32-bit RISC Microcontroller

TXZ Family

Reference Manual
Multi-Function DMA Controller (MDMAC-A)

Revision 2.0

2018-03

TOSHIBA ELECTRONIC DEVICES & STORAGE CORPORATION

Contents

List of Figures

List of Tables

Preface

Related document

Document name
Product Information
Exception
Clock Control and Operation Mode
Asynchronous Serial Communication Circuit
Full Universal Asynchronous Receiver Transmitter Circuit
Serial Peripheral Interface

Conventions

- Numeric formats follow the rules as shown below:
  Hexadecimal: 0xABC
  Decimal: 123 or 0d123 – Only when it needs to be explicitly shown that they are decimal numbers.
  Binary: 0b111 – It is possible to omit the "0b" when the number of bit can be distinctly understood from a sentence.
- "_N" is added to the end of signal names to indicate low active signals.
- It is called "assert" that a signal moves to its active level, "deassert" to its inactive level.
- When two or more signal names are referred, they are described like as [m: n].
  Example: S[3: 0] shows four signal names S3, S2, S1 and S0 together.
- The characters surrounded by ***[ ]*** defines the register.
  Example: ***[ABCD]***
- "n" substitutes suffix number of two or more same kind of registers, fields, and bit names.
  Example: ***[XYZ1], [XYZ2], [XYZ3] → [XYZn]***
- "x" substitutes suffix number or character of units and channels in the Register List.
  In case of unit, "x" means A, B, and C . . .
  Example: ***[ADACR0], [ADBCR0], [ADCCR0] → [ADxCR0]***
  In case of channel, "x" means 0, 1, and 2 . . .
  Example: [***T32A0RUNA], [T32A1RUNA], [T32A2RUNA] → [T32AxRUNA]***
- The bit range of a register is written like as [m: n].
  Example: Bit[3: 0] expresses the range of bit 3 to 0.
- The configuration value of a register is expressed by either the hexadecimal number or the binary number.
  Example: ***[ABCD]***<EFG> = 0x01 (hexadecimal), ***[XYZn]***<VW> = 1 (binary)
- Word and Byte represent the following bit length.
  Byte: 8 bits
  Half word: 16 bits
  Word: 32 bits
  Double word: 64 bits
- Properties of each bit in a register are expressed as follows:
  R: Read only
  W: Write only
  R/W: Read and Write are possible
- Unless otherwise specified, register access supports only word access.
- The register defined as reserved must not be rewritten. Moreover, do not use the read value.
- The value read from the bit having default value of "-" is unknown.
- When a register containing both of writable bits and read-only bits is written, read-only bits should be written with their default value, In the cases that default is "-", follow the definition of each register.
- Reserved bits of the Write-only register should be written with their default value. In the cases that default is "-", follow the definition of each register.
- Do not use read-modified-write processing to the register of a definition which is different by writing and read out.

The Flash memory uses the Super Flash® technology under the license of Silicon Storage Technology, Inc.
Super Flash® is registered trademark of Silicon Storage Technology, Inc.

All other company names, product names, and service names mentioned herein may be trademarks of their respective companies.

Terms and Abbreviation

Some of abbreviations used in this document are as follows:

DMA	Direct Memory Access
DMAC	Direct Memory Access Controller
FUART	Full Universal Asynchronous Receiver Transmitter
MDMAC	Multi function DMAC
TSPI	Toshiba Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter

1. Outlines

The main functions of MDMAC per unit are shown in the following table.

Table 1.1 MDMAC function (per unit)

Function category	Function	Description
Transfer factor and Transfer type	Start-up factor	DMA can be started up by a peripheral function or software.
	Unit transfer	A specified size data is transferred. And MDMAC waits for the next transfer request.
	Continuous transfer	All data are transferred.
Transfer mode	Normal transfer	Normal transfer has the following two modes. - Unit normal transfer: Unit transfer is done by one transfer request. - Continuous normal transfer: Continuous transfer is done by one transfer request.
	Chain transfer	Chain transfer executes data transfer according to the information of a descriptor. The mixture of a unit transfer and a continuous transfer is available.
Channel control	Transfer address	Setting of a transfer source address and a transfer destination address
	Transfer data size	1/2/4/8/16/32 Bytes (Unit size)
	Priority	The priority at execution is determined by a channel number and a two-stage priority setting.
	Arbitration	Arbitration is done per transfer of a unit size data.
Transfer count	Normal transfer	Maximum 1,048,575 Bytes (finite) or infinite
	Chain transfer	Infinite (controlled by a descriptor)
Interrupt	Transfer completion interrupt	When data transfer completes, Transfer completion interrupt is generated.
	Error interrupt	When a bus error or a descriptor error occurs, Error interrupt is generated.

2. Configuration

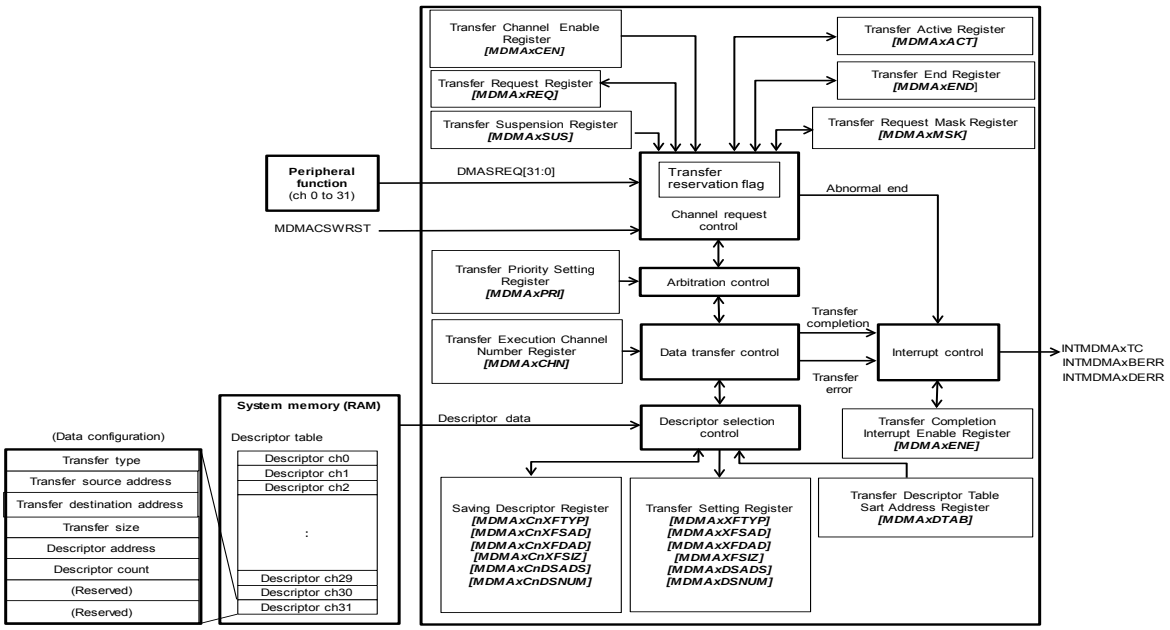


Figure 2.1 Block diagram of MDMAC (common to units)

Table 2.1 List of signals

No.	Symbol	Signal name	I/O	Reference manual
1	INTMDMAxTC	MDMAC transfer completion interrupt	Output	Exception
2	INTMDMAxDERR	MDMAC descriptor error interrupt	Output	Exception
3	INTMDMAxBERR	MDMAC bus error interrupt	Output	Exception
4	DMASREQ[31:0]	Single transfer request	Input	Product Information
5	MDMACSWRST	Reset/Error clear	Input	Clock control and operation mode
6	-	Descriptor data	Input	This document

3. Function and Operation

Multi-function DMA controller (MDMAC) is controlled by registers and descriptor in RAM. The descriptor information is located in a memory. For the data configuration of descriptor, refer to “3.3Descriptor”.

3.1. Clock Supply

When MDMAC is used, the corresponding clock enable bits should be set to “1” (Clock supply) in fsys supply stop register A (***[CGFSYSENA]*** and ***[CGFSYSMENA]***), fsys supply stop register B (***[CGFSYSENB]*** and ***[CGFSYSMENB]***), and fc supply stop register (***[CGFCEN]***). The corresponding registers and the bit locations depend on a product. Some products do not have all registers. For the details, refer to “Clock Control and Operation Mode” in Reference manual.

3.2. Operation Outlines

Figure 3.1 shows the operation flowchart of MDMAC.

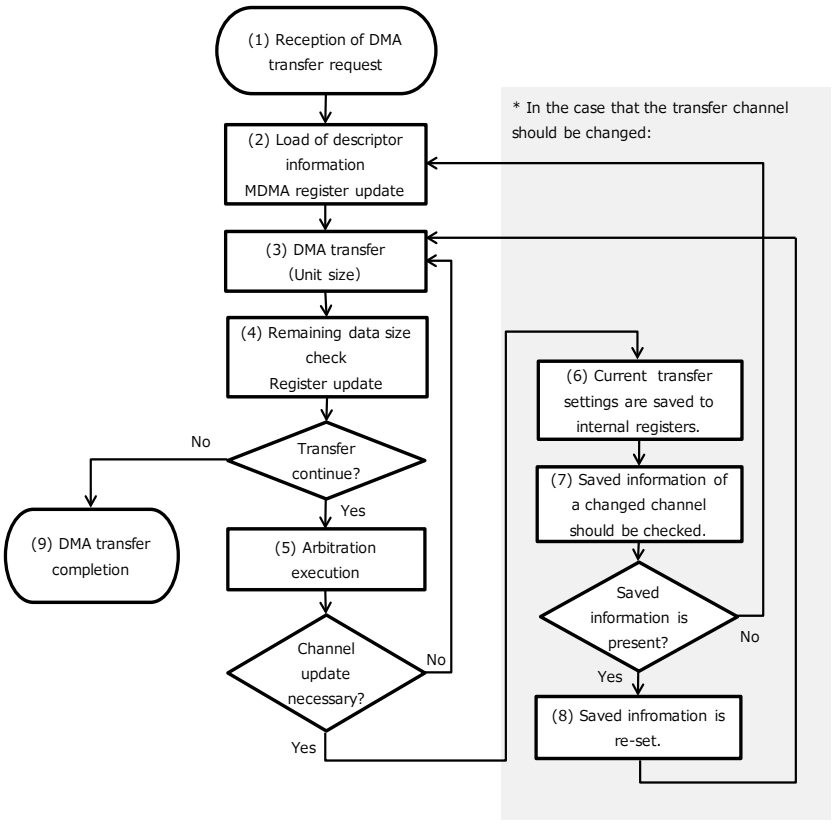


Figure 3.1 MDMAC operation flowchart

(1) The operation starts with the reception of DMA transfer request.

(2) The descriptor information of the request channel is loaded from a memory, and it is copied to registers.

(3) According to the descriptor information, the DMA transfer of the data of a specified unit size is executed.

(4) The remaining data transfer size and descriptor count should be checked. If there is no remaining, the transfer completes (9).

(5) If there is any remaining, the arbitration is executed according to the priority order.

If the change of the transfer channel is not necessary, the selected channel should not be changed and the next data of the unit size is transferred. (3), (4), and (5) steps are executed until the remaining of data transfer size and descriptor number becomes "0".

The following steps are done when the transfer change occurs after the arbitration (5).

