



PHILIPS

Philips Semiconductors

Connectivity

March 2002

AN10009-01

ISP1581 Using the Odd Bit Indicator for DMA

Rev 1.0

Revision History:

Version	Date	Descriptions	Author
1.0	Feb 2002	First draft	Wei Leong Chui

We welcome your feedback. Send it to wired.support@philips.com



PHILIPS

ISP1581 Using the Odd Bit Indicator for DMA

Rev. 1.0

This is a legal agreement between you (either an individual or an entity) and Philips Semiconductors. By accepting this product, you indicate your agreement to the disclaimer specified as follows:

DISCLAIMER

PRODUCT IS DEEMED ACCEPTED BY RECIPIENT. THE PRODUCT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, PHILIPS SEMICONDUCTORS FURTHER DISCLAIMS ALL WARRANTIES, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANT ABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT. THE ENTIRE RISK ARISING OUT OF THE USE OR PERFORMANCE OF THE PRODUCT AND DOCUMENTATION REMAINS WITH THE RECIPIENT. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL PHILIPS SEMICONDUCTORS OR ITS SUPPLIERS BE LIABLE FOR ANY CONSEQUENTIAL, INCIDENTAL, DIRECT, INDIRECT, SPECIAL, PUNITIVE, OR OTHER DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THIS AGREEMENT OR THE USE OF OR INABILITY TO USE THE PRODUCT, EVEN IF PHILIPS SEMICONDUCTORS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CONTENTS

1. INTRODUCTION	4
2. LOCATION OF THE ODD OR EVEN BIT	4
3. FUNCTIONAL BEHAVIOR OF THE DMA TRANSFER COUNTER AND THE ODD_IND BIT.....	5
3.1. UPDATING OF THE ODD_IND BIT	5
3.2. CLEARING OF THE ODD_IND BIT	5
3.3. TRACKING THE NUMBER OF BYTES TRANSFERRED FROM THE OUT TOKEN	5
3.3.1. <i>Examples: Tracking the Total DMA Bytes Transferred.....</i>	<i>8</i>
4. REFERENCES.....	13

TABLES

Table 2-1: DMA Interrupt Reason Register: (address: 50H)	4
Table 3-1: Calculation of the Total Bytes Tracked by the DMA Counter When Transfer Count _(after) \neq 0	7
Table 3-2: Calculation of the Total Bytes Tracked by the DMA Counter When Transfer Count _(after) = 0	7

FIGURES

Figure 3-1: An example of the 8-bit mode DMA transfer counter starting with an odd byte.....	5
Figure 3-2: An example of the 8-bit mode DMA transfer counter starting with an even byte.	6
Figure 3-3: An example of the 16-bit mode DMA transfer counter starting with an odd byte.	6
Figure 3-4: An example of the 16-bit mode DMA transfer counter starting with an even byte.....	7

1. Introduction

The odd bit is used as an indicator to allow the firmware to read the exact number of data bytes that are transferred across the direct memory access (DMA) transfer counter (34H) from the ISP1581 to the external DMA bus. This odd bit indicator is valid only for the OUT token data.

The implementation will track whether an odd number has been transferred from the last OUT token packet to the DMA bus.

2. Location of the Odd or Even Bit

The bit is the bit 12 (ODD_IND) of the DMA Interrupt Reason register, which is located at the address 50H (see Table 2-1). When the bit is set, it means that odd bytes have been transferred from the last OUT token packet to the DMA.

Table 2-1: DMA Interrupt Reason Register: (address: 50H)

