Freescale Semiconductor, Inc.

MOTOROLA

Consumer Systems Group

Order this document by
MC68040UMAD/AD

MC68040

Addendum to
MC68040
User Manual

June 22, 1998

This addendum to the initial release of the MC68040UM/AD User's Manual provides corrections to the original text, plus additional information not included in the original. This document and other information on this product is maintained on the World Wide Web at http://sps.motorola.com/hpes/

Theta Jc for FE (Ceramic Quad) package:

Theta Jc for the FE package is the same as for the Pin Grid Array (RC) package: 3 Degrees Centigrade/Watt.

PGA Drawing correction:

Page 12-2, 12-3, 12-5: PGA drawing shows pin S6 correctly as MDIS*. Table at bottom of page incorrectly lists pin S6 as Internal Logic GND.

Erroneous Bus Arbitration Examples:

Page 7-52 to 7-58: Section 8.2 Bus Arbitration Examples section contains many errors in various figures. Please ignore this section if it confuses the reader.

State Machine Transition:

Page 7-47 Figure 7-30: State machine should indicate that a transistion from the Implicit Ownership state to the OWN/PARK state be caused by the condition: BG (bus grant) asserted and IBR (internal bus request) asserted.

Access Error Stack Frame Combonations:

Page 8-31 Table 8-6: In the column "Hard Cleanup Action", the comment "Write PD3-0 and skip" applies only to the the Write Page Fault case in which a MOVE16 (2M16=1)in the WB2S.

General Operation:

The following items provide clarification to the operation of the 68040. The description of these items takes precedence over any previous description contained in the M68040 User's Manual.

1.During instruction line fetches, theTLN pins are not valid. However, they are valid for data line fetches.

2.The RESET instruction causes the PSTx signals to indicate exception stacking instead of START/ CONTINUE encoding.

3.Under certain conditions, the processor asserts BR* and negates it later without executing a bus cycle. External arbiter designs take into account this possible behavior.

If in supervisor mode, a stack frame is restored which has an odd address PC and an SR which indicates user trace mode, then an address error is taken, but the SR stacked for the address error has the supervisor bit set in the SR. For the 030, the supervisor bit is clear.

4.In a two, '040-type master system when a master which implicitly owned the bus has its BG* negated it tristates the address and attribute lines it was driving in the next clock period. However, if another '040-type master is given the bus with its BG* asserted at the same time the previous master has its BG* negated, this master can start driving its address and attribute lines at the same time the previous master is trying to tristate them. External arbiters must be designed such that there is at least one rising BCLK edge in which all bus masters have their respective BG* inputs negated when transferring bus ownership to another master.

5.A locked access can erroneously hit in the data cache, even though the data cache is disabled. When the data cache is re-enabled at a later time, the first non-locked access that misses in the data cache will not be allocated in the data cache. Although it is unclear what impact this has on typical systems, to avoid encountering this item, it is best to invalidate the data cache when it is disabled.

6. When a TLB or TTR write protect fault is taken on the MOVE16 write, in which the MOVE16 source is dirty in the cache, and that the source and destination addresses for the MOVE16 are the same, (specifically, a MOVE16 (Ax)+,(Ay)+ instruction in which Ax=Ay), the associated dirty cache line is invalidated and lost.

7.If a LOCKED access (TAS/CAS instruction) hits a dirty entry in the data cache, it invalidates the matching valid entry and pushes it out. The 040 will begin asserting the external LOCK* pin on this cache push. Strictly speaking, the proper behavior of the 040 should be to perform the cache line push first before asserting the LOCK* signal. A workaround for systems that cannot tolerate long periods of time in which the LOCK* is asserted, is to place semaphores in cache-inhibited space to avoid this situation.

8.Bus error (TEA* asserted) on cache line push within the neighborhood of an interrupt exception processing sequence may cause a Spurious Interrupt Exception instead of an Access Error Exception.The following sequence describes the problem:

1) IPLx* lines are asserted to signal an interrupt request.

2) IPEND* is asserted to signal a pending interrupt processing.

3) A cache push of a data cache line occurs. This cache push to memory results in a physical bus error (TEA* asserted). The PSTx=$F indicates exception stacking, although the 040 has not yet begun exception processing or exception stacking for the interrupt.

4) The 040 starts exception stacking for a Spurious Interrupt Exception.

In the strictest definition of a Spurious Interrupt, there is only one case in which the Spurious Interrupt Exception is signaled: physical bus error during an Interrupt Acknowledge Cycle. In this case, a bus error terminates a cache line push access instead of an Interrupt Acknowledge cycle. However, a cache line push access should be rare, if ever, in a typical system environment. The fact that the cache line resided in the cache prior to its push indicates that the physical memory associated with that cache line had been coherent and accessible at the time of its allocation into the cache. The workaround is treat this case as a catastrophic, non-recoverable case and to point the Spurious Interrupt Exception handler to the appropriate catastrophic system error entry point.

9.A misaligned data cache lookup with Snoop Invalidate may result in wrong data. The following sequence describes this problem:

1) The 040 performs a misaligned read that spans two cache lines, and the first part of a misaligned data read hits in the first cache line, and the data is used.

2) A snoop occurs on the second cache line associated with the second part of the misaligned read, hence invalidating the second cache line and updating main memory with new data.