(6) If the change of the transfer channel is necessary, the current channel transfer settings (the transfer address, transfer size, and others) should be saved to registers in MDMAC.

(7) MDMAC should check that the transfer settings of the changed channel are saved in the internal registers. If not, the descriptor information in the memory should be loaded.

(8) If the transfer settings of the changed channel have been saved, the setting information is re-set and the unit transfer begins.

3.3. Descriptor

A descriptor defines necessary information to do DMA transfer. MDMAC reads the descriptor information in a memory and executes data transfer.

3.3.1. Setting Information

Before DMA transfer is done, the following information should be set to a descriptor.

Table 3.1 Descriptor information and its corresponding register

Descriptor information	Register	Description
Transfer Type	***[MDMAxXFTYP]***	Transfer addressing mode, Unit mode, Unit size, and Transfer type are set.
Transfer Source Address	***[MDMAxXFSAD]***	Transfer source address is set. When the transfer type of the transfer source is "Region", this address increments or decrements according to the addressing mode and the unit size whenever data transfer of the unit size completes.
Transfer Destination Address	***[MDMAxXFDAD]***	Transfer destination address is set. When the transfer type of the transfer destination is "Region", this address increments or decrements according to the addressing mode and the unit size whenever data transfer of the unit size completes.
Transfer Size	***[MDMAxXFSIZ]***	Transfer size is set. This size of Byte unit decrements whenever data transfer of the unit size completes.
Descriptor Address	***[MDMAxDSADS]***	This register sets a start address of the descriptor which is executed next when the chain transfer is done.
Descriptor Number	***[MDMAxDSNUM]***	Total count of the execution descriptors is set when the chain transfer is done. Maximum 255 can be set. (Note)

Note: Unless the chain transfer is used, ***[MDMAxDSNUM]*** should be set to "0" or "1" either. The setting value in ***[MDMAxDSADS]*** is ignored.

3.3.2. Table Configuration and Address

Figure 3.2 shows the descriptor table configuration. The descriptor table needs 32-Byte capacity per channel. 32 descriptor channels are available at maximum, which means the total maximum capacity is 1024 Bytes.

Each descriptor information (refer to Table 3.1) should be located every 4 Bytes in the descriptor per channel.

The start address of the descriptor table is set to ***[MDMAxDTAB]***. The start address can be assigned to only any byte of 1024-Byte boundary.

When DMA transfer execution channel is determined, MDMAC finds the descriptor address of the channel in ***[MDMAxDTAB]***. Then, the descriptor information of the channel is loaded to a corresponding registers.

Descriptor configuration of each channel	Address
Transfer type	0x000
Transfer source address	0x004
Transfer destination address	0x008
Transfer size	0x00C
Descriptor address	0x010
Descriptor count	0x014
Reserved.	0x018
Reserved.	0x01C

Base address: ***[MDMAxDTAB]***

Descriptor table	
Address (Base+)	Data
0x000	Descriptor ch00
0x020	Descriptor ch01
0x040	Descriptor ch02
0x060	Descriptor ch03
0x080	Descriptor ch04
0x0A0	Descriptor ch05
0x0C0	Descriptor ch06
0x0E0	Descriptor ch07
0x100	Descriptor ch08
0x120	Descriptor ch09
0x140	Descriptor ch10
0x160	Descriptor ch11
0x180	Descriptor ch12
0x1A0	Descriptor ch13
0x1C0	Descriptor ch14
0x1E0	Descriptor ch15
0x200	Descriptor ch16
0x220	Descriptor ch17
0x240	Descriptor ch18
0x260	Descriptor ch19
0x280	Descriptor ch20
0x2A0	Descriptor ch21
0x2C0	Descriptor ch22
0x2E0	Descriptor ch23
0x300	Descriptor ch24
0x320	Descriptor ch25
0x340	Descriptor ch26
0x360	Descriptor ch27
0x380	Descriptor ch28
0x3A0	Descriptor ch29
0x3C0	Descriptor ch30
0x3E0	Descriptor ch31

Figure 3.2 Descriptor table configuration

Note: The definition of an unused channel's descriptor is not necessary. In this case, ***[MDMAxCEN]*** <CEN[n]> =0(n=0 to 31) should be set to make the channel setting invalid.

3.3.3. Chain Transfer and Descriptor

Figure 3.3 shows an example of descriptor configuration when a chain transfer is done on the transfer channel “n”.

The first DMA transfer is done according to the descriptor information of the transfer request channel “n” in the descriptor table. Then, the descriptor specified by the descriptor address ***[MDMAxDSADS]*** is loaded to MDMAC registers. The second DMA transfer and later are done according to the loaded descriptor information.

The descriptor address used for the chain transfer can be set in the units of 4 Bytes.

When the DMA transfer of one descriptor completes, the value in the descriptor count register ***[MDMAxDSNUM]*** in MDMAC decrements (-1). MDMAC loads the descriptor specified by ***[MDMAxDSADS]*** and executes DMA transfer until the count of the remaining descriptors becomes 0.

“Descriptor Number” in the descriptor table can be set to 255 at maximum. When 2 or more is set, the chain transfer is executed. Unless the chain transfer is used, “Descriptor Number” (***[MDMAxDSNUM]***) should be set “0” or “1”. The setting in “Descriptor Address” (***[MDMAxDSADS]***) is ignored.

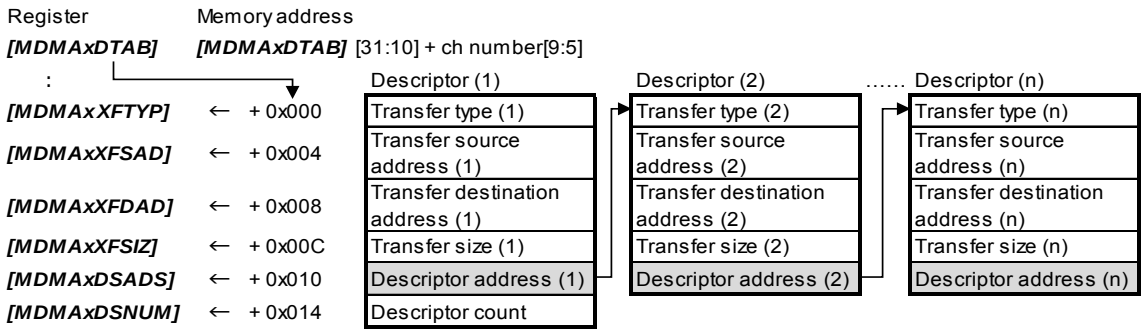


Figure 3.3 Descriptor configuration when chain transfer

3.3.4. Infinite Transfer and Descriptor

Figure 3.4 shows an example of descriptor configuration in the infinite transfer mode.

When ***[MDMAxDSNUM]***<DSINF>=1, ***[MDMAxDSNUM]***<DSNUM[7:0]> is ignored. In this case, the next descriptor addresses should be set to a loop not to cut the chain. When the transfer is stopped, DMA transfer active register ***[MDMAxACT]***<ACT[n]> should be set to "1" to force the transfer to stop.

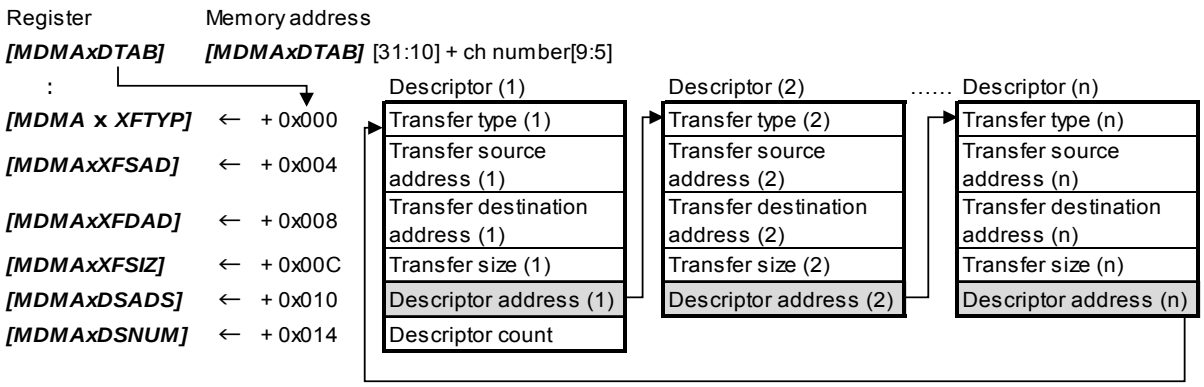


Figure 3.4 Descriptor configuration in the infinite transfer mode

3.3.5. Saving Descriptor Register

MDMAC has a saving descriptor register which stores the descriptor information of each channel. The relation among the descriptor information, the transfer register, and the saving descriptor register is shown in Figure 3.5

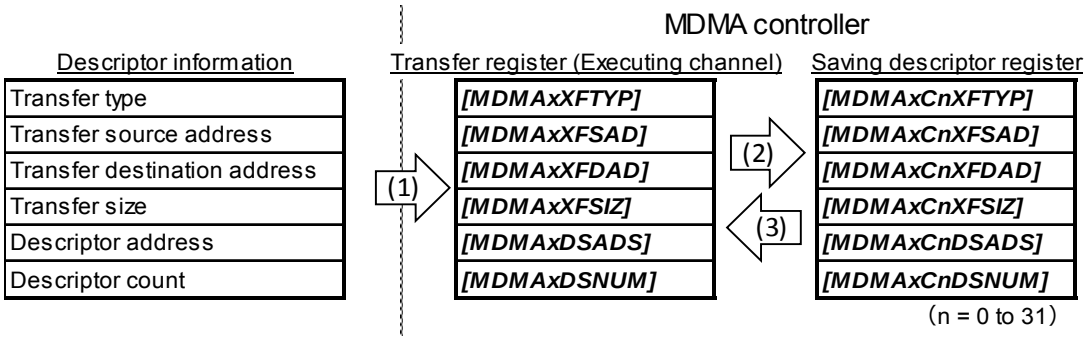


Figure 3.5 Relation between descriptors and MDMAC registers

(1) When DMA transfer execution channel is determined, MDMAC loads the descriptor information of the execution channel from the descriptor table to the transfer register. The DMA transfer starts according to the information in the register.

(2) Then, MDMAC copies the content in the transfer register to the saving descriptor register of the selected channel when one of the followings.

- Unit size data transfer completes. (Note1)
- DMA transfer of the selected channel completes.
- After arbitration, the execution channel changes to another channel. (Note2)

(3) DMA transfer of the selected channel completes. If un-transferred information is present in the saving descriptor register, it is copied to the transfer register from the saving descriptor register.

Note1: Whenever the transfer of the unit size set in ***[MDMAxXFTYP]***<USIZE> completes, the copy is done even in the continuous transfer.

Note2: When the channel is changed after arbitration during the chain transfer, the copy is done.

3.4. DMA Start-up Factor

DMA start-up factors are a transfer request from a peripheral device and a software request.

For the relation between each peripheral device and its start-up factor of the transfer request, refer to “Product Information” in Reference manual.