Bit	Symbol	Description
15 to 13	—	Reserved
12	ODD_IND	Logic 1 indicates that the last packet with odd bytes have been transferred from the OUT token buffer to the DMA. This is applicable only for the OUT token data in the DMA slave mode. It has no meaning for the IN token data.
11	EXT_EOT	Logic 1 indicates that an external EOT is detected. This is applicable only in the GDMA slave mode
10	INT_EOT	Logic 1 indicates that an internal EOT is detected.
9	INTRQ_PENDING	Logic 1 indicates that a pending interrupt was detected on pin INTRQ.
8	DMA_XFER_OK	Logic 1 indicates that the DMA transfer has been completed (The DMA transfer counter has become zero.). This bit is only used in the GDMA (slave) mode and the MDMA (master) mode.
7	1F0_WF_E	Logic 1 indicates that the 1F0 write FIFO is empty and the microcontroller can start writing data.
6	1F0_WF_F	Logic 1 indicates that the 1F0 write FIFO is full and the microcontroller must stop writing data.
5	1F0_FR_E	Logic 1 indicates that the 1F0 read FIFO is empty and the microcontroller must stop reading data.
4	READ_1F0	Logic 1 indicates that 1F0 FIFO contains unread data and the microcontroller can start reading data
3	BSY_DONE	Logic 1 indicates that the BSY status bit has become zero and polling has been stopped.
2	TF_RD_DONE	Logic 1 indicates that the Read Task Files command has been completed.
1	CMD_INTRQ_OK	Logic 1 indicates that all bytes from the FIFO have been transferred (DMA Transfer Count zero) and an interrupt on pin INTRQ was detected.
0	—	Reserved

3. Functional Behavior of the DMA Transfer Counter and the ODD_IND Bit

3.1. Updating the ODD_IND Bit

The ODD_IND bit is not updated in real time. That an odd number of bytes have been transferred will not be reflected in this bit. Only when there is a DMA interrupt *and* the INT_EOT bit is asserted (this happens when an internal EOT is detected), then will this bit be updated.

3.2. Clearing the ODD_IND Bit

To clear the ODD_IND bit, write a "1". As the ODD_IND bit is automatically cleared whenever a new OUT packet arrives, it is not necessary to clear this bit every time you restart the DMA. Therefore, this bit will always accurately indicate whether an odd number of bytes have passed through the DMA bus from the start of the final packet to the point before the assertion of the INT_EOT bit (bit 10, 50H, see Table 2-1).

3.3. Tracking the Number of Bytes Transferred from the OUT Token

The DMA transfer counter behaves as follows:

8-bit DMA mode: The DMA transfer counter will decrement twice on every two data bytes that are strobed OUT/IN through the DMA.

Here are the possible scenarios:

When the total DMA transfer counter starts with an odd byte count:

The example in Figure 3-1 shows a DMA OUT token of 9 bytes and the DMA transfer counter decrementing.

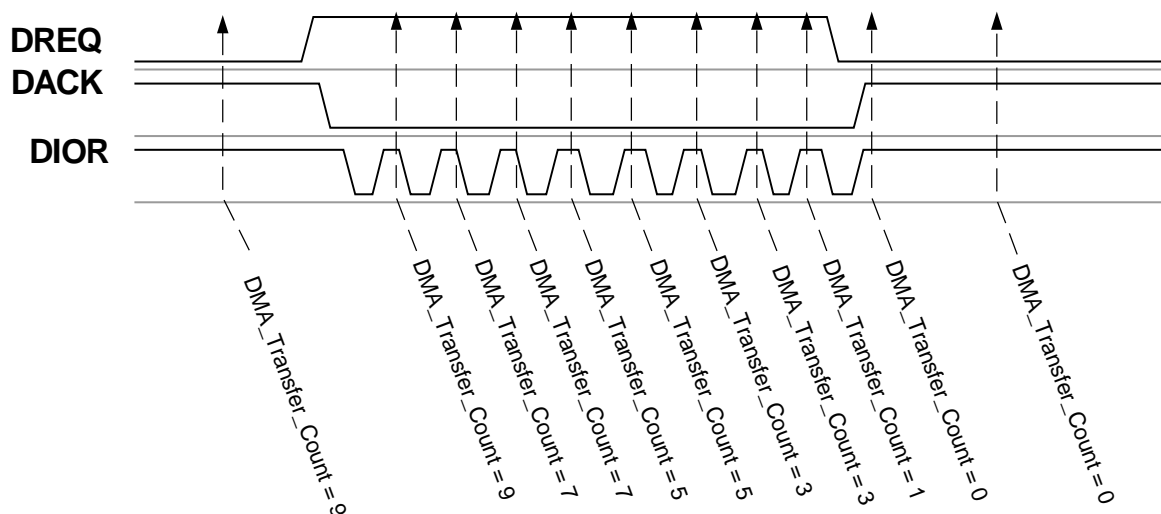


Figure 3-1: An example of the 8-bit mode DMA transfer counter starting with an odd byte.