3) The second part of the misaligned read misses in the data cache, goes out to main memory and picks up the new data.

Hence, the operand read by the 040 is half old, half new. In this case, the correct data is dependent on the timing of the read relative to the snoop access. The correct answer needs to be the "old" data in its entirety, or the "new" data in its entirety, not the half-new, half-old as supplied by the processor in this case. There are two possible workarounds:

1) Shared memory space must be accessed using aligned operand accesses only

2) Shared memory pages must be accessible to only one master at any one time (implemented via semaphores, and these semaphores must not be misaligned).

10. If a write in copyback space is misaligned so that the operand request spans two cache lines, and a bus error (TEA* asserted) is received on the fetch of the second, third or fourth longword of the second cache line fill, then the Access Error exception handling may not be recoverable without loss of data.Specifically, if subsequent instructions also write to an address that hits in the same first cache line used by the original instruction, then the data written by any number of these subsequent instructions may be lost, until an instruction misses in the cache or the fetch of the second line is terminated by a TEA*. To recover from the bus errors (TEA* asserted), 1) do not allow misaligned operands to span across two cache lines when using the copyback mode, or 2) place a NOP instruction after instructions which have misaligned write operands that span two cache lines in copyback space.

11. If a line transfer is burst inhibited via the assertion of TBI*, three additional longword bus cycles are run by the 040 to complete the original line transfer. Within the tenure of these fake-burst transfers, assertion of TA* during the BCLK edges in which TS* s asserted will result in improper sequencing of the line transfer.

Normally, the 040 ignores TA* during these edges. The workaround is to ensure that TA* is negated whenever TS* is asserted.

The following items are MMU related. Hence, they apply only to the 68040, 68040V, 68LC040.

12. Under limited instruction alignment and tablewalk-related conditions of a MOVES instruction execution resulting in an operand bus error, and the MOVES is immediately followed by a MOVEM instruction, the processor may improperly set the CM bit in the access error frame. When an RTE is eventually encountered to exit the access error handler and to restore this erroneous frame, and that the instruction to be restarted, as pointed to by the stacked program counter is not a MOVEM instruction, pending interrupts will not be reported until the next MOVEM instruction is encountered. The workaround is to place instructions between the MOVES-MOVEM instruction pair.

13.Table and page descriptors must not be placed in cacheable copyback pages. Also, the operating system must not leave page descriptors in pathological combinations such as U=0, M=1. This error condition may cause silent data corruption.

14.The P bit of the TC is undefined out of reset. This bit must be set properly to the desired page size before enabling the MMU.

15. If the last 16 bits of a page is one of the special exception-causing change of flow cases: Illegal, Chk, A-Line, Unimplemented floating-point (type $2 stack frame), and the next page is non-existent (or non-resident, pdt=0), the exception is not reported immediately. Instead, the 040 attempts to prefetch the next instruction on the non-existent (or non-resident) page, resulting in an access fault exception in which the stacked program counter points to the special exception-causing change of flow instruction opcode, and the fault address points to the beginning of the non-existent (or non-resident) page. The workaround is to either avoid the above scenario, or to have the access error handler allocate the non-resident page, execute an RTE to effectively re-start the special exception-causing change of flow instruction.

16.When accessing I/O peripherals that are sensitive to double writes, the following guidelines must be followed:

1) The peripheral must reside in non-cacheable, serialized memory.

2) If possible, use only instructions that can generate one data page fault per instruction.

3) Do not the use of the following instructions: bfclr, bfset, bfins, movem, fmove, fmovem, fsave, movep, movem.

17. If a table walk occurs during exception stacking for a write access which was bus errored, the access error stack frame will incorrectly indicate valid WB1 and WB2 with the same address and data. To avoid a duplicate write, the access error handler must detect this case and discard the WB1 write-back. The pseudo-code is as follows:

if (WB1V==1 && WB2V==1 && WB1S==WB2S && WB1A==WB2A){ WB1V=0; /* clear WB1V, i.e. do not write-back WB1D*/.

}This workaround will not compromise data integrity, nor will it discard intentional multiple writes to serialized space.

18.Under certain circumstances, a MOVE16 write ATC fault improperly invalidates a dirty cache line. The following steps are needed to encounter this item:

1) Assume a physical cache line at address $xxxxxZZZ that is in the data cache and is marked dirty.

2) Execute MOVE16 src,$yyyyyZZZ, where the page descriptor for logical page $yyyyy is marked invalid, but the physical address field in this invalid descriptor (supposed to be undefined) is $xxxxx.

3) The MOVE16 write access results, as it should, in an access error exception since the page descriptor for the MOVE16 write destination is an invalid descriptor. However, the physical address field in the invalid descriptor is incorrectly used as a valid translation of logical page $yyyyy, and the "matching" cache line $xxxxxZZZ is incorrectly invalidated, causing a loss of data. A workaround is to set the physical address field in all invalid descriptors to a physical page which is never mapped in the system. A MOVE16 write fault will never find a matching line in the cache to (incorrectly) invalidate.

The following items are Floating-point related. Hence, they apply to the MC68040 only.

19.A floating point BSUN exception handler must use the PC in the integer stack frame to point to the offending floating point instruction. The value in the FPIAR is invalid.

Literature Distribution Centers:
USA/EUROPE: Motorola Literature Distribution; P.O. Box 20912, Arizona 85036.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.
ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.