3.4.1. Reception of Transfer Request

When a corresponding channel is enabled (***[MDMAxCEN]***<CEN[n]>=1; n=0 to 31) and the channel request is also enabled (***[MDMAxREQ]***<REQ[n]>=1), MDMAC can receive the transfer request. (Note1)

Even though the channel which is different from the requested channel is now transferring data, the transfer request is received. The received request from a peripheral device is stored in the internal Transfer reservation flag and is copied on ***[MDMAxREQ]*** at the next arbitration (refer to Section 3.7.2). (Note2) (Note3)

When the requested transfer completes, the corresponding channel bits in ***[MDMAxCEN]*** and ***[MDMAxREQ]*** are automatically cleared to “0”.

Note1: There are no order restrictions between the setting of a request to ***[MDMAxREQ]*** and the setting of a channel enable to ***[MDMAxCEN]***. They can be set independently. The request can be received when both corresponding channel bits become “1”.

Note2: A request setting to the channel when the same channel bit in ***[MDMAxREQ]*** has been “1”, the second request is ignored.

Note3: When a channel has been stopped forcibly during data transfer before, ***[MDMAxREQ]***<REQ[n]> of the channel may be "1" before the peripheral device operates. If so, the forced stop (refer to Section 3.8.2) should be done again.

3.4.2. Software Request

When the start-up is done by software, the corresponding channel bit in Transfer request register ***[MDMAxREQ]*** should be set to “1”.

3.4.3. Transfer Request from Peripheral Device

When MDMAC receives a DMA transfer request from a peripheral device, the request is stored in Transfer reservation flag. If no channels transfer any data or the transfer request has not been received by the target channel (***[MDMAxREQ]***<REQ[n]>=0), the corresponding channel bit in ***[MDMAxREQ]*** is set to “1” at the next arbitration. In this case, it is necessary that the mask bit of the corresponding channel in Transfer request mask register ***[MDMAxMSK]*** is cleared (set to “1”).

For the relation between each peripheral device and its start-up factor of the transfer request, refer to “Product Information” in Reference manual.

3.4.4. Precaution for Use of Transfer Request Signal

The following precautions should be considered when a unit transfer is executed using a transfer interrupt of communication function (UART, TSPI, and others).

- The first unit transfer cannot be started up by the DMA transfer request. It is necessary that a software request is used or CPU transfer is executed.
- When the transfer completes the transfer of the set size data, MDMAC finishes the transfer. But the corresponding channel bit in ***[MDMAxREQ]*** becomes "1" because the communication function generates last a request. It should be noted that DMA transfer begins when ***[MDMAxCEN]*** is set. This does not occur, if ***[MDMAxREQ]*** is cleared by ***[MDMAxACT]*** setting after the transfer completion (refer to "3.8.2Forced Stop of Transfer").

3.5. Transfer Type

The size of data which MDMAC transfers for one transfer request depends on a transfer type.

The transfer type can be selected between the followings by ***[MDMAxXFTYP]***<UMODE> setting.

- Unit transfer(<UMODE>=1)

  Data of a unit size (***[MDMAxXFTYP]***<USIZE>) is transferred for one transfer request. After the transfer completion, the corresponding channel bit in ***[MDMAxREQ]*** is cleared to "0", and MDMAC waits for the next transfer request.

- Continuous transfer (<UMODE>=0)

  All data of a transfer size (***[MDMAxXFSIZ]***) are transferred for one transfer request. After all data transfer completes, the corresponding channel bit in ***[MDMAxREQ]*** is cleared to "0".

3.6. Transfer Mode

A normal transfer or a chain transfer is selected by the descriptor count setting (***[MDMAxDSNUM]***).

- Normal transfer

  [MDMAxDSNUM] = 0 or 1:

  Only one descriptor is used. Using the transfer type setting, "Unit normal transfer" or "Continuous normal transfer" can be selected.

- Chain transfer

  [MDMAxDSNUM] ≥ 2

  Two or more descriptors are used. A chain is constructed by assigning the next descriptor address in ***[MDMAxDSADS]***.

  The mixture of a unit transfer and a continuous transfer is available in the chain transfer (one descriptor can select only one transfer type.)

3.6.1. Normal Transfer

A normal transfer has two kinds of transfer: Unit normal transfer and Continuous normal transfer.

3.6.1.1. Unit Normal Transfer

In the unit normal transfer, when a transfer request is received, the corresponding channel bit is set to "1" (<REQ[n]> = 1) in Transfer request register ***[MDMAxREQ]***. Then, the descriptor information of the corresponding channel is loaded and is copied to Transfer setting registers. After data of a unit size is transferred, the corresponding channel bit is cleared to "0" (<REQ[n]> = 0) in ***[MDMAxREQ]***, and MDMAC waits for the next transfer request. Arbitration is done, if necessary. The above procedure repeats until the data of the transfer size (***[MDMAxXFSIZ]***) is transferred completely. When all transfers complete, Transfer completion interrupt is generated.

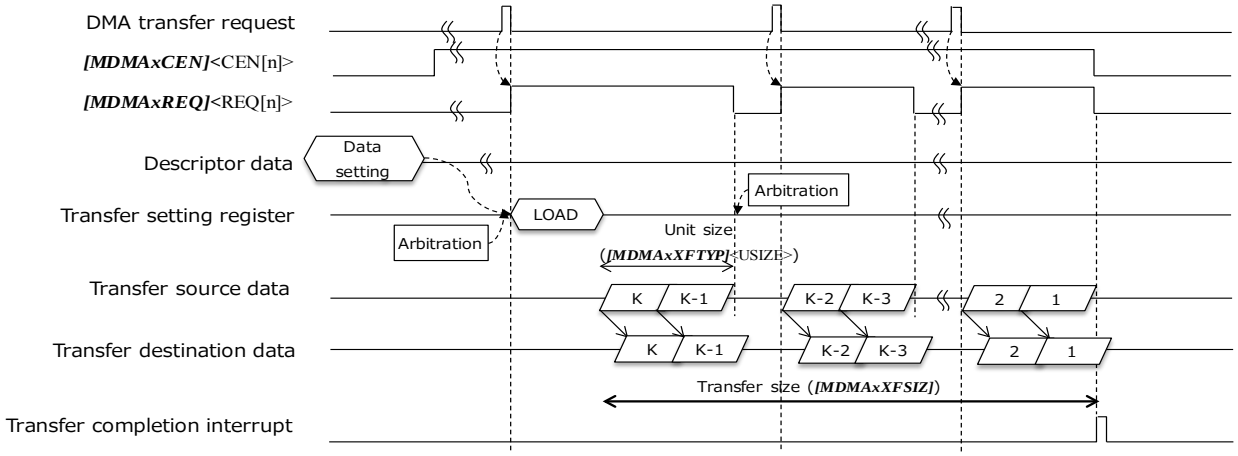


Figure 3.6 Unit normal transfer

3.6.1.2. Continuous Normal Transfer

Though the basic operation is the same as the unit normal transfer, the corresponding channel bit is not cleared to "0" in ***[MDMAxREQ]*** at the completion of the transfer of the unit size in the continuous normal transfer. After an arbitration, the next data of the unit size is transferred. MDMAC does not wait for a transfer request. The above procedure repeats until the data of the transfer size (***[MDMAxXFSIZ]***) is transferred completely. When all transfers complete, Transfer completion interrupt is generated.

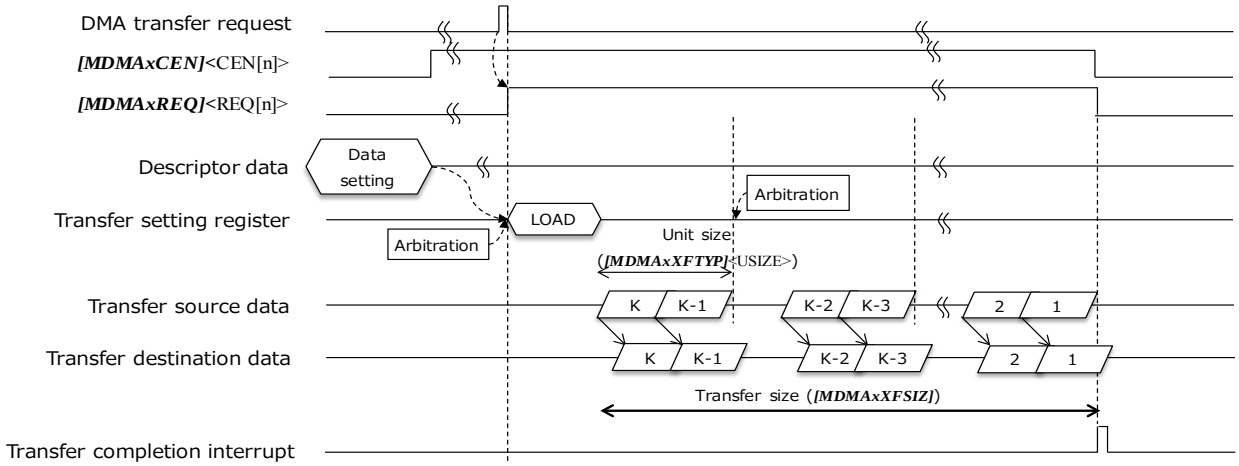


Figure 3.7 Continuous normal transfer

3.6.2. Chain Transfer

In order to use a chain transfer, it is necessary that two or more descriptors are configured in advance as shown in Figure 3.3. The next descriptor address is assigned in ***[MDMAxDSADS]***.

The total of the transfer size set by each descriptor is the data size which the chain transfer can transmit.

The chain transfer can handle the mixture data of a continuous transfer and a unit transfer. So, neither the unit transfer mode nor the continuous transfer mode is present, which is different from the normal transfer. But the descriptor change should be done carefully. For the details, refer to "3.6.3Precautions for Chain Transfer".

3.6.3. Precautions for Chain Transfer

The operation of Transfer request register ***[MDMAxREQ]*** is different between a continuous transfer and a unit transfer. When a continuous transfer and a unit transfer are mixed in the chain transfer, the followings should be noted.

(1) Change from a continuous transfer to a unit transfer

When the transfer changes from a continuous transfer to a unit transfer in a chain transfer, the first unit transfer is done without the transfer request (write to ***[MDMAxREQ]*** or assertion of the transfer request).

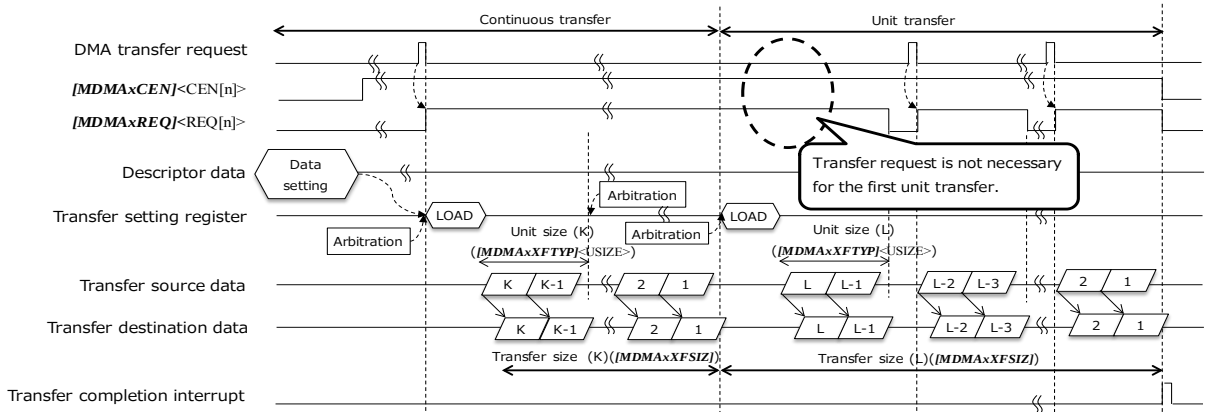


Figure 3.8 Change from a continuous transfer to a unit transfer

(2) Change from a unit transfer to a continuous transfer

When the transfer changes from a unit transfer to a continuous transfer in a chain transfer, the transfer request (write to ***[MDMAxREQ]*** or assertion of the transfer request) is necessary.