ISP1581 Using the Odd Bit Indicator for DMA

Rev. 1.0

When the total DMA transfer counter starts with an even byte count.

The example in Figure 3-2 shows a DMA OUT token of 10 bytes and the DMA transfer counter decrementing.

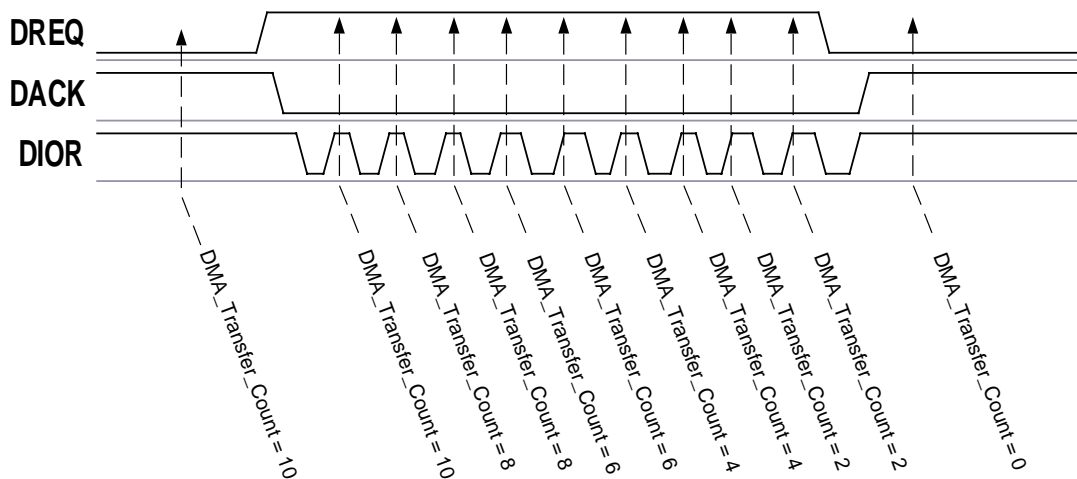


Figure 3-2: An example of the 8-bit mode DMA transfer counter starting with an even byte.

16-bit DMA mode: The DMA transfer counter will decrement twice on every word that is strobed OUT/IN through the DMA.

When the total DMA transfer counter starts with an odd byte count:

Figure 3-3 shows an example of a DMA OUT token of 19 bytes and the DMA transfer counter decrementing.

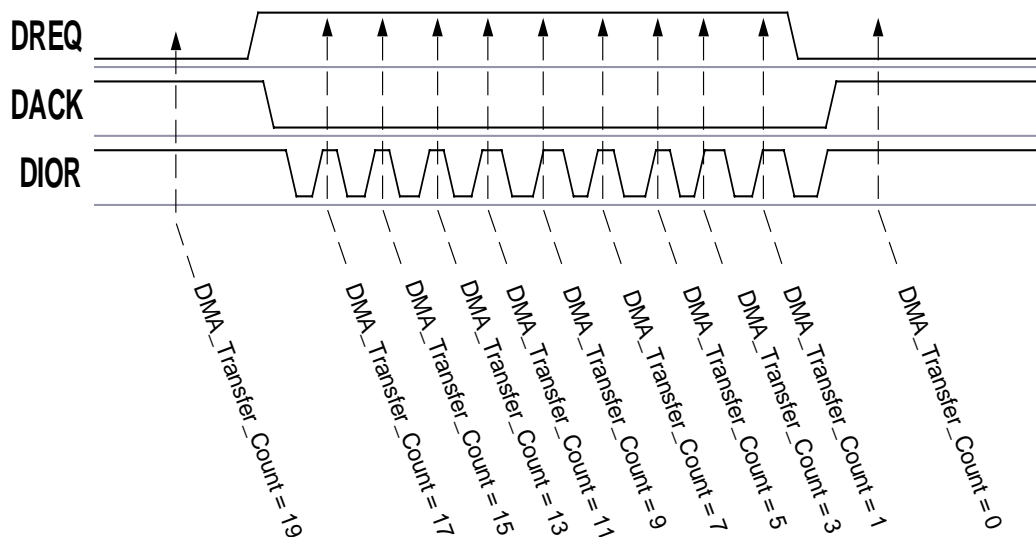


Figure 3-3: An example of the 16-bit mode DMA transfer counter starting with an odd byte.

