



Revision history

Rev	Date	Description
3.0	June 2004	Added errata 4. Changed "Device Controller" to "Peripheral Controller"
2.0	Mar 2002	Errata on empty packet added
1.0	Sep 2001	DMA errata (counter and EOT interrupt)

Contact information

For additional information, please visit: <http://www.semiconductors.philips.com/>

For sales office addresses, please send an email to: sales.addresses@www.semiconductors.philips.com



Errata 1: In the 16-bit mode, the DMA counter does not reach zero.

1.1 Problem Description

In the 16-bit mode, after the normal DMA transfer, an EOT interrupt will be received from ISP1181A. At this time, the DMA counter value is 1. In principle, however, it is supposed to be 0, after a normal DMA transfer.

The following test cases will explain the problem in detail:

In the case of an OUT transaction:

- The DMA counter is set to 512.
- The host sends 512 bytes.
- The EOT interrupt is received.
- The DMA Counter value is 1 but not 0. In principle, the counter value must be 0 because all the 512 bytes have been successfully transferred.

Remark: There is no error in the data transferred. This problem exists only in the 16-bit mode and not in the 8-bit mode.

1.2 Implication

The problem occurs if the host sends 511 bytes (instead of 512 bytes), forcing the ISP1181A to generate a short packet EOT. At this time, when the DMA counter is read, 1 will still be shown, making it impossible to differentiate between a short packet (less than 1 byte) EOT and normal EOT. The occurrence of a short packet with 1 byte less, however, is very rare. If the number of missing bytes in the short packet is more than 1 byte of the designated size, it can be differentiated by reading the DMA counter value. Therefore, the problem occurs only when there is a short packet with 1 byte less.

1.3 Workaround

There is no firmware workaround. The occurrence of such a case is very rare. Therefore, if it is really necessary, the external DMA Controller (master) counter can be used.

1.4 Status

This issue is fixed in the ISP1181B. A bit in the Interrupt register is used to denote the Short Packet EOT interrupt.

Errata 2: *In the DMA mode, an EOT interrupt is sometimes missing during the transfer.*

2.1 Problem Description

Sometimes it is noticed that the ISP1181A does not generate an EOT interrupt, after completing a normal DMA transfer. The data integrity is not lost. The missing EOT interrupt is because of the strict design constraint, which does not effectively consider jitter conditions.

2.2 Implication

The implication is serious. If the design is based on the internal EOT generated from the ISP1181A, then the design cannot be reliable because sometimes the microcontroller cannot get an interrupt even, after a proper DMA transfer.

2.3 Workaround

Since the internal EOT of the ISP1181A is not reliable, the external DMA Controller (master) counter can be used to know the EOT condition and the DMA transfer can be stopped using either of the following two methods:

- Once the DMA counter (master) has reached zero, the DMA master can assert the EOT signal to the ISP1181A to end the DMA transfer.
- Or
- Once the DMA counter (master) has reached zero, the microcontroller can deassert the DMAEN bit in the DMA Configuration register of the ISP1181A. This in turn disables the DMA.

2.4 Status

Fixed in ISP1181B.

Errata 3: *During the DMA transfer, empty packets are sometimes sent for an IN token.*

3.1 Problem Description

In the DMA mode, for an IN token sometimes an empty packet is generated and sent to the host because an empty packet is written to the buffer when the following two events occur at the same time:

- The ISP1181A receives an ACK from the Host Controller
and

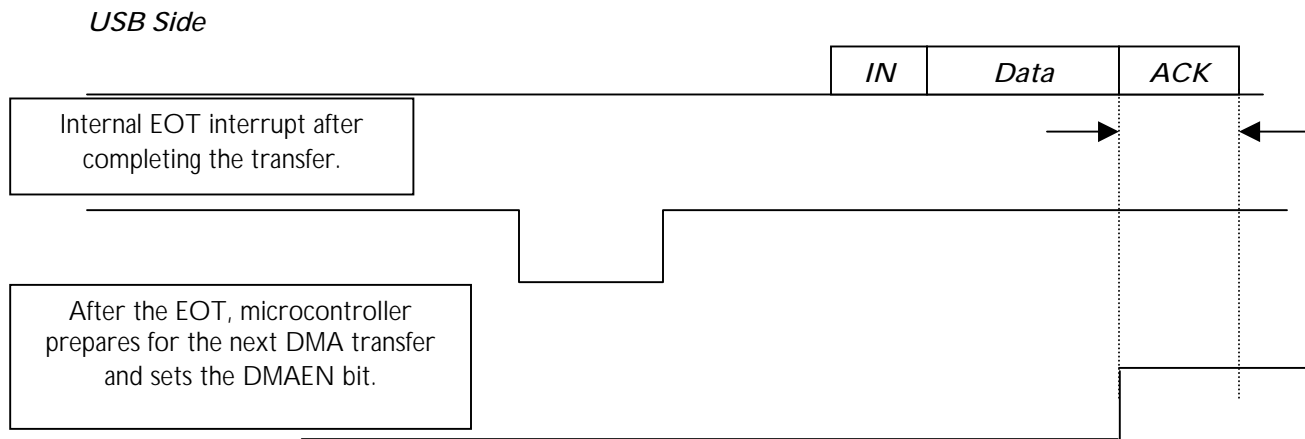
- The microcontroller sets the DMAEN bit in the DMA Configuration register.

When these two events occur at the same time, an empty packet is automatically written to the internal buffer, which in turn will be sent out for the next IN token.

In a normal application, the scenario will be as follows:

- 1) The microcontroller sets the DMA counter, disables the endpoint interrupts (EOT interrupt enabled) and enables the DMA transfer by setting the DMAEN bit.
- 2) The DMA transfer starts (writing to the IN endpoint) and once the DMA transfers the number of bytes as programmed in the DMA Counter register, an internal EOT interrupt is generated.
- 3) The microcontroller prepares itself for the next DMA transfer by setting the DMAEN bit. During this time, the host might receive the last packet of the previous DMA transfer and ACKs.

The events are given in the following diagram:



The preceding events occur inside the ISP1181A. The dotted lines indicate the coincidence of these events.

3.2 Implication

The implication is serious in certain applications in which the empty packet is considered as an error condition or termination of the transfer. For example, in certain mass storage implementations, the empty packet is considered as an error condition and the Host Controller will try to send a command to rectify the error, which cannot be seen by the firmware. This is because the microcontroller has already set the DMAEN bit, activating the DMA transfer and has also disabled endpoint interrupts to reduce the overhead. In view of this, whatever the Host Controller sends for error recovery will not be seen and the system stops responding.

3.3 Workaround

To avoid the coincidence of the two events, the firmware must check whether the corresponding DMA IN endpoint is empty (in the case of double buffering, check whether both

the buffers are empty) before setting the DMAEN bit in the DMA Configuration register. This ensures that the data from the IN endpoint is already sent to the Host Controller, and the ACK from the Host Controller is received well before the DMAEN bit is set.

3.4 Status

The firmware workaround is a proven one with little overhead on the CPU.

Errata 4: Data corruption during write operation to the ISP1181A Peripheral Controller.

4.1 Problem Description

After a data write operation, the ISP1181A Peripheral Controller requires a 132 ns (min) delay before a write assertion can be issued for other devices. This must be fulfilled even after CS_N is deasserted (See Figure 1 and Figure 2).

