

UTMC Errata Sheet

UT80CXX196KD (JD02A and JD02B) CMPL Operation with Wait-States

Anomaly:

For the “A” and “B” revisions of the JD02 die (UT80CXX196KD) (identified by the corresponding markings on the package lid as JD02A and JD02B), there exists an anomaly such that the compare long (CMPL) instruction latches the incorrect comparison data when wait-states are used for instruction fetches. This anomaly occurs when the CMPL instruction has to wait for the second operand to be available in the instruction queue. The data for the first operand is provided for one clock, and therefore gets latched correctly. However, if the instruction queue is not ready with the second operand, the first operand latch continues to be clocked and is compared against invalid data. The only two ways to prevent this anomaly is to avoid using the CMPL instruction, or by ensuring that the instruction queue is always loaded with the correct destination operand before the internal microcode engine begins executing the at-risk instruction.

This failure does not occur for all uses of the CMPL instruction. The following lists the conditions necessary to precondition the UT80CXX196KD into the failure mode described above:

- 1) The UT80CXX196KD must be fetching instructions from either 8- or 16-bit wide memory with at least 1 wait-state inserted.
- 2) The instruction queue must have recently been emptied due to a redirection of the program counter (ie. a jump, interrupt servicing, RET, etc.)

Anomaly Solution:

UTMC will re-spin the UT80CXX196KD design to correct the CMPL anomaly, and the prototype version JD02 Revision C will be available in the November 1999 with flight units ready by the December 1999. If you are using the JD02A or the JD02B, the following list describes available solutions to avoid the CMPL failure:

- 1) Ensure the instruction queue is half-full whenever a CMPL instruction occurs. This may be accomplished by:
 - a) inserting a lengthy instruction (ie. SHL) that consumes a large number of internally processor execution time prior to executing the CMPL instruction.
 - b) ensuring that interrupt service routines do not return to a CMPL instruction by surrounding this instruction with a DI followed by an EI instruction. This protection prevents interrupts from interfering with the program flow until the instruction following EI has completed execution. However, this is hard to implement if the application software is written in C and compiled into assembly with a C compiler.
- 2) Avoid using wait-states for all external instruction memory accesses. As long as the UT80CXX196KD can fetch instructions without wait-states, it will always fill the instruction queue before the internal microcode engine attempts to read the second operand of the CMPL instruction.