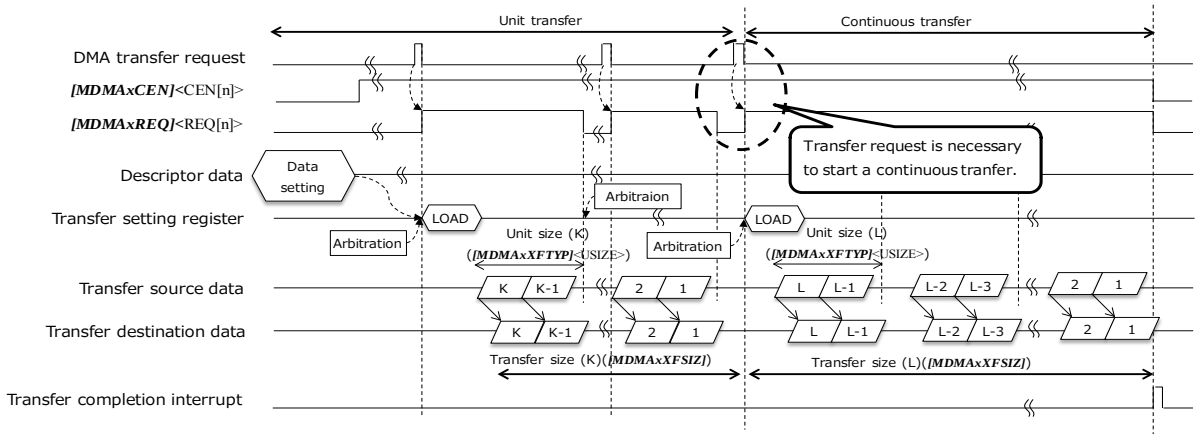


Figure 3.9 Change from a unit transfer to a continuous transfer

3.6.4. Infinite Transfer

When ***[MDMAxDSNUM]***<DSINF>=1 is set, MDMAC executes an infinite transfer in the chain transfer. MDMAC continues to transfer according to the assignment of Descriptor address in ***[MDMAxDSADS]*** regardless of the setting of ***[MDMAxDSNUM]***<DSNUM[7:0]>. So, the next descriptor address in ***[MDMAxDSADS]*** should be set in order to make a loop, as shown in Figure 3.4.

A corresponding channel bit should be set to “1” in MDMA transfer active register ***[MDMAxACT]*** to stop the infinite transfer forcibly.

Note: It is recommended that the infinite transfer mode should be used when DMA transfer is done intermittently among unit transfers.

3.7. Priority Order and Arbitration

When multiple channels receive each transfer request at the same time, MDMAC does arbitration according to the priority order and determines the channel which should be used for the transfer.

3.7.1. Priority Order

The priority order is determined by the following settings, as shown in Table 3.2.

- Priority setting

  MDMA transfer priority setting register ***[MDMAxPRI]*** is used.

  The channel bit <PRI[n]>=1 means “High priority” and <PRI[n]>=0, “Normal priority”. “High priority” is prioritized.

  The default setting is “Normal priority” (<PRI[n]>=0).

- Channel number

  The channel number 0 is the highest. The smaller number channel has the higher priority.

Table 3.2 Channel priority order

[MDMAxPRI] setting	Channel number	Priority order (descending order)
<PRI[n]> = 1 High priority	ch 0	Highest priority channel
	ch 1	↑
	:	↑
	ch 31	↑
<PRI[n]> = 0 Normal priority	ch 0	↓
	ch 1	↓
	:	↓
	ch 31	Lowest priority channel

3.7.2. Arbitration

MDMAC executes an arbitration to determine the next transfer channel whenever a data transfer of the unit size set in ***[MDMAxXFTYP]***<USIZE> completes.

Figure 3.10 shows an example of an arbitration operation. For example, during the transfer of Channel 1, the transfer request of Channel 0 is generated. Channel 0 has higher priority than Channel 1. MDMAC completes the current transfer of the unit size. Then it reflects the transfer request of Channel 0 in the arbitration. As a result, it determines to change the transfer channel to the prioritized Channel 0, and loads Channel 0 descriptor to execute a unit transfer. Moreover, if MDMA transfer priority of Channel 1 is set to "High priority" during the transfer, MDMAC reflects this change in the arbitration after the completion of the current unit transfer, again. Then, it changes the transfer channel to Channel 1.

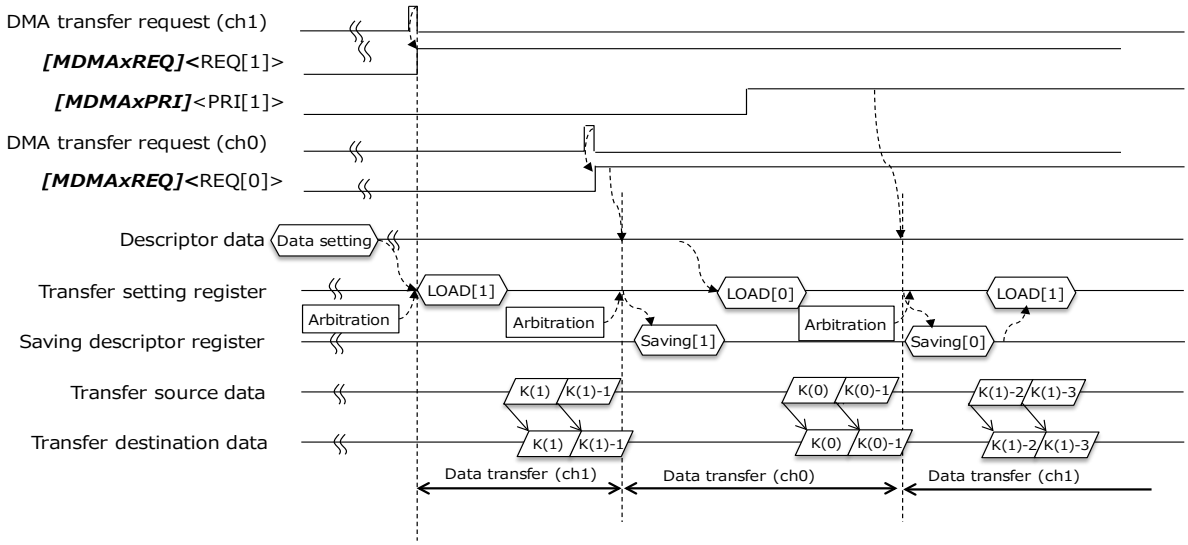


Figure 3.10 Example of an arbitration operation

Transfer suspension and forced stop can be done during DMA transfer shown in the section "3.8Transfer Suspension and Forced Stop". The setting is reflected in arbitration after the current transfer of a unit size completes.

3.8. Transfer Suspension and Forced Stop

3.8.1. Transfer Suspension and Restart

After DMA transfer starts, it can be suspended, and it can also restart during its suspension.

The transfer should be suspended by setting MDMA transfer suspension register ***[MDMAxSUS]***<SUS[n]>=1 (n = 0 to 31). If this setting is done, after MDMAC completes the current unit transfer, the transfer settings of the suspended channel (Transfer type, Transfer source and destination addresses, Transfer size, and others) are stored to saving descriptor register region in MDMAC.

The transfer restart should be done by setting MDMA transfer suspension register ***[MDMAxSUS]***<SUS[n]>=0 (n = 0 to 31). If this setting is done, MDMAC adds the restarted channel as an arbitration candidate. If the channel is determined as the next transfer one, DMA setting information of the suspended channel in saving descriptor register region is retrieved and the transfer restarts.

The suspended channel information can be read from saving descriptor register region.

The status in MDMA transfer active register ***[MDMAxACT]***<ACT[n]>=1 (n=0 to 31) is held during the transfer suspension.

3.8.2. Forced Stop of Transfer

When a channel is enabled, the transfer in the channel can be stopped forcibly after DMA transfer request is generated or the transfer starts. The procedure of the forced stop depends on the content of the transfer request.

- The enable of the corresponding channel (***[MDMAxCEN]***<CEN[n]>=1) should be cleared, or ***[MDMAxREQ]***<REQ[n]>=1 before the peripheral device operates.

  --> Refer to (1).

- No transfer requests are generated by a peripheral device after start-up by the software request (example: Memory transfer).

  --> Refer to (1).

- A transfer request is generated by a peripheral device after start-up by the software request (example: Serial data transmission).

  --> Refer to (2).

- Start-up is done by a transfer request of a peripheral device (example: Serial data reception, Timer interrupt, and others).

  --> Refer to (2).

(1) No transfer requests are generated by a peripheral device.

"1" should be set to the corresponding channel bit in MDMA transfer active register ***[MDMAxACT]*** <ACT[n]> (n=0 to 31). MDMAC clears the corresponding channel bits in ***[MDMAxCEN]***, ***[MDMAxREQ]***, and ***[MDMAxSUS]*** when it detects a write to ***[MDMAxACT]***.

Then, if a transfer is executed, the current unit transfer completes and the corresponding channel bit in MDMA transfer end register ***[MDMAxEND]*** is set to "1", which completes the forced stop procedure. If a transfer is not executed, the forced stop procedure is done immediately. It is necessary that the corresponding channel bit in ***[MDMAxEND]*** should be cleared by setting "1" after the forced stop procedure completes. (Note)

Note: When the forced stop is done after a transfer request is generated (both ***[MDMAxCEN]*** and ***[MDMAxREQ]*** are set to "1"), the following may occur. Even though ***[MDMAxACT]***<ACT[n]> is "0" before the start of the forced stop procedure, ***[MDMAxEND]*** may be "1" after the completion of the forced stop procedure. This occurs because MDMA controller may accept a transfer request before it executes the forced stop procedure and after ***[MDMAxACT]***<ACT[n]> is read. So, when the forced stop is done after a transfer request is generated, even though ***[MDMAxACT]***<ACT[n]> is "0", ***[MDMAxEND]*** should be read after the completion of the forced end procedure and the register should be cleared, if necessary.

(2) A transfer request is generated by a peripheral device.

When a peripheral generates a transfer request, Transfer reservation flag should be also cleared. The following is an example to clear it using an unexecuted channel.

(Setting example)

- Target forced-stop channel: Channel 0 (Priority = Normal)
- Unexecuted channel: Channel 31 (Priority = High)
- Descriptor allocation: ***[MDMAxDTAB]*** = 0x20028000 //"Necessary data A" should be shown later.
- Transfer source: 0x20028500 ("Necessary data B" should be set in RAM.)
- Transfer destination: 0x400A4010 (***[MDMAxACT]*** register)
- Necessary data:

(A. Descriptor: Allocation to ***[MDMAxDTAB]*** + 0x20 × 31 = 0x200283E0)

```
0x200283E0 = 0x00000303;    // Transfer type (Increment and continuous 2 words)
0x200283E4 = 0x20028500;    // Transfer source ("Necessary data B")
0x200283E8 = 0x400A4010;    // Transfer destination (***[MDMAxACT]*** register)
0x200283EC = 0x00000010;    // Transfer size (2 Words × 2 times = 16 byte transfer)
0x200283F0 = 0x00000000;    // Next descriptor address (End)
0x200283F4 = 0x00000000;    // Descriptor count (Unused.)
```

(B. Transfer source data: Allocation to any addresses)

```
0x20028500 = 0x00000001;    // For ***[MDMAxACT]*** register
0x20028504 = 0x00000001;    // For ***[MDMAxEND]*** register
```

(1) The input mask of a transfer request by Channel 0 (***[MDMAxMSK]***<MSK0>=0)

This prevents the collision between a transfer request by a peripheral and a forced stop procedure.

