OPTi

82C465MV Product Update

1 August 94

To:	All OPTi 82C465MV Customers
From:	Mark R. Williams, Technical Marketing Manager, (408) 486-8412
Subject:	82C465MV DMA/SMI Failure

There has been a bug reported in the production 82C465MV silicon that involves DMA transfers that occur during SMM. This problem affects ALL systems that include DMA peripheral devices.

DMA Failure During SMM

When the chipset system is operating in SMM (SMIACT# active), a DMA transfer to or from a memory address location with the same segment as SMBASE will access SMM memory space, not the intended memory buffer in system DRAM. For DMA reads, the wrong data will be transferred to the DMA peripheral device. For DMA writes, the SMM space will be overwritten and corrupted. The 82C465MV chip fails this operation because it does not internally qualify SMIACT# with HLDA and so always remaps SMBASE addresses to SMM memory.

For example: if a DMA write transfer to a buffer at address 3000:200h is in progress and an SMI occurs, the 82C465MV chip will go to SMM. If the SMBASE is at its default value of 3000:0h, the 82C465MV logic remaps all 3000h segment accesses to B000h because SMIACT# is low. However, on the next DMA write transfer, SMIACT# remains low and an access to 3000:200h arrives at B000:200h.

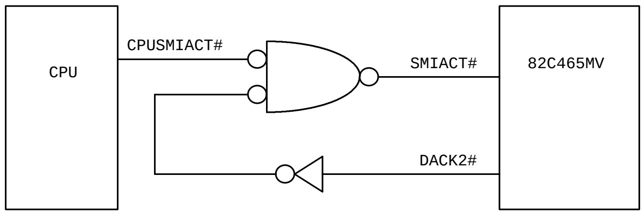
The problem can be corrected in hardware or software. OPTi recommends the hardware fix at this time. Note, however, that neither solution has been fully tested at this time (since the problem was only recently discovered). OPTi will provide test results at the earliest possible opportunity.

Hardware Solution

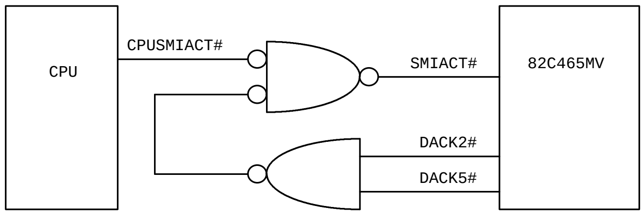
It is necessary to externally qualify SMIACT# from the CPU with any DMA Acknowledge (DACK0-7#) signals in use from the 82C465MV chip using an OR gate. This solution prevents the 82C465MV logic from remapping the address by masking SMIACT# during DMA transfers.

There is a drawback to this solution. The 82C465MV logic internally uses SMIACT# to hold off further SMI# activity. When the logic sees SMIACT# toggle, it will also toggle SMI# if any PMI events are still active and so will preset the CPU for another entry into SMM. Therefore, at the end of the current SMI service, the CPU will again be triggered into SMM. The second entry will be spurious and software will see no PMIs active, so no harm is done and no software changes are needed. The process incurs additional SMI service overhead for the extra SMI, but the extra SMI occurs only if DMA transfer has taken place during the first SMI.

The circuit shown below handles only DMA channel 2, the most commonly used channel.



A circuit to handle two DMA channels would require the components shown below.



A circuit to handle all DMA channels is better handled by monitoring the DACKMUX2-0 outputs. Only if these outputs drive 100b, corresponding to "DACK4#" being low, should SMIACT# be allowed to pass to the 82C465MV chip. If a discrete DACKMUX decoder is used in the system design, the "DACK4#" output of the decoder is convenient to use for this purpose.

Software Solution

The failure occurs only when the DMA buffer is in the same memory address segment as SMBASE. Therefore, if practical, SMBASE can be reprogrammed, both on the CPU and in the 82C465MV part, to a segment value where DMA is known not to occur. For example, a base segment of 0h could be chosen if the assumption is made that the operating system will always occupy segments 0 and 1 and will never attempt to establish DMA buffers in these segments. Accesses to segments 0000 and 1000h would still be remapped to B000h and A000h during SMM.

Before choosing this solution, make sure that the operating systems being considered will **never** establish DMA buffers in the selected range.

Long-Term Solution

This problem has been corrected in the 82C465MV/A part, which will be in production in Q4 of '94 or Q1 of '95.