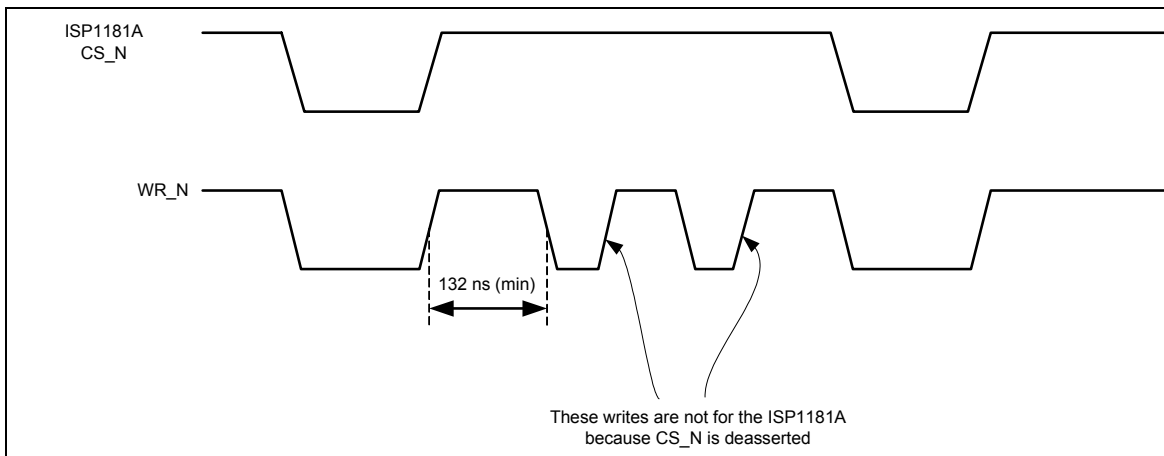


Figure 1

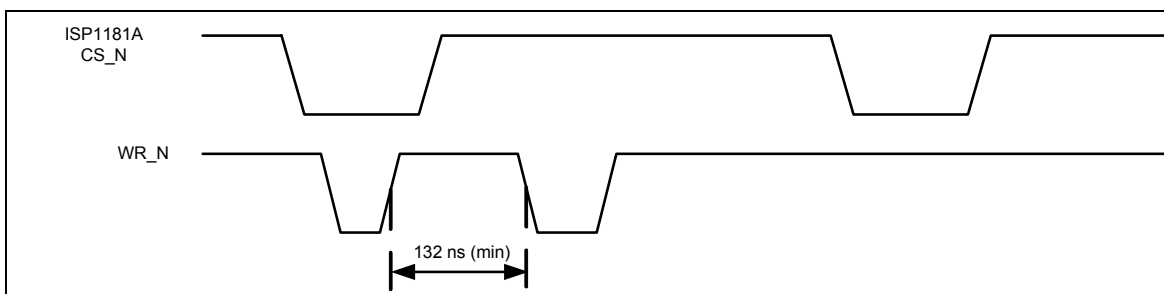


Figure 2

4.2 Implications

Will lead to data corruption if the timing requirements in Figure 1 and Figure 2 are not handled.

4.3 Workaround

Make sure that the system can handle the write timing requirements as given in Figure 1 and Figure 2. If the system is really fast and needs immediate write accesses to other devices, then it is better to qualify WR_N with respect to CS_N and provide the resultant write signal to the ISP1181A (see Figure 3).

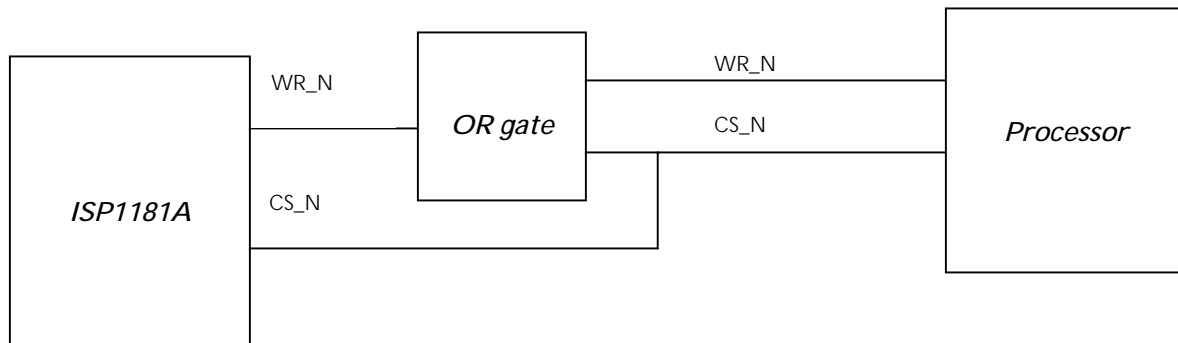


Figure 3

Note: Please take into consideration the propagation delay of the logics.

4.4 Status

No fix is planned.