It is recommended that the target peripheral device should be also stopped at that time.

(2) Transfer end bit in Channel 31 should be cleared (***[MDMAxEND]***<END31>=1).

Transfer end bit of the unexecuted channel should be cleared before the channel is used.

(3) "Necessary data" should be prepared.

(4) The priority of Channel 31 should be set to High (***[MDMAxPRI]***<PRI31>=1), and DMA is started up by the software request (***[MDMAxREQ]***<REQ31>=1).

When the first clear of the transfer request of Channel 0 is done (***[MDMAxACT]***<ACT0>=1), MDMAC clears the bit 0's in ***[MDMAxCEN]***, ***[MDMAxREQ]***, and ***[MDMAxSUS]***.

Any one channel transfer completes, the request of transfer reservation flag of Channel 0 is cleared (***[MDMAxREQ]***<REQ0> becomes 1).

When the second clear of the transfer request of Channel 0 is done (***[MDMAxACT]***<ACT0> =1), all transfer requests are cleared.

(5) The high priority setting of Channel 31 should be cleared (***[MDMAxPRI]***<PRI31>=0).

(6) The input mask of the transfer request by Channel 0 should be cleared (***[MDMAxMSK]***<MSK0> =1), if necessary.

3.9. Software Reset

Software reset can be asserted to MDMAC by ***[CGEXTEND2]*** register. After the software reset assertion, MDMAC returns to the initial state which is the same as the state after hardware reset assertion. Refer to "Clock Control and Operation Mode" in Reference manual.

(1) Set ***[CGEXTEND2]***<RSV22>=0.

(2) Set ***[CGEXTEND2]***<RSV22>=1. Wait 4 cycles or more of the MDMA supply clock.

(3) Again, set ***[CGEXTEND2]***<RSV22>=0

The software reset should be used only at the following events.

- Detection of a bus error or a descriptor error
- No DMA transfers

This is the case that there are no channels which are corresponding to MDMA transfer active register ***[MDMAxACT]***<ACT[n]>=1 (n=0 to 31).

When a bus error or a descriptor error occurs and the channel corresponding to <ACT[n]>=1 exists, the transfer of the channel has stopped. Assertion of the software reset can be done without any problems.

Note: The Transfer reservation flag is not cleared by this software reset. There may be the channel in which ***[MDMAxREQ]***<REQ[n]> =1 before the peripheral device operates. If so, the forced stop (refer to "3.8.2 Forced Stop of Transfer") should be done to the channel again.

3.10. Interrupt

MDMAC can generate a transfer completion interrupt and an error interrupts (a descriptor error and a bus error). When an error interrupt is generated, the transfer stops.

3.10.1. Transfer Completion Interrupt

MDMAC can generate MDMA transfer completion interrupt at the completion of DMA transfer. When an interrupt is used, MDMA transfer completion interrupt enable register ***[MDMAxENE]***<ENE[n]>=1 (n=0 to 31) should be set before the start of DMA transfer.

MDMA transfer completion interrupt is generated when a transfer completes and transfer end register ***[MDMAxEND]***<END[n]>=1 (n=0 to 31) is set. The interrupt is also generated when the transfer is stopped by a forced stop after the start of transfer. And it is generated at the completion of all transfers in both the unit transfer mode and the continuous transfer mode. It is not generated after each transfer of a unit size data is done in the unit transfer mode.

3.10.2. Descriptor Error Interrupt

MDMAC detects an error when the setting of the channel descriptor which is loaded in MDMAC has one of the following problems. It notifies CPU of the descriptor error.

- Transfer data size is 0:

  When the descriptor transfer size of the original descriptor (not a descriptor chain) is “0” and it is loaded into the MDMA transfer register ***[MDMAxXFSIZ]***.

  When "0" of descriptor transfer size is loaded to MDMA transfer size register ***[MDMAxXFSIZ]*** at descriptor setting of last descriptor chain (***[MDMAxDSNUM]***=0).

- The alignment of the transfer address is wrong:

  The unit size ***[MDMAxXFTYP]***<USIZE> specifies a word or more. But transfer source address register ***[MDMAxXFSAD]*** or transfer destination address register is not aligned with a word.

  The unit size ***[MDMAxXFTYP]***<USIZE> specifies a half word. But transfer source address register ***[MDMAxXFSAD]*** or transfer destination address register ***[MDMAxXFDAD]*** is not aligned with a half word.

- The alignment of the transfer size is wrong:

  The unit size ***[MDMAxXFTYP*** <USIZE> specifies a word or more. But transfer size register ***[MDMAxXFSIZ]*** is not aligned with a word.

  The unit size ***[MDMAxXFTYP]***<USIZE> specifies a half word. But transfer size register ***[MDMAxXFSAD]*** is not aligned with a half word.

Table 3.3 Alignment and address

Alignment	Source Address/Destination Address
000: 1 byte	0x0, 0x1, 0x2, 0x3, 0x4…
001: 2 byte (half word)	Set to be multiple of 2 (0x0,0x2,0x4,0x6,0x8,0xA,0xC,0xE…)
010: 4 byte(1 word)	Set to be multiple of 4 (0x0, 0x4, 0x8, 0xC…)
011: 8 byte(2 word)	
100: 16 byte(4 word)	
101: 32 byte(8 word)	

3.10.3. Bus Error Interrupt

When MDMAC detects an error during descriptor load or data transfer, it notifies CPU of the bus error.

3.10.4. Initialization Procedure at Error Occurrence

When MDMAC detects a descriptor error or a bus error, it stops the transfer operation. In order to restart the transfer, MDMAC should be initialized and the error information should be cleared by the following procedure.

The software reset is also asserted at the same time. So the re-settings of MDMAC should be done.

Refer to “Clock Control and Operation Mode” of reference manual about ***[CGEXTEND2]***.

(1) Set ***[CGEXTEND2]***<RSV22><RSV21><RSV20>=0.
(2) Set one of the followings according to the error. Then, wait 4 cycles or more of MDMA supply clock.
  (a) Bus error: Set ***[CGEXTEND2]***<RSV22><RSV20>=1
  (b) Descriptor error: Set ***[CGEXTEND2]***<RSV22><RSV21>=1
(3) Set ***[CGEXTEND2]***<RSV22><RSV21><RSV20>=0, again

4. Registers

4.1. List of Registers

MDMAC registers and their addresses are shown in the following tables.

Peripheral function		Channel/Unit	Base address	
			TYPE 1	TYPE 2
MDMA controller	**MDMAC**	Unit A	-	0x400A4000

Note: The channel/unit and base address type are different by products. Please refer to "Product Information" of the reference manual for the details.

Register name		Address(Base+)
Reserved	-	0x0000
Transfer Channel Enable Register	***[MDMAxCEN]***	0x0004
Transfer Request Register	***[MDMAxREQ]***	0x0008
Transfer Suspension Register	***[MDMAxSUS]***	0x000C
Transfer Active Register	***[MDMAxACT]***	0x0010
Transfer End Register	***[MDMAxEND]***	0x0014
Transfer Priority Setting Register	***[MDMAxPRI]***	0x0018
Transfer Completion Interrupt Enable Register	***[MDMAxENE]***	0x001C
Transfer Descriptor Table Start Address Register	***[MDMAxDTAB]***	0x0020
Reserved	-	0x0024
Transfer Execution Channel Number Register	***[MDMAxCHN]***	0x0028
Transfer Type Register	***[MDMAxXFTYP]***	0x002C
Transfer Source Address Register	***[MDMAxXFSAD]***	0x0030
Transfer Destination Address Register	***[MDMAxXFDAD]***	0x0034
Transfer Size Register	***[MDMAxXFSIZ]***	0x0038
Transfer Descriptor Storage Address Register	***[MDMAxDSADS]***	0x003C
Transfer Descriptor Count Register	***[MDMAxDSNUM]***	0x0040
Reserved	-	0x0044 to 0x004C
Saving Descriptor Register	(Refer to the following table.)	0x0050 to 0x044C
Reserved	-	0x0450 to 0x07FC
Transfer Request Mask Register	***[MDMAxMSK]***	0x0800

Register name	Function Name	Channel/Unit	Address(Base+)
Saving Descriptor Register	**MDMAC**	ch 00	0x0050
		ch 01	0x0070
		ch 02	0x0090
		ch 03	0x00B0
		ch 04	0x00D0
		ch 05	0x00F0
		ch 06	0x0110
		ch 07	0x0130
		ch 08	0x0150
		ch 09	0x0170
		ch 10	0x0190
		ch 11	0x01B0
		ch 12	0x01D0
		ch 13	0x01F0
		ch 14	0x0210
		ch 15	0x0230
		ch 16	0x0250
		ch 17	0x0270
		ch 18	0x0290
		ch 19	0x02B0
		ch 20	0x02D0
		ch 21	0x02F0
		ch 22	0x0310
		ch 23	0x0330
		ch 24	0x0350
		ch 25	0x0370
		ch 26	0x0390
		ch 27	0x03B0
		ch 28	0x03D0
		ch 29	0x03F0
		ch 30	0x0410
		ch 31	0x0430

Register name		Offset Address
ch n Transfer Type Saving Register	***[MDMAxCnXFTYP]***	0x0000
ch n Transfer Source Address Saving Register	***[MDMAxCnXFSAD]***	0x0004
ch n Transfer Destination Address Saving Register	***[MDMAxCnXFDAD]***	0x0008
ch n Transfer Size Saving Register	***[MDMAxCnXFSIZ]***	0x000C
ch n Transfer Descriptor Storage Address Saving Register	***[MDMAxCnDSADS]***	0x0010
ch n Transfer Descriptor Count Saving Register	***[MDMAxCnDSNUM]***	0x0014

Note: The Saving Descriptor Register of each channel consists of the above registers. The address of each register adds the offset address to the address of the Saving Descriptor Register.

4.2. Detail of Register

4.2.1. *[MDMAxCEN]* (Transfer Channel Enable Register)

Bit	Bit Symbol	After Reset	Type	Description
31:0	CEN[31:0]	0x00000000	R/W	This register sets the enable of DMA transfer channel. Each bit corresponds to one of the channels 31 to 0 of DMA transfer. 0: Channel setting is disabled. The transfer request is ignored. 1: Channel setting is enabled. The transfer request is accepted. Write of "1" sets the channel to the enable. While this register is set to the enable, the setting of the transfer channel becomes valid if the channel whose corresponding bit in ***[MDMAxREQ]*** is set to "1" receives a request. The transfer request of the channel is reflected to the next transfer channel arbitration. When the corresponding channel transfer completes, the channel bit is automatically cleared to "0".

Note1: "1" should be written to the channel bit which is newly changed to the enable. It is invalid to write "1" to the bit which has been already set to the enable. RMW (Read-modify-write) is inhibited.

