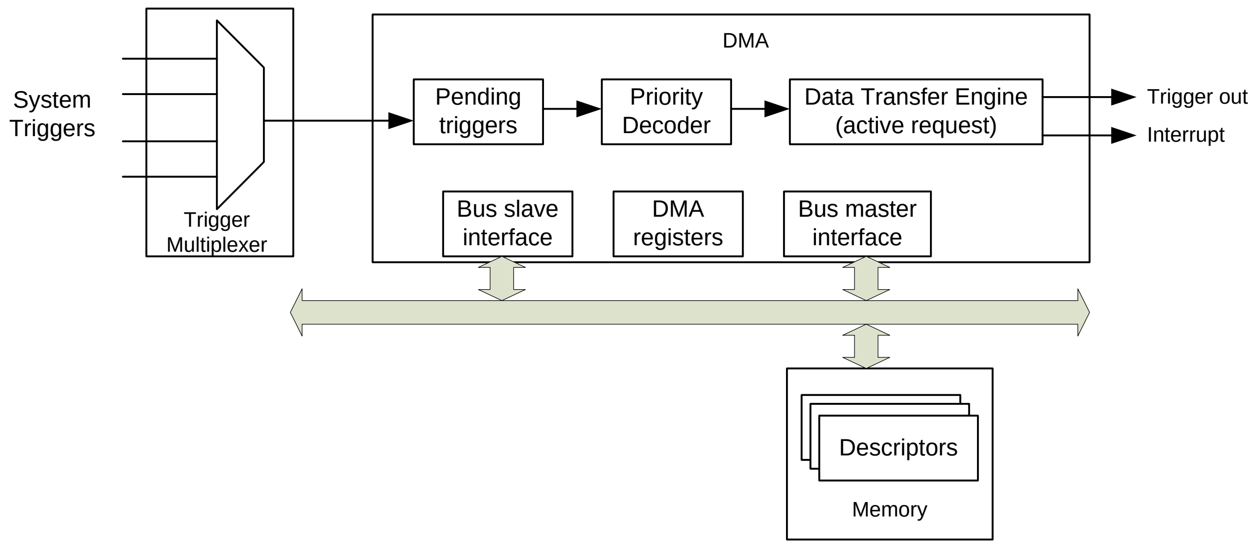
Definitions

- DMA: Direct memory access. In this document the term DMA refers to the DMA block in MXS40 platform.
- DMA Component: This term refers to the DMA Component in PSoC Creator that configures a DMA channel in MXS40 platform.

- CyDMA peripheral driver: Peripheral driver for the DMA block.
- Descriptor: Descriptor that sets up the transfer parameters for a DMA channel transfer. The descriptor is initialized in SRAM and reference using a pointer in the DMA channel. Multiple descriptors can be chained.

Block Diagram and Configuration

The following is a simplified diagram of the DMA hardware:



DMA controller supports multiple independent data transfers that are managed by a channel. Each channel connects to a specific system trigger through a trigger multiplexer that is outside the DMA controller.

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the Component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator Components
- specific deviations – deviations that are applicable only for this Component

This section provides information on Component-specific deviations. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The DMA Component has the following specific deviations:

Rule	Rule Class	Rule Description	Description of Deviation(s)
1.1	R	The keyword 'inline' has been used.	Deviated since INLINE functions are used to allow more efficient code.
3.1, 11.3	A	Cast between a pointer and an integral type.	The cast from unsigned int to pointer does not have any unintended effect, as it is a consequence of the definition of a structure based on hardware registers.
21.1	R	This operation is redundant. The value of the result is always that of the left-hand operand.	Deviated since _CLR_SET_FLD32U macros are used to allow more efficient code.

Registers

See the chip *Technical Reference Manual (TRM)* for more information about the registers.

Resources

The Component uses one DMA channel.

DC and AC Electrical Characteristics

Refer to Digital Peripherals in the Electrical Specifications section of the Device Family Datasheet.

Component Changes

This section lists the major changes in the Component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.0.a	Updated datasheet	Changed several API descriptions.
1.0	First Component version	