ISP1581 Using the Odd Bit Indicator for DMA

Rev. 1.0

When the total DMA transfer counter starts with an even byte count.

The example in Figure 3-4 shows a DMA OUT token of 20 bytes and the DMA transfer counter decrementing.

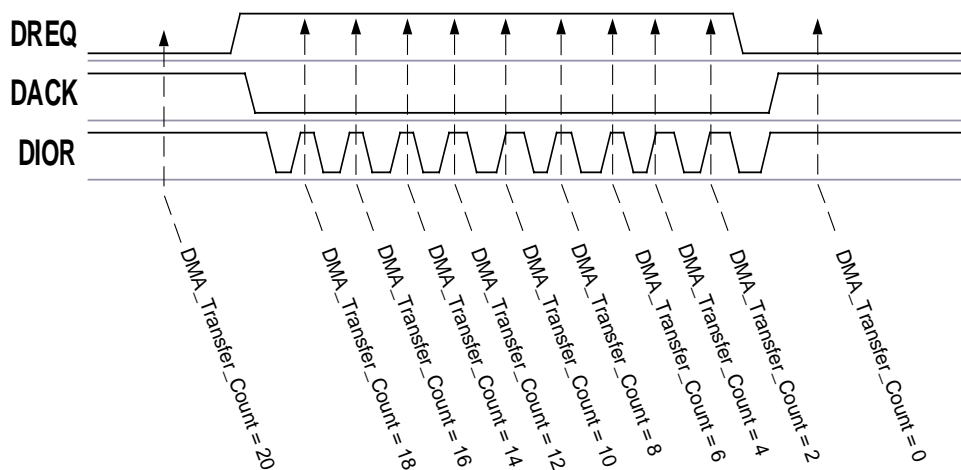


Figure 3-4: An example of the 16-bit mode DMA transfer counter starting with an even byte.

Therefore, the total number of bytes transferred can be calculated, depending on the final value in the DMA transfer counter. If the final value in the DMA transfer counter is non-zero, the total number of bytes tracked by the DMA counter can be calculated based on Table 3-1.

Table 3-1: Calculation of the Total Bytes Tracked by the DMA Counter When Transfer Count_(after) != 0

ODD_IND	Total Number of Bytes Transferred	
	8-bit DMA Mode	16-bit DMA Mode
1	Transfer Count _(before) - (Transfer Count _(after) - 1)	Transfer Count _(before) - (Transfer Count _(after) + 1)
0	Transfer Count _(before) - Transfer Count _(after)	Transfer Count _(before) - Transfer Count _(after)

If the final value in the DMA transfer counter is zero, the total number of bytes tracked by the DMA counter can be calculated based on Table 3-2.

Table 3-2: Calculation of the Total Bytes Tracked by the DMA Counter When Transfer Count_(after) = 0

Is Transfer Count _(before) an odd number?	ODD_IND	Total Bytes Transferred	
		8-bit DMA Mode	16-bit DMA Mode
Yes	1	Transfer Count _(before)	Transfer Count _(before)
Yes	0	Transfer Count _(before)	Cannot happen
No	1	Transfer Count _(before)	Transfer Count _(before) - 1
No	0	Transfer Count _(before)	Transfer Count _(before)

ISP1581 Using the Odd Bit Indicator for DMA

Rev. 1.0

3.3.1. Examples: Tracking the Total Number of DMA Bytes Transferred

8-bit DMA

The DMA Transfer Counter register counts the number of bytes transferred from the host to the ISP1581. The DMA transfer counter decrements twice whenever two bytes are strobed out.