Note2: When this register should be cleared before the completion of the transfer, a forced stop should be done by writing "1" to ***[MDMAxACT]***. Write of "0" cannot clear this register.

4.2.2. *[MDMAxREQ]* (Transfer Request Register)

Bit	Bit Symbol	After Reset	Type	Description
31:0	REQ[31:0]	0x00000000	R/W	This register sets the transfer request of DMA transfer channel. Each bit corresponds to one of the channels 31 to 0 of DMA transfer. 0: No transfer request is present. 1: A transfer request is present. Write of "1" requests DMA transfer (software request). And, the assertion of a transfer request from each peripheral is detected, the corresponding bit becomes "1" automatically (assertion of a transfer request signal). While the corresponding bit in transfer channel enable register ***[MDMAxCEN]*** is set to the enable, the transfer request is valid if this register receives a transfer request. The transfer request is reflected to the next transfer channel arbitration.(Refer to 3.4.1) When the corresponding channel transfer completes, the channel bit is automatically cleared to "0".

Note1: "1" should be written to the channel bit which is newly changed to the enable. It is invalid to write "1" to the bit which has been already set to the enable. RMW (Read-modify-write) is inhibited.

Note2: When this register should be cleared before the completion of the transfer, a forced stop should be done by writing "1" to ***[MDMAxACT]***.(Refer to 3.8.2) Write of "0" cannot clear this register.

4.2.3. *[MDMAxSUS]* (Transfer Suspension Register)

Bit	Bit Symbol	After Reset	Type	Description
31:0	SUS[31:0]	0x0000000	R/W	This register sets suspension of the transfer in DMA transfer channel. Each bit corresponds to one of the channels 31 to 0 of DMA transfer. 0: No suspension request is present. 1: A suspension request is present. Write of “1” suspends the transfer after the current transfer of a unit size completes. Write of “0” to the corresponding bit of a suspending channel restarts a transfer.

Note: When a forced stop is done to a suspending channel by writing to ***[MDMAxACT]***, the corresponding bit is cleared to “0” in this register.

4.2.4. *[MDMAxACT]* (Transfer Active Register)

Bit	Bit Symbol	After Reset	Type	Description
31:0	ACT[31:0]	0x00000000	R	This register shows that the transfer of DMA transfer channel is executing. Each bit corresponds to one of the channels 31 to 0 of DMA transfer. 0: Transfer has stopped. 1: Transfer is operating. A transfer request is generated and an arbitration among channels is done to determine the target channel which is used for the next transfer. Then, the corresponding bit in this register is set to “1” after the descriptor is loaded. Even though the transfer of the target channel is suspended by an arbitration result or a suspension caused by ***[MDMAxSUS]*** setting, the bit in this register holds “1”. When the transfer completes, the corresponding bit is automatically cleared to “0”.
			W	To forced stop the DMA transfer, write "1" to the corresponding channel bit. 0: - 1: Forced stop the transfer operation When forced stop the transfer operation, each bit corresponding of ***[MDMAxCEN]***, ***[MDMAxREQ]***, ***[MDMAxSUS]*** is cleared to “0”. Also, each bit corresponding of ***[MDMAxEND]*** is set to “1”.

Note: “1” should be written to the bit of the channel which should be newly forced to stop. RMW (Read-modify-write) is inhibited.

4.2.5. *[MDMAxEND]* (Transfer End Register)

Bit	Bit Symbol	After Reset	Type	Description
31:0	END[31:0]	0x00000000	R	This register shows that a transfer of DMA transfer channel completes. Each bit corresponds to one of the channels 31 to 0 of DMA transfer. 0: DMA transfer has not completed or the transfer is cleared. 1: DMA transfer has completed. When DMA transfer completes, the corresponding channel bit is set to “1”.
			W	This register should be cleared before the next transfer is executed. 0: - 1: Write of “1” clears the bit to “0”.

Note: Write of “1” to ***[MDMAxACT]*** register after the start of transfer stops the transfer forcibly. Then this register is also set to “1”. But a forced stop before the start of transfer does not set this register to “1”.

4.2.6. *[MDMAxPRI]* (Transfer Priority Setting Register)

Bit	Bit Symbol	After Reset	Type	Description
31:0	PRI[31:0]	0x00000000	R/W	This register sets a priority of DMA transfer channel. Each bit corresponds to one of the channels 31 to 0 of DMA transfer. 0: Normal priority 1: High priority The corresponding channel of the bit written to “1” has a high priority. When two DMA transfer requests are generated, the corresponding channel of the bit written “1” in this register is executed before. If the priority in this register is the same, the smaller number channel has a higher priority.

4.2.7. *[MDMAxENE]* (Transfer Completion Interrupt Enable Register)

Bit	Bit Symbol	After Reset	Type	Description
31:0	ENE[31:0]	0x00000000	R/W	This register enables DMA transfer completion interrupt. Each bit corresponds to one of the channels 31 to 0 of DMA transfer. 0: DMA transfer completion interrupt is disabled. 1: DMA transfer completion interrupt is enabled. When DMA transfer interrupt is used, the corresponding bit of the channel should be set to “1” in this register before the start of DMA transfer.

4.2.8. *[MDMAxDTAB]* (Transfer Descriptor Table Start Address Register)

Bit	Bit Symbol	After Reset	Type	Description
31:10	DTAB[31:10]	0x000000	R/W	Specify the start address of the descriptor table stored in the memory.
9:0	DTAB[9:0]	0x000	R	Read as “0”.

Note1: The address set in this register can be only a value of the boundary of 1024 Bytes.

Note2: This register should be set before DMA transfer request is done. When a channel is executing a transfer or receives a transfer request, this register cannot be written, that is, when the channel which ***[MDMAxCEN]***<CEN[n]> and ***[MDMAxREQ]***<REQ[n]> are “1” is present, the write is ignored.

4.2.9. *[MDMAxCHN]* (Transfer Execution Channel Number Register)

Bit	Bit Symbol	After Reset	Type	Description
31:5	-	0	R	Read as “0”.
4:0	CHNUM[4:0]	00000	R	This field shows the number of DMA transfer channel which is currently executing. It is updated when descriptor information of the DMA transfer channel requested for DMA transfer is read. Also when the transfer execution channel is changed as a result of arbitration, it is updated.

4.2.10. *[MDMAxXFTYP]* (Transfer Type Register)

Bit	Bit Symbol	After Reset	Type	Description
31:25	-	0	R	Read as “0”.
24	DMODE	0	R	This bit shows an addressing mode of DMA transfer.It is valid for the transfer source/transfer destination address where "Region" is specified by <TTYPE[1:0]>. 0: Increment mode 1: Decrement mode If the direction of the addressing of DMA transfer is increasing, this bit should be set to “0”. Contrarily, if the direction of the addressing of DMA transfer is decreasing, this bit should be set to “1”.
23:17	-	0	R	Read as “0”.
16	UMODE	0	R	This bit shows the transfer mode where a transfer is done by one transfer request. 0: Continuous transfer mode 1: Unit transfer mode The continuous transfer mode executes the transfer of all data. The unit transfer mode executes the transfer of the unit size specified by <USIZE[2:0]>. And MDMAC waits for the next transfer request. Transfer completion interrupt is generated only when all transfers complete in either mode.
15:11	-	0	R	Read as “0”.
10:8	USIZE[2:0]	000	R	This field shows a unit size of the data which is transferred in one transfer cycle. 000: 1 byte 001: 2 byte (half word) 010: 4 byte (1 word) 011: 8 byte (2 word) 100: 16 byte (4 word) 101: 32 byte (8 word) 110: Reserved 111: Reserved Whenever a unit size data is transferred, a channel priority is checked(arbitration).
7:2	-	0	R	Read as “0”.
1:0	TTYPE[1:0]	00	R	This field shows DMA transfer type. transfer source transfer destination 00: Region -> Region 01: Fixed -> Region 10: Region -> Fixed 11: Fixed -> Fixed The “Region” is incremented/decremented for each unit size transfer. The “Fixed” is region of one unit size. The address is not incremented/decremented.

4.2.11. *[MDMAxXFSAD]* (Transfer Source Address Register)

Bit	Bit Symbol	After Reset	Type	Description
31:0	XFSAD[31:0]	0x00000000	R	This register shows DMA transfer source address of the current transfer. When DMA transfer execution channel is determined, the transfer source address of the corresponding channel is loaded from the descriptor table. In the chain transfer, the transfer source address is loaded from the descriptor which is referred in this transfer. When DMA transfer type of the transfer source address is "Region" (***[MDMAxXFTYP]***<TTYPE[1:0]>=00 or 10), this register is incremented / decremented whenever a transfer of a unit size data completes, according to DMA transfer addressing mode (***[MDMAxXFTYP]***<DMODE>).

Note: The address set in this register should be aligned with a unit size in ***[MDMAxXFTYP]***<USIZE>.

4.2.12. *[MDMAxXFDAD]* (Transfer Destination Address Register)

Bit	Bit Symbol	After Reset	Type	Description
31:0	XFDAD[31:0]	0x00000000	R	This register shows DMA transfer destination address of the current transfer. When DMA transfer execution channel is determined, the transfer destination address of the corresponding channel is loaded from the descriptor table. In the chain transfer, the transfer destination address is loaded from the descriptor which is referred in this transfer. When DMA transfer type of the transfer destination address is "Region" (***[MDMAxXFTYP]***<TTYPE[1:0]>=00 or 10), this register is incremented / decremented whenever a transfer of a unit size data completes, according to DMA transfer addressing mode (***[MDMAxXFTYP]***<DMODE>).

Note: The address set in this register should be aligned with a unit size in ***[MDMAxXFTYP]***<USIZE>.

4.2.13. *[MDMAxXFSIZ]* (Transfer Size Register)

Bit	Bit Symbol	After Reset	Type	Description
31:20	-	0	R	Read as “0”.
19:0	XFSIZ[19:0]	0x00000	R	This field shows the size of the remaining transfer data of the current DMA transfer in Byte unit. When DMA transfer execution channel is determined, the transfer size of the corresponding channel is loaded from the descriptor table. In the chain transfer, the transfer size is loaded from the descriptor which is referred in this transfer. When DMA transfer starts, this field shows the size of all data. Whenever a transfer of unit size completes, the data byte size is decremented. When the transfer of all data completes, this field becomes “0” (except the case of the forced stop of the transfer).

Note1: The address set in this register should be aligned with a unit size in ***[MDMAxXFTYP]***<USIZE>.

Note2: The transfer size set in this register in a descriptor table should be “1” or more.

4.2.14. *[MDMAxDSADS]* (Transfer Descriptor Storage Address Register)

Bit	Bit Symbol	After Reset	Type	Description
31:2	DSADS[31:2]	0x00000000	R	This field shows the start address of the descriptor which executes next. When DMA transfer execution channel is determined, the descriptor address of the corresponding channel is loaded from the descriptor table. In the chain transfer, the address of the descriptor which is referred next is loaded.
1:0	DSADS[1:0]	00	R	Read as “0”.

Note1: The address set in this register can be only the value of the unit of 4 bytes.

Note2: When the chain transfer is not used, this register setting is ignored.

