

UTMC Errata Sheet

UT80CXX196KD MicroController (JD02B) Wait State Anomaly

Overview:

There exists an anomaly with the “B” revision of the JD02 die (UT80CXX196KD) in that the microcontroller locks-up and stops controlling all memory interface signals (Address/Data bus, ALE, RD, WR, etc.). The anomaly occurs when the UT80CXX196KD attempts to execute a specific set of instructions with register-direct or register-indexed addressing modes as it tries to fetch the destination operand from an empty instruction queue and, there by, locks-up in an idle state with no exit direction until a reset occurs. The three ways to prevent this anomaly are to use program memory with zero wait states, or avoid using a defined set of failure causing instructions, or ensure the instruction queue is always loaded with the correct destination operand before the internal microcode engine begins executing the at-risk instruction.

Background:

The UT80CXX196KD can dynamically access 8- and 16-bit wide instruction memory with any number of wait-states. Because certain instructions can execute faster than others based solely on the addressing mode associated with the instruction, there exists the possibility that the microcontroller can execute the instructions available in the instruction queue faster than the external instruction fetches can fill the queue. Depending on the program flow, the size of memory accesses, and the number of wait-states used, the probability that the internal microcode engine will fetch the destination operand from an empty instruction queue is increased.

UTMC has identified the following two instruction addressing modes that exhibit the controller lock-up anomaly. This anomaly is limited only to instructions whose source operands point to an internal memory location on the UT80CXX196KD (i.e. addresses 0000h through 03FFh):

- 1) Register-indirect addressing instructions (i.e. LD R0, [R1])
- 2) Instructions with register-indexed operands (i.e. ADD R0, 2[R1])
(where R1 points to an address from 0000-03FFh.)

Analysis of the JD02B design shows that the instruction decode state machine has an error where it expects the DESTINATION register address (the R0 part of the opcode in the above examples) to immediately be available after it has completed fetching the SOURCE operand of a register-indirect or -indexed instruction. Specifically, once the instruction decode is complete, the UT80CXX196KD quickly fetches (approx. 1 state clock) the SOURCE data from an internal memory location (0000h through 03FFh). After the SOURCE data has been fetched the UT80CXX196KD looks for the destination of the SOURCE data. If the external opcode fetching is too slow (i.e. wait-states are inserted), the DESTINATION address may not be available and the state machine gets stuck in that particular state.

Furthermore, this failure does not occur for all instructions using the register-direct or register-indirect modes of operations. 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. Data memory accesses with wait states do not exhibit this anomaly.
- 2) The instruction queue must have recently been emptied due to a redirection of the program counter (i.e. a jump, interrupt servicing, RET, etc.) or a string of fast (2 clk) instructions (i.e. NOP).
- 3) A register-indirect or register-indexed instruction is executed with the source data coming from internal memory.

UTMC Errata Sheet

Anomaly Solution:

UTMC plans to correct the Wait State Anomaly in the next revision of the UT80CXX196KD (i.e. version JD02C). The JD02C should be available in December 1999. If you are using the JD02B, the following list describes available solutions to avoid the controller lock-up scenario:

- 1) Restrict usage of register-indirect and -indexed addressing to external memory addresses only. Performing external memory accesses with a register-indirect and -indexed addressing mode allows the UT80CXX196KD to fill the instruction queue sufficiently as it completes execution of the current instruction. However, this is difficult to control if the application code is written in C and compiled into assembly by a C compiler.
- 2) Ensure the instruction queue is half-full whenever a register-indirect or -indexed instruction occurs. This can be accomplished by:
 - a) Inserting a lengthy instruction (i.e., SHL) that consumes a large amount of internal processor execution time prior to executing a register-indirect or -indexed type of instruction.
 - b) Ensuring that interrupt service routines do not return to a register-indirect or -indexed 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. Again, this is hard to implement if the application software is written in C and compiled into assembly with a C compiler.
- 3) 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 destination operand of any instruction.