8-bit DMA CASE I

Consider a case in which 1024 bytes are to be transferred to the DMA by using the OUT token using transfer counter. For a packet size of 512 bytes, there will be only two OUT tokens. The following example shows the DMA transfer counter set to the exact number of bytes to be sent:

```
DMA_Transfer_Counter = 1024;

OUT
<0 - 511 Bytes>

OUT
<512 - 1023 Bytes>

Value = DMA_Transfer_Counter;
printf ("Value = %d");    /// Value = 0;

Value = DMA_Interrupt_Reason;
printf ("Value = %x");    /// Value = 0100H; DMA_XFER_OK is set, ODD_IND is cleared
```

Total number of bytes transferred = $1024_{(\text{before})} - 0_{(\text{after})} = 1024$.

8-bit DMA CASE II

Consider a case in which 1026 bytes are to be transferred to the DMA by using the OUT token using the transfer counter. For a packet size of 512 bytes, there will be only three OUT tokens. The following example shows the DMA transfer counter set to the exact number of bytes to be sent:

```
DMA_Transfer_Counter = 1026;

OUT
<0 - 511 Bytes>

OUT
<512 - 1023 Bytes>

OUT
<1024 - 1025 Bytes>

Value = DMA_Transfer_Counter;
printf ("Value = %d");    /// Value = 0;

Value = DMA_Interrupt_Reason;
printf ("Value = %x");    /// Value = 0500H; DMA_XFER_OK is set, ODD_IND is cleared,
                          /// INT_EOT is set
```

Total number of bytes transferred = $1026_{(\text{before})} - 0_{(\text{after})} = 1026$.

8-bit DMA CASE III

Consider a case in which 1024 bytes are to be transferred to the DMA by using the OUT token using the transfer counter. For a packet size of 512 bytes, there will be only three OUT tokens. The following example shows the DMA transfer counter set to a size larger than the current number of bytes sent. A residual is left in the DMA transfer counter.

```
DMA_Transfer_Counter = 2048;

OUT
<0 - 511 Bytes>
```

ISP1581 Using the Odd Bit Indicator for DMA
Rev. 1.0

```

OUT
<512 - 1023 Bytes>

Value = DMA_Transfer_Counter;
printf ("Value = %d");    /// Value = 1024;

Value = DMA_Interrupt_Reason;
printf ("Value = %x");    /// Value = 0000H; DMA_XFER_OK is cleared, ODD_IND is cleared

```

Total number of bytes transferred = $2048_{(\text{before})} - 1024_{(\text{after})} = 1024$.

8-bit DMA CASE IV

Consider a case in which 1026 bytes are to be transferred to the DMA by using the OUT token using the transfer counter. For a packet size of 512 bytes, there will be only three OUT tokens. The following example contains the DMA transfer counter set to a size larger than the current number of bytes sent. A residual is left in the DMA transfer counter.

```

DMA_Transfer_Counter = 2048;

OUT
<0 - 511 Bytes>

OUT
<512 - 1023 Bytes>

OUT
<1024 - 1025 Bytes>

Value = DMA_Transfer_Counter;
printf ("Value = %d");    /// Value = 1022;

Value = DMA_Interrupt_Reason;
printf ("Value = %x");    /// Value = 0400H; DMA_XFER_OK is cleared, ODD_IND is cleared,
                          /// INT_EOT is set

```

Total number of bytes transferred = $2048_{(\text{before})} - 1022_{(\text{after})} = 1026$.

8-bit DMA CASE V

Consider a case in which 1025 bytes are to be transferred to the DMA by using the OUT token using transfer counter. For a packet size of 512 bytes, there will be only three OUT tokens.

```

DMA_Transfer_Counter = 1026;

OUT
<0 - 511 Bytes>

OUT
<512 - 1023 Bytes>

OUT
<1024 Byte>

Value = DMA_Transfer_Counter;
printf ("Value = %d");    /// Value = 2;

Value = DMA_Interrupt_Reason;
printf ("Value = %x");    /// Value = 1500H; DMA_XFER_OK is set, ODD_IND is set,
                          /// INT_EOT set

```

Total number of bytes transferred = $1026_{(\text{before})} - (2_{(\text{after})} - 1) = 1025$.