4.2.15. *[MDMAxDSNUM]* (Transfer Descriptor Count Register)

Bit	Bit Symbol	After Reset	Type	Description
31:9	-	0	R	Read as “0”.
8	DSINF	0	R	The mode of a chain transfer is shown. 0: Normal mode 1: Infinite transfer mode In the infinite transfer mode, the chain transfer continues regardless of the descriptor count set in <DSNUM[7:0]>.
7:0	DSNUM[7:0]	0x00	R	This field shows the count of the remaining descriptors in the chain transfer. When DMA transfer execution channel is determined, the descriptor count of the corresponding channel is loaded from the descriptor table. This register shows total descriptor count when the chain transfer starts. Whenever DMA transfer of one descriptor completes, the value in this field decrements (-1). When the DMA transfer of all descriptors completes, the value in this register becomes “0”. When 2 or more is set, a chain transfer operates. When the chain transfer is not used (Normal transfer), “0” or “1” should be set. The value of 255 at maximum is available. If the value of this register is “0” at the start of the execution of DMA transfer, the mode should be a normal mode. DMA transfer of only one descriptor is done

4.2.16. *[MDMAxMSK]* (Transfer Request Mask Register)

Bit	Bit Symbol	After Reset	Type	Description
31	MSK31	0	R/W	This register sets a mask for the transfer request of DMA transfer channel. Each bit corresponds to one of the channels 31 to 0 of DMA transfer. 0: DMAREQ[n] input signal is masked (Disable a transfer request). 1: DMAREQ[n] input signal is not masked (Enable a transfer request).
30	MSK30	0	R/W	
29	MSK29	0	R/W	
28	MSK28	0	R/W	
27	MSK27	0	R/W	
26	MSK26	0	R/W	
25	MSK25	0	R/W	
24	MSK24	0	R/W	
23	MSK23	0	R/W	
22	MSK22	0	R/W	
21	MSK21	0	R/W	
20	MSK20	0	R/W	
19	MSK19	0	R/W	
18	MSK18	0	R/W	
17	MSK17	0	R/W	
16	MSK16	0	R/W	
15	MSK15	0	R/W	
14	MSK14	0	R/W	
13	MSK13	0	R/W	
12	MSK12	0	R/W	
11	MSK11	0	R/W	
10	MSK10	0	R/W	
9	MSK9	0	R/W	
8	MSK8	0	R/W	
7	MSK7	0	R/W	
6	MSK6	0	R/W	
5	MSK5	0	R/W	
4	MSK4	0	R/W	
3	MSK3	0	R/W	
2	MSK2	0	R/W	
1	MSK1	0	R/W	
0	MSK0	0	R/W	

4.2.17. *[MDMAxCnXFTYP]* (ch n Transfer Type Saving Register; n = 00 to 31)

Bit	Bit Symbol	After Reset	Type	Description
31:25	-	0	R	Read as “0”.
24	DMODE	0	R	The addressing mode of DMA transfer is shown. It is valid for the transfer source/transfer destination address where "Region" is specified by <TTYPE[1:0]>. 0: Increment mode 1: Decrement mode If the direction of the addressing of DMA transfer is increasing, this bit should be set to “0”. Contrarily, if the direction of the addressing of DMA transfer is decreasing, this bit should be set to “1”.
23:17	-	0	R	Read as “0”.
16	UMODE	0	R	The transfer mode for one transfer request is shown. 0: Continuous transfer mode 1: Unit transfer mode The continuous transfer executes the transfer of all data. The unit transfer executes the transfer of the unit size specified by <USIZE[2:0]>. And MDMAC waits for the next transfer request. Transfer completion interrupt is generated only when all transfers complete in either mode.
15:11	-	0	R	Read as “0”.
10:8	USIZE[2:0]	000	R	This field shows a unit size of the data which is transferred in one transfer cycle. 000: 1 byte 001: 2 byte (half word) 010: 4 byte (1 word) 011: 8 byte (2 word) 100: 16 byte (4 word) 101: 32 byte (8 word) 110: Reserved 111: Reserved Whenever a unit size data is transferred, a channel priority is checked(arbitration).
7:2	-	0	R	Read as “0”.
1:0	TTYPE[1:0]	00	R	This field shows DMA transfer type. transfer source transfer destination 00: Region -> Region 01: Fixed -> Region 10: Region -> Fixed 11: Fixed -> Fixed The “Region” is incremented/decremented for each unit size transfer. The “Fixed” is region of one unit size. The address is not incremented/decremented.

4.2.18. *[MDMAxCnXFSAD]* (ch n Transfer Source Address Saving Register; n = 00 to 31)

Bit	Bit Symbol	After Reset	Type	Description
31:0	XFSAD[31:0]	0x00000000	R	This register shows DMA transfer source address of the corresponding channel. When the corresponding channel starts DMA transfer, the transfer source address is loaded from the descriptor table. In the chain transfer, the transfer source address is loaded from the descriptor which is referred in this transfer. When DMA transfer type of the transfer source address is “Region” (***[MDMAxCnXFTYP]***<TTYPE[1:0]>=00 or 10), this register is incremented / decremented whenever a transfer of a unit size data completes, according to DMA transfer addressing mode (***[MDMAxCnXFTYP]***<DMODE>).

4.2.19. *[MDMAxCnXFDAD]* (ch n Transfer Destination Address Saving Register; n = 00 to 31)

Bit	Bit Symbol	After Reset	Type	Description
31:0	XFDAD[31:0]	0x00000000	R	This register shows DMA transfer destination address of the corresponding channel. When the corresponding channel starts DMA transfer, the transfer destination address is loaded from the descriptor table. In the chain transfer, the transfer destination address is loaded from the descriptor which is referred in this transfer. When DMA transfer type of the transfer destination address is “Region” (***[MDMAxCnXFTYP]***<TTYPE[1:0]>=00 or 10), this register is incremented / decremented whenever a transfer of a unit size data completes, according to DMA transfer addressing mode (***[MDMAxCnXFTYP]***<DMODE>).

4.2.20. *[MDMAxCnXFSIZ]* (ch n Transfer Size Saving Register; n = 00 to 31)

Bit	Bit Symbol	After Reset	Type	Description
31:20	-	0	R	Read as “0”.
19:0	XFSIZ[19:0]	0x00000	R	This field shows the size of the remaining DMA transfer data of the corresponding channel in byte unit. When the corresponding channel starts DMA transfer, the transfer size is loaded from the descriptor table. In the chain transfer, the transfer size is loaded from the descriptor which is referred in this transfer. When DMA transfer starts, this field shows the size of all data. Whenever a transfer of unit size completes, the data byte size is decremented. When the transfer of all data completes, this field becomes “0” (except the case of the forced stop of the transfer).

4.2.21. *[MDMAxCnDSADS]* (ch n Transfer Descriptor Storage Address Saving Register; n = 00 to 31)

Bit	Bit Symbol	After Reset	Type	Description
31:2	DSADS[31:2]	0x00000000	R	This field shows the start address of the descriptor which executes next. When the corresponding channel starts DMA transfer, the descriptor address is loaded from the descriptor table. In the chain transfer, the address of the descriptor which is referred next is loaded in this transfer.
1:0	DSADS[1:0]	00	R	Read as “0”.

Note: The address set in this register can be only the value of the boundary of 4 bytes.

4.2.22. *[MDMAxCnDSNUM]* (ch n Transfer Descriptor Count Saving Register; n = 00 to 31)

Bit	Bit Symbol	After Reset	Type	Description
31:9	-	0	R	Read as “0”.
8	DSINF	0	R	The mode of a chain transfer is shown. 0: Normal mode 1: Infinite transfer mode In the infinite transfer mode, the chain transfer continues regardless of the descriptor count set in <DSNUM[7:0]>.
7:0	DSNUM[7:0]	0x00	R	This field shows the count of the remaining descriptors in the chain transfer mode. When DMA transfer execution channel is determined, the descriptor count of the corresponding channel is loaded from the descriptor table. Whenever DMA transfer of one descriptor completes, the value in this field decrements (-1). When the DMA transfer of all descriptors completes, the value in this register becomes “0”. When 2 or more is set, a chain transfer operates. When the chain transfer is not used (Normal transfer), “0” or “1” should be set. The value of 255 at maximum is available. If the value of this register is “0” at the start of the execution of DMA transfer, the mode should be a normal mode. DMA transfer of only one descriptor is done.

5. Example of Usage

5.1. Unit Normal Transfer

This is an example that a unit normal transfer of 100-byte data (16-bit data x 50) is done using TSPI reception. The transfer is in High priority.

(Prerequisite conditions)

- DMA start-up factor: TSPI transmission (DMA transfer channel number=0)
- Transfer source: TSPI data register (Address=0x400CB900)
- Transfer destination: Reception buffer in RAM (Address=0x20028400)

(1) Used function should be set.

TSPI reception and DMA interface should be set.

(2) Descriptor information should be allocated to RAM.

// The allocation address is ***[MDMAxDTAB]*** value + DMA transfer channel number × 0x20.

// In this case, 0x20028000 + 0 × 0x20 = 0x20028000

Descriptor	value	
0x20028000 =	0x00001201	// Transfer type
		// (Increment, Unit transfer, 2-byte unit, and “Fixed -> Region”)
0x20028004 =	0x400CB900	// Transfer source address (TSPI data register)
0x20028008 =	0x20028400	// Transfer destination address
		// (Reception buffer start address in RAM)
0x2002800C =	0x00000064	// Transfer size (100 byte)
0x20028010 =	0x00000000	// Descriptor address (Chain transfer is unused.)
0x20028014 =	0x00000000	// Descriptor count (Chain transfer is unused.)

(3) Registers should be set.

[MDMAxDTAB] =0x20028000	// Descriptor base address is set
[MDMAxMSK] <MSK[0]>= 1	// ch0 transfer request enable
[MDMAxENE] <ENE[0]>= 1	// Interrupt is generated after the completion of transfer.
[MDMAxPRI] <PRI[0]>= 1	// High priority is set.
[MDMAxCEN] = 0x00000001	// ch0 is enabled (Word access to inhibit RMW)

(4) DMA transfer request of TSPI reception completion

When DMA transfer request occurs, data register of TSPI is read. And it is written to reception buffer in RAM.

5.2. Continuous Normal Transfer

This is an example of the setting that ROM data (1 KB) is transferred to an external bus triggered by external pin signal.

(Prerequisite conditions)

- DMA start-up factor: External pin (DMA transfer channel number=31)
- Transfer source: Flash ROM data (Address = 0x5E080000: Mirror region (Note))
- Transfer destination: External bus (Address = 0x60000000: CS0 region)

Note: MDMAC should access Flash ROM data in Mirror region.

(1) Used function should be set.

External pin and an external bus should be set.

(2) Descriptor information should be allocated to RAM.

// The allocation address is ***[MDMAxDTAB]*** value + DMA transfer channel number × 0x20.

// In this case, 0x20028000 + 31 × 0x20 = 0x200283E0.

Descriptor	value	
0x200283E0 =	0x00000500	// Transfer type
		// (Increment, Continuous transfer, 32-byte unit, and “Region -> Region”).
0x200283E4 =	0x5E080000	// Transfer source address (Flash ROM data)
0x200283E8 =	0x60000000	// Transfer destination address (CS0 region in the external bus)
0x200283EC =	0x00000400	// Transfer size (1 KB)
0x200283F0 =	0x00000000	// Descriptor address (Chain transfer is unused.)
0x200283F4 =	0x00000000	// Descriptor count (Chain transfer is unused.)

(3) Registers should be set

[MDMAxDTAB] = 0x20028000	// Descriptor base address is set
[MDMAxMSK]<MSK[31]>=1	// ch31 transfer request enable
[MDMAxENE]<ENE[31]>=1	// Interrupt is generated after the completion of transfer.
[MDMAxCEN] = 0x80000000;	// ch31 is enabled (Word access to inhibit RMW.)

(4) Trigger input by external pin

ROM data(1 KB) is transferred to external bus by trigger input.

5.3. Chain Transfer

This is an example of the setting that a chain transfer of 100-byte data (16 bits×50K) is done using TSPI transmission.

(Prerequisite conditions)

- start-up factor: TSPI transmission (DMA transfer channel number=1)
- Transfer source: Flash ROM data (Address = 0x5E080000: Mirror region (Note))
- Transfer destination: TSPI data register (Address = 0x400CB900)
- Descriptor count: 2 (50KB×2)

Note: MDMAC should access Flash ROM data in Mirror region.

(1) Used function should be set.

TSPI transmission and DMA interface should be set.

(2) Descriptor information should be allocated to RAM.

// The allocation address is ***[MDMAxDTAB]*** value + DMA transfer channel number×0x20.

// In this case, 0x20028000 + 1×0x20 = 0x20028020.

Descriptor value	
0x20028020 = 0x00001202	// Transfer type
	// (Increment, Unit transfer, 2-Byte unit, and “Region -> Fixed”)
0x20028024 = 0x5E080000	// Transfer source address (Flash ROM data)
0x20028028 = 0x400CB900	// Transfer destination address (TSPI data register)
0x2002802C = 0x0000C800	// Transfer size (50 KB)
0x20028030 = 0x20028400	// Descriptor address (Next descriptor address)
0x20028034=0x00000002	// Descriptor count (Chain transfer is used twice.)
0x20028400 = 0x00001202	// Transfer type
0x20028404 = 0x5E08C800	// Transfer source address (Flash ROM data)
0x20028408 = 0x400CB900	// Transfer destination address (TSPI data address)
0x2002840C = 0x0000C800	// Transfer size (50 KB)
0x20028410 = 0x00000000	// Descriptor address (Chain transfer completion)

(3) Registers should be set.

[MDMAxDTAB] = 0x20028000	// Descriptor base address is set
[MDMAxMSK]<MSK[1]>=1	// ch1 transfer request enable
[MDMAxENE]<ENE[1]>=1	// Interrupt is generated after the completion of the transfer.
[MDMAxCEN] = 0x00000002	// ch1 is enabled (Word access to inhibit RMW).

(4) DMA transfer request of TSPI transfer completion

TSPI transfer start then DMA transfer request is occurred, ROM data in Flash is read. And it is written to TSPI data register.

5.4. Infinite Transfer

This is an example of the setting that an infinite transfer of 16-bit data is done to RAM using 1-KB ring buffer allocated in RAM and TSPI transmission. An interrupt is not generated after the completion of transfer.

(Prerequisite conditions)

- DMA start-up factor: TSPI transmission (DMA transfer channel number=0)
- Transfer source: TSPI data register (Address=0x400CB900)
- Transfer destination: 1 KB ring buffer in RAM (Start address=0x20028400)
- Descriptor count: Infinite

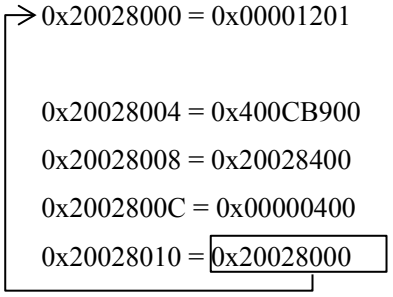
(1) Used function should be set.

SPI reception and DMA interface should be set.

(2) Descriptor information should be allocated to RAM.

// The allocation address is ***[MDMAxDTAB]*** value + DMA transfer channel number×0x20.

// In this case, 0x20028000+0×0x20=0x20028000.



```
Descriptor          value
0x20028000 = 0x00001201     // Transfer type
                            // (Increment, Unit transfer, 2-Byte unit, and “Fixed -> Region”)
0x20028004 = 0x400CB900     // Transfer source address (TSPI data register)
0x20028008 = 0x20028400     // Transfer destination address (1-KB ring buffer in RAM)
0x2002800C = 0x00000400     // Transfer size (1 KB)
0x20028010 = 0x20028000     // Descriptor address
                            // (Next descriptor address: A loop is constructed by the
                            // self-reference.)
0x20028014 = 0x00000100     // Descriptor count (Infinite transfer)
```

(3) Registers should be set.

```
[MDMAxDTAB] = 0x20028000    // Descriptor base address is set
[MDMAxMSK] <MSK[0]>= 1      // ch0 transfer request enable
[MDMAxENE] <ENE[0]>= 0      // Transfer completion interrupt is unused.
[MDMAxCEN] = 0x00000001     // ch0 is enabled (Word access to inhibit RMW)
```

(4) DMA transfer request of TSPI reception completion

When DMA transfer request occurs, data register of TSPI is read. And it is written to ring buffer in RAM.

6. Precaution

The following precautions should be noted when a transfer is done using MDMAC.

6.1. Error Handling

When MDMAC's bus error or descriptor error occurs, the initialization procedure described in "Section 3.10.4" should be done in the error interrupt routine. If the initialization is not done and the interrupt routine completes, a new error interrupt cannot be generated because DMA transfer still stops. It is difficult to detect the occurrence of an error.

6.2. Precaution for UART and FUART

When 9 bit or more data is handled, registers should be accessed with half-word or word unit for both UART and FUART. If byte unit access is done, a transfer request is not generated by UART and FUART, then the transfer stops. Refer to "Asynchronous Serial Communication Circuit(UART)" or "Full Universal Asynchronous Receiver Transmitter Circuit (FUART)" in details.

7. Revision History

Table 7.1 Revision History

Revision	Date	Description
1.0	2018-01-16	First release
2.0	2018-03-27	- 4.1 List of Registers Modified each register name of Saving Descriptor Register -4.2.16 ***[MDMAxMSK]*** Modified bit symbol name. -4.2.17.***[MDMAxCnXFTYP]*** Modified Register Name Title -4.2.18.***[MDMAxCnXFSAD]*** Modified Register Name Title -4.2.19.***[MDMAxCnXFDAD]*** Modified Register Name Title -4.2.20.***[MDMAxCnXFSIZ]*** Modified Register Name Title -4.2.21.***[MDMAxCnDSADS]*** Modified Register Name Title -4.2.22 ***[MDMAxCnDSNUM]*** Modified Register Name Title

RESTRICTIONS ON PRODUCT USE

Toshiba Corporation and its subsidiaries and affiliates are collectively referred to as "TOSHIBA".
Hardware, software and systems described in this document are collectively referred to as "Product".

- TOSHIBA reserves the right to make changes to the information in this document and related Product without notice.
- This document and any information herein may not be reproduced without prior written permission from TOSHIBA. Even with TOSHIBA's written permission, reproduction is permissible only if reproduction is without alteration/omission.
- Though TOSHIBA works continually to improve Product's quality and reliability, Product can malfunction or fail. Customers are responsible for complying with safety standards and for providing adequate designs and safeguards for their hardware, software and systems which minimize risk and avoid situations in which a malfunction or failure of Product could cause loss of human life, bodily injury or damage to property, including data loss or corruption. Before customers use the Product, create designs including the Product, or incorporate the Product into their own applications, customers must also refer to and comply with (a) the latest versions of all relevant TOSHIBA information, including without limitation, this document, the specifications, the data sheets and application notes for Product and the precautions and conditions set forth in the "TOSHIBA Semiconductor Reliability Handbook" and (b) the instructions for the application with which the Product will be used with or for. Customers are solely responsible for all aspects of their own product design or applications, including but not limited to (a) determining the appropriateness of the use of this Product in such design or applications; (b) evaluating and determining the applicability of any information contained in this document, or in charts, diagrams, programs, algorithms, sample application circuits, or any other referenced documents; and (c) validating all operating parameters for such designs and applications. **TOSHIBA ASSUMES NO LIABILITY FOR CUSTOMERS' PRODUCT DESIGN OR APPLICATIONS.**
- **PRODUCT IS NEITHER INTENDED NOR WARRANTED FOR USE IN EQUIPMENTS OR SYSTEMS THAT REQUIRE EXTRAORDINARILY HIGH LEVELS OF QUALITY AND/OR RELIABILITY, AND/OR A MALFUNCTION OR FAILURE OF WHICH MAY CAUSE LOSS OF HUMAN LIFE, BODILY INJURY, SERIOUS PROPERTY DAMAGE AND/OR SERIOUS PUBLIC IMPACT ("UNINTENDED USE").** Except for specific applications as expressly stated in this document, Unintended Use includes, without limitation, equipment used in nuclear facilities, equipment used in the aerospace industry, medical equipment, equipment used for automobiles, trains, ships and other transportation, traffic signaling equipment, equipment used to control combustions or explosions, safety devices, elevators and escalators, devices related to electric power, and equipment used in finance-related fields. **IF YOU USE PRODUCT FOR UNINTENDED USE, TOSHIBA ASSUMES NO LIABILITY FOR PRODUCT.** For details, please contact your TOSHIBA sales representative.
- Do not disassemble, analyze, reverse-engineer, alter, modify, translate or copy Product, whether in whole or in part.
- Product shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable laws or regulations.
- The information contained herein is presented only as guidance for Product use. No responsibility is assumed by TOSHIBA for any infringement of patents or any other intellectual property rights of third parties that may result from the use of Product. No license to any intellectual property right is granted by this document, whether express or implied, by estoppel or otherwise.
- **ABSENT A WRITTEN SIGNED AGREEMENT, EXCEPT AS PROVIDED IN THE RELEVANT TERMS AND CONDITIONS OF SALE FOR PRODUCT, AND TO THE MAXIMUM EXTENT ALLOWABLE BY LAW, TOSHIBA (1) ASSUMES NO LIABILITY WHATSOEVER, INCLUDING WITHOUT LIMITATION, INDIRECT, CONSEQUENTIAL, SPECIAL, OR INCIDENTAL DAMAGES OR LOSS, INCLUDING WITHOUT LIMITATION, LOSS OF PROFITS, LOSS OF OPPORTUNITIES, BUSINESS INTERRUPTION AND LOSS OF DATA, AND (2) DISCLAIMS ANY AND ALL EXPRESS OR IMPLIED WARRANTIES AND CONDITIONS RELATED TO SALE, USE OF PRODUCT, OR INFORMATION, INCLUDING WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY OF INFORMATION, OR NONINFRINGEMENT.**
- Do not use or otherwise make available Product or related software or technology for any military purposes, including without limitation, for the design, development, use, stockpiling or manufacturing of nuclear, chemical, or biological weapons or missile technology products (mass destruction weapons). Product and related software and technology may be controlled under the applicable export laws and regulations including, without limitation, the Japanese Foreign Exchange and Foreign Trade Law and the U.S. Export Administration Regulations. Export and re-export of Product or related software or technology are strictly prohibited except in compliance with all applicable export laws and regulations.
- Please contact your TOSHIBA sales representative for details as to environmental matters such as the RoHS compatibility of Product. Please use Product in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. **TOSHIBA ASSUMES NO LIABILITY FOR DAMAGES OR LOSSES OCCURRING AS A RESULT OF NONCOMPLIANCE WITH APPLICABLE LAWS AND REGULATIONS.**