Consider a case in which 1025 bytes are to be transferred to the DMA by using the OUT token using transfer counter. For a packet size of 512 bytes, there will be only three OUT tokens. The following example shows the DMA transfer counter set to the exact number of bytes to be sent.

```

DMA_Transfer_Counter = 1025;

```

ISP1581 Using the Odd Bit Indicator for DMA
Rev. 1.0

```

    OUT
    <0 - 511 Bytes>

    OUT
    <512 - 1023 Bytes>

    OUT
    <1024 Byte>

    Value = DMA_Transfer_Counter;
    printf ("Value = %d");    /// Value = 0;

    Value = DMA_Interrupt_Reason;
    printf ("Value = %x");    /// Value = 1500H; DMA_XFER_OK is set, ODD_IND is set,
    /// INT_EOT set

```

Total number of bytes transferred = 1025_(before) – 0_(after) = 1025.

8-bit DMA CASE VI

Consider a case in which 1025 bytes are to be transferred to the DMA by using the OUT token using transfer counter. For a packet size of 512 bytes, there will be only three OUT tokens. The following example contains the DMA transfer counter set to a size larger than the current number of bytes sent. A residual is left in the DMA transfer counter.

```

    DMA_Transfer_Counter = 2048;

    OUT
    <0 - 511 Bytes>

    OUT
    <512 - 1023 Bytes>

    OUT
    <1024 Bytes>

    Value = DMA_Transfer_Counter;
    printf ("Value = %d");    /// Value = 1024;

    Value = DMA_Interrupt_Reason;
    printf ("Value = %x");    /// Value = 1400H; DMA_XFER_OK is cleared, ODD_IND is set,
    /// INT_EOT is set

```

Total Bytes Transferred = 2048_(before) – (1024_(after) – 1) = 1025.

16-bit DMA

The DMA Transfer Counter register counts the number of bytes transferred from the host to the ISP1581. The DMA transfer counter decrements once for every word strobed out.

16-bit DMA CASE I

Consider a case in which 1024 bytes are to be transferred to the DMA by using the OUT token using transfer counter. For a packet size of 512 bytes, there will be only two OUT tokens. The following example shows the DMA transfer counter set to the exact number of bytes to be sent:

```

    DMA_Transfer_Counter = 1024;

    OUT
    <0 - 511 Bytes>

    OUT
    <512 - 1023 Bytes>

    Value = DMA_Transfer_Counter;
    printf ("Value = %d");    /// Value = 0;

    Value = DMA_Interrupt_Reason;
    printf ("Value = %x");    /// Value = 0100H; DMA_XFER_OK is set, ODD_IND is cleared

```

ISP1581 Using the Odd Bit Indicator for DMA

Rev. 1.0

Total number of bytes transferred = $1024_{(before)} - 0_{(after)} = 1024$.

16-bit DMA CASE II

Consider a case in which 1026 bytes are to be transferred to the DMA by using the OUT token using transfer counter. For a packet size of 512 bytes, there will be only three OUT tokens. The following example shows the DMA transfer counter set to the exact number of bytes to be sent:

```
DMA_Transfer_Counter = 1026;

OUT
<0 - 511 Bytes>

OUT
<512 - 1023 Bytes>

OUT
<1024 - 1025 Bytes>

Value = DMA_Transfer_Counter;
printf ("Value = %d");    /// Value = 0;

Value = DMA_Interrupt_Reason;
printf ("Value = %x");    /// Value = 0500H; DMA_XFER_OK is set, ODD_IND is cleared,
                          /// INT_EOT is set
```

Total Bytes Transferred = $1026_{(before)} - 0_{(after)} = 1026$.

16-bit DMA CASE III

Consider a case in which 1024 bytes are to be transferred to the DMA by using the OUT token using transfer counter. For a packet size of 512 bytes, there will be only two OUT tokens. The following example contains the DMA transfer counter set to a size larger than the current number of bytes sent. A residual is left in the DMA transfer counter.

```
DMA_Transfer_Counter = 2048;

OUT
<0 - 511 Bytes>

OUT
<512 - 1023 Bytes>

Value = DMA_Transfer_Counter;
printf ("Value = %d");    /// Value = 1024;

Value = DMA_Interrupt_Reason;
printf ("Value = %x");    /// Value = 0000H; DMA_XFER_OK is cleared, ODD_IND is cleared
```

Total number of bytes transferred = $2048_{(before)} - 1024_{(after)} = 1024$.

16-bit DMA CASE IV

Consider a case in which 1026 bytes are to be transferred to the DMA by using the OUT token using transfer counter. For a packet size of 512 bytes, there will be only three OUT tokens. The following example contains the DMA transfer counter set to a size larger than the current number of bytes sent. A residual is left in the DMA transfer counter.

```
DMA_Transfer_Counter = 2048;

OUT
<0 - 511 Bytes>

OUT
<512 - 1023 Bytes>

OUT
```

ISP1581 Using the Odd Bit Indicator for DMA
Rev. 1.0

```

    <1024 - 1025 Bytes>

    Value = DMA_Transfer_Counter;
    printf ("Value = %d");    /// Value = 1022;

    Value = DMA_Interrupt_Reason;
    printf ("Value = %x");    /// Value = 0400H; DMA_XFER_OK is cleared, ODD_IND is cleared,
    /// INT_EOT is set

```

Total number of bytes transferred = $2048_{(\text{before})} - 1022_{(\text{after})} = 1026$.

16-bit DMA CASE V

Consider a case in which 1025 bytes are to be transferred to the DMA by using the OUT token using transfer counter. For a packet size of 512 bytes, there will be only three OUT tokens.

```

    DMA_Transfer_Counter = 1025;

    OUT
    <0 - 511 Bytes>

    OUT
    <512 - 1023 Bytes>

    OUT
    <1024 Byte>

    Value = DMA_Transfer_Counter;
    printf ("Value = %d");    /// Value = 0;

    Value = DMA_Interrupt_Reason;
    printf ("Value = %x");    /// Value = 1500H; DMA_XFER_OK is set, ODD_IND is set,
    /// INT_EOT is set

```

Total number of bytes transferred = $1025_{(\text{before})} - 0_{(\text{after})} = 1025$.

Consider a case in which 1025 bytes are to be transferred to the DMA by using the OUT token using transfer counter. For a packet size of 512 bytes, there will be only three OUT tokens. The following example shows the DMA transfer counter set to the exact number of bytes to be sent:

```

    DMA_Transfer_Counter = 1026;

    OUT
    <0 - 511 Bytes>
    OUT
    <512 - 1023 Bytes>

    OUT
    <1024 Byte>

    Value = DMA_Transfer_Counter;
    printf ("Value = %d");    /// Value = 0;

    Value = DMA_Interrupt_Reason;
    printf ("Value = %x");    /// Value = 1500H; DMA_XFER_OK is set, ODD_IND is set,
    /// INT_EOT is set

```

Total number of bytes transferred = $1026_{(\text{before})} - (0_{(\text{after})} + 1) = 1025$.

16-bit DMA CASE VI

Consider a case in which 1025 bytes are to be transferred to the DMA by using the OUT token using transfer counter. For a packet size of 512 bytes, there will be only three OUT tokens. The following example contains the DMA transfer counter set to a size larger than the current number of bytes sent. A residual is left on the DMA transfer counter.

```

    DMA_Transfer_Counter = 2048;

    OUT

```

ISP1581 Using the Odd Bit Indicator for DMA

Rev. 1.0

```
<0 - 511 Bytes>
OUT
<512 - 1023 Bytes>
OUT
<1024 Bytes>
Value = DMA_Transfer_Counter;
printf ("Value = %d");    /// Value = 1022;
Value = DMA_Interrupt_Reason;
printf ("Value = %x");    /// Value = 1400H; DMA_XFER_OK is cleared, ODD_IND is set,
                        /// INT_EOT is set
```

Total number of bytes transferred = $2048_{(\text{before})} - (1022_{(\text{after})} + 1) = 1025$.

4. References

- *Universal Serial Bus Specification Rev. 2.0*
- *ISP1581 Universal Serial Bus 2.0 high-speed interface